

Web-Based Music School Manager

Technical Manual

Author: Aaron Redmond

Student ID: C00002396

Supervisor: Paul Barry

Contents

[1 Introduction 11](#_Toc290028557)

[2 Code 12](#_Toc290028558)

[2.1 Application 12](#_Toc290028559)

[2.1.1 configs 12](#_Toc290028560)

[2.1.1.1 Application.ini 12](#_Toc290028561)

[2.1.1.2 db.sql 12](#_Toc290028562)

[2.1.1.3 installed.php 23](#_Toc290028563)

[2.1.2 Forms 24](#_Toc290028564)

[2.1.2.1 Login.php 24](#_Toc290028565)

[2.1.2.2 Request.php 25](#_Toc290028566)

[2.1.2.3 Admin 26](#_Toc290028567)

[2.1.2.3.1 Manage 26](#_Toc290028568)

[2.1.2.3.1.1 Cancelbutton.php 26](#_Toc290028569)

[2.1.2.3.1.2 editgroupstudentcheckboxes.php 27](#_Toc290028570)

[2.1.2.3.1.3 Editroom.php 28](#_Toc290028571)

[2.1.2.3.1.4 Editschool.php 29](#_Toc290028572)

[2.1.2.3.1.5 Editstudent.php 31](#_Toc290028573)

[2.1.2.3.1.6 Editsubject.php 33](#_Toc290028574)

[2.1.2.3.1.7 Editteacher.php 34](#_Toc290028575)

[2.1.2.3.1.8 Gradeselect.php 36](#_Toc290028576)

[2.1.2.3.1.9 Groupadd.php 37](#_Toc290028577)

[2.1.2.3.1.10 Groupconstraints.php 39](#_Toc290028578)

[2.1.2.3.1.11 Groupedit.php 40](#_Toc290028579)

[2.1.2.3.1.12 Groups.php 43](#_Toc290028580)

[2.1.2.3.1.13 Grouptypeedit.php 44](#_Toc290028581)

[2.1.2.3.1.14 Lessonadd.php 45](#_Toc290028582)

[2.1.2.3.1.15 Lessonedit.php 47](#_Toc290028583)

[2.1.2.3.1.16 Lessons.php 49](#_Toc290028584)

[2.1.2.3.1.17 Roomadd.php 50](#_Toc290028585)

[2.1.2.3.1.18 Rooms.php 52](#_Toc290028586)

[2.1.2.3.1.19 School.php 53](#_Toc290028587)

[2.1.2.3.1.20 Studentadd.php 54](#_Toc290028588)

[2.1.2.3.1.21 Students.php 57](#_Toc290028589)

[2.1.2.3.1.22 Subjects.php 58](#_Toc290028590)

[2.1.2.3.1.23 Subjectadd.php 59](#_Toc290028591)

[2.1.2.3.1.24 Subjects.php 60](#_Toc290028592)

[2.1.2.3.1.25 Teacheradd.php 61](#_Toc290028593)

[2.1.2.3.1.26 Teacherconstraints.php 64](#_Toc290028594)

[2.1.2.3.1.27 Teachers.php 65](#_Toc290028595)

[2.1.2.3.2 Myaccount 66](#_Toc290028596)

[2.1.2.3.2.1 Password.php 66](#_Toc290028597)

[2.1.2.3.2.2 Username.php 68](#_Toc290028598)

[2.1.2.3.3 Setup 69](#_Toc290028599)

[2.1.2.3.3.1 Fileupload.php 69](#_Toc290028600)

[2.1.2.3.3.2 Instrumentadd.php 70](#_Toc290028601)

[2.1.2.3.3.3 Instrumentaddbutton.php 71](#_Toc290028602)

[2.1.2.3.3.4 Instrumentedit.php 72](#_Toc290028603)

[2.1.2.3.3.5 News.php 73](#_Toc290028604)

[2.1.2.3.4 Timetabling 74](#_Toc290028605)

[2.1.2.3.4.1 Generate.php 74](#_Toc290028606)

[2.1.2.3.4.2 Information.php 75](#_Toc290028607)

[2.1.2.3.4.3 Printbutton.php 77](#_Toc290028608)

[2.1.2.3.4.4 Subtype.php 78](#_Toc290028609)

[2.1.2.3.4.5 Timetables.php 79](#_Toc290028610)

[2.1.2.3.4.6 newItemEdit.php 80](#_Toc290028611)

[2.1.3 Layouts 81](#_Toc290028612)

[2.1.3.1 Layout.phtml 81](#_Toc290028613)

[2.1.3.2 Login\_layout.phtml 81](#_Toc290028614)

[2.1.3.3 Request\_layout.phtml 82](#_Toc290028615)

[2.1.3.4 Admin 83](#_Toc290028616)

[2.1.3.4.1 Admin\_layout.phtml 83](#_Toc290028617)

[2.1.3.4.2 Admin\_menu.phtml 84](#_Toc290028618)

[2.1.3.4.3 Menu\_script.phtml 86](#_Toc290028619)

[2.1.3.4.4 Personalisation\_layout.phtml 87](#_Toc290028620)

[2.1.3.5 Student 88](#_Toc290028621)

[2.1.3.5.1 menu\_script.phtml 88](#_Toc290028622)

[2.1.3.5.2 student\_layout.phtml 88](#_Toc290028623)

[2.1.3.5.3 student\_menu.phtml 89](#_Toc290028624)

[2.1.3.6 Teacher 90](#_Toc290028625)

[2.1.3.6.1 menu\_script.phtml 90](#_Toc290028626)

[2.1.3.6.2 teacher\_layout.phtml 90](#_Toc290028627)

[2.1.3.6.3 teacher\_menu.phtml 91](#_Toc290028628)

[2.1.4 Models 92](#_Toc290028629)

[2.1.4.1 DAO 92](#_Toc290028630)

[2.1.4.1.1 Abstract.php 92](#_Toc290028631)

[2.1.4.1.2 Groupconstraints.php 93](#_Toc290028632)

[2.1.4.1.3 Groups.php 95](#_Toc290028633)

[2.1.4.1.4 Grouptype.php 98](#_Toc290028634)

[2.1.4.1.5 Instruments.php 101](#_Toc290028635)

[2.1.4.1.6 Lessons.php 104](#_Toc290028636)

[2.1.4.1.7 News.php 108](#_Toc290028637)

[2.1.4.1.8 Result.php 111](#_Toc290028638)

[2.1.4.1.9 Rooms.php 114](#_Toc290028639)

[2.1.4.1.10 Schoolinfo.php 116](#_Toc290028640)

[2.1.4.1.11 Session.php 118](#_Toc290028641)

[2.1.4.1.12 Studentgrades.php 122](#_Toc290028642)

[2.1.4.1.13 Subjects.php 124](#_Toc290028643)

[2.1.4.1.14 Teacherconstraints.php 127](#_Toc290028644)

[2.1.4.1.15 Timetableinfo.php 130](#_Toc290028645)

[2.1.4.1.16 users.php 133](#_Toc290028646)

[2.1.4.2 Entities 140](#_Toc290028647)

[2.1.4.2.1 Groupconstraints.php 140](#_Toc290028648)

[2.1.4.2.2 Groups.php 143](#_Toc290028649)

[2.1.4.2.3 Grouptype.php 147](#_Toc290028650)

[2.1.4.2.4 Instruments.php 149](#_Toc290028651)

[2.1.4.2.5 Lessons.php 152](#_Toc290028652)

[2.1.4.2.6 News.php 156](#_Toc290028653)

[2.1.4.2.7 Rooms.php 159](#_Toc290028654)

[2.1.4.2.8 Schoolinfo.php 162](#_Toc290028655)

[2.1.4.2.9 Session.php 166](#_Toc290028656)

[2.1.4.2.10 Studentgrades.php 170](#_Toc290028657)

[2.1.4.2.11 Subjects.php 172](#_Toc290028658)

[2.1.4.2.12 Teacherconstraints.php 174](#_Toc290028659)

[2.1.4.2.13 Timetableinfo.php 177](#_Toc290028660)

[2.1.4.2.14 users.php 181](#_Toc290028661)

[2.1.5 Proxies 189](#_Toc290028662)

[2.1.6 Modules 189](#_Toc290028663)

[2.1.6.1 Admin 189](#_Toc290028664)

[2.1.6.1.1 Controllers 189](#_Toc290028665)

[2.1.6.1.1.1 IndexController 189](#_Toc290028666)

[2.1.6.1.1.2 manageController 192](#_Toc290028667)

[2.1.6.1.1.3 myaccountController 238](#_Toc290028668)

[2.1.6.1.1.4 setupController 241](#_Toc290028669)

[2.1.6.1.1.5 timetablingController 252](#_Toc290028670)

[2.1.6.1.2 Views 272](#_Toc290028671)

[2.1.6.1.2.1 Index 272](#_Toc290028672)

[2.1.6.1.2.1.1 Help.phtml 272](#_Toc290028673)

[2.1.6.1.2.1.2 Index.phtml 279](#_Toc290028674)

[2.1.6.1.2.1.3 Manage.phtml 279](#_Toc290028675)

[2.1.6.1.2.1.4 Myaccount.phtml 279](#_Toc290028676)

[2.1.6.1.2.1.5 Systemsetup.phtml 280](#_Toc290028677)

[2.1.6.1.2.1.6 Timetabling.phtml 280](#_Toc290028678)

[2.1.6.1.2.1.7 userpanel.phtml 280](#_Toc290028679)

[2.1.6.1.2.2 Manage 281](#_Toc290028680)

[2.1.6.1.2.2.1 \_group-constraint-item.phtml 281](#_Toc290028681)

[2.1.6.1.2.2.2 \_group-item.phtml 282](#_Toc290028682)

[2.1.6.1.2.2.3 \_group-student-item.phtml 283](#_Toc290028683)

[2.1.6.1.2.2.4 \_grouptype-item.phtml 284](#_Toc290028684)

[2.1.6.1.2.2.5 \_instrument-item.phtml 285](#_Toc290028685)

[2.1.6.1.2.2.6 \_lesson-item.phtml 286](#_Toc290028686)

[2.1.6.1.2.2.7 \_room-item.phtml 287](#_Toc290028687)

[2.1.6.1.2.2.8 \_student-item.phtml 288](#_Toc290028688)

[2.1.6.1.2.2.9 \_subject-item.phtml 289](#_Toc290028689)

[2.1.6.1.2.2.10 \_teacher-constraint-item.phtml 290](#_Toc290028690)

[2.1.6.1.2.2.11 \_teacher-item.phtml 291](#_Toc290028691)

[2.1.6.1.2.2.12 editschool.phtml 292](#_Toc290028692)

[2.1.6.1.2.2.13 groupadd.phtml 292](#_Toc290028693)

[2.1.6.1.2.2.14 groupconstraintdelete.phtml 292](#_Toc290028694)

[2.1.6.1.2.2.15 groupconstraints.phtml 293](#_Toc290028695)

[2.1.6.1.2.2.16 groupdelete.phtml 294](#_Toc290028696)

[2.1.6.1.2.2.17 groupedit.phtml 294](#_Toc290028697)

[2.1.6.1.2.2.18 groupeditstudents.phtml 295](#_Toc290028698)

[2.1.6.1.2.2.19 groups.phtml 296](#_Toc290028699)

[2.1.6.1.2.2.20 grouptpyedelete.phtml 296](#_Toc290028700)

[2.1.6.1.2.2.21 grouptype.phtml 297](#_Toc290028701)

[2.1.6.1.2.2.22 grouptypeadd.phtml 298](#_Toc290028702)

[2.1.6.1.2.2.23 grouptypeedit.phtml 298](#_Toc290028703)

[2.1.6.1.2.2.24 lessonadd.phtml 298](#_Toc290028704)

[2.1.6.1.2.2.25 lessondelete.phtml 298](#_Toc290028705)

[2.1.6.1.2.2.26 lessonedit.phtml 298](#_Toc290028706)

[2.1.6.1.2.2.27 lessons.phtml 299](#_Toc290028707)

[2.1.6.1.2.2.28 roomadd.phtml 299](#_Toc290028708)

[2.1.6.1.2.2.29 roomdelete.phtml 300](#_Toc290028709)

[2.1.6.1.2.2.30 roomedit.phtml 300](#_Toc290028710)

[2.1.6.1.2.2.31 rooms.phtml 300](#_Toc290028711)

[2.1.6.1.2.2.32 school.phtml 301](#_Toc290028712)

[2.1.6.1.2.2.33 studentadd.phtml 301](#_Toc290028713)

[2.1.6.1.2.2.34 studentdelete.phtml 301](#_Toc290028714)

[2.1.6.1.2.2.35 studentedit.phtml 302](#_Toc290028715)

[2.1.6.1.2.2.36 studenteditinstruments.phtml 302](#_Toc290028716)

[2.1.6.1.2.2.37 studenteditinstused.phtml 302](#_Toc290028717)

[2.1.6.1.2.2.38 studentingroup.phtml 303](#_Toc290028718)

[2.1.6.1.2.2.39 students.phtml 303](#_Toc290028719)

[2.1.6.1.2.2.40 subjectadd.phtml 304](#_Toc290028720)

[2.1.6.1.2.2.41 subjectdelete.phtml 304](#_Toc290028721)

[2.1.6.1.2.2.42 subjectedit.phtml 304](#_Toc290028722)

[2.1.6.1.2.2.43 subjects.phtml 305](#_Toc290028723)

[2.1.6.1.2.2.44 teacheradd.phtml 305](#_Toc290028724)

[2.1.6.1.2.2.45 teacherconstraintdelete.phtml 305](#_Toc290028725)

[2.1.6.1.2.2.46 teacherconstraints.phtml 306](#_Toc290028726)

[2.1.6.1.2.2.47 teacherdelete.phtml 307](#_Toc290028727)

[2.1.6.1.2.2.48 teacheredit.phtml 307](#_Toc290028728)

[2.1.6.1.2.2.49 teachers.phtml 308](#_Toc290028729)

[2.1.6.1.2.3 Myaccount 309](#_Toc290028730)

[2.1.6.1.2.3.1 Password 309](#_Toc290028731)

[2.1.6.1.2.3.2 Username 309](#_Toc290028732)

[2.1.6.1.2.4 Setup 310](#_Toc290028733)

[2.1.6.1.2.4.1 \_image-list-item.phtml 310](#_Toc290028734)

[2.1.6.1.2.4.2 \_instrument-item.phtml 310](#_Toc290028735)

[2.1.6.1.2.4.3 \_news-item.phtml 311](#_Toc290028736)

[2.1.6.1.2.4.4 instrumentadd.phtml 311](#_Toc290028737)

[2.1.6.1.2.4.5 instrumentdelete.phtml 312](#_Toc290028738)

[2.1.6.1.2.4.6 instrumentedit.phtml 312](#_Toc290028739)

[2.1.6.1.2.4.7 instruments.phtml 312](#_Toc290028740)

[2.1.6.1.2.4.8 instrumentused.phtml 312](#_Toc290028741)

[2.1.6.1.2.4.9 news.phtml 313](#_Toc290028742)

[2.1.6.1.2.4.10 newsarticledelete.phtml 313](#_Toc290028743)

[2.1.6.1.2.4.11 newsitemedit.phtml 313](#_Toc290028744)

[2.1.6.1.2.4.12 personalisation.phtml 313](#_Toc290028745)

[2.1.6.1.2.4.13 useimage.phtml 314](#_Toc290028746)

[2.1.6.1.2.5 Timetabling 315](#_Toc290028747)

[2.1.6.1.2.5.1 \_timetable-subtype-item.phtml 315](#_Toc290028748)

[2.1.6.1.2.5.2 Information.phtml 315](#_Toc290028749)

[2.1.6.1.2.5.3 Timetables.phtml 316](#_Toc290028750)

[2.1.6.2 Default 318](#_Toc290028751)

[2.1.6.2.1 Controllers 318](#_Toc290028752)

[2.1.6.2.1.1 LoginController 318](#_Toc290028753)

[2.1.6.2.1.2 RequestController 321](#_Toc290028754)

[2.1.6.2.2 Views 324](#_Toc290028755)

[2.1.6.2.2.1 \_news-item.phtml 324](#_Toc290028756)

[2.1.6.2.2.2 index.phtml 324](#_Toc290028757)

[2.1.6.3 Student 325](#_Toc290028758)

[2.1.6.3.1 Controllers 325](#_Toc290028759)

[2.1.6.3.1.1 IndexController 325](#_Toc290028760)

[2.1.6.3.1.2 myaccountController 328](#_Toc290028761)

[2.1.6.3.1.3 timetableController 331](#_Toc290028762)

[2.1.6.3.2 Views 334](#_Toc290028763)

[2.1.6.3.2.1 Index 334](#_Toc290028764)

[2.1.6.3.2.1.1 \_news-item.phtml 334](#_Toc290028765)

[2.1.6.3.2.1.2 index.phtml 334](#_Toc290028766)

[2.1.6.3.2.2 Myaccount 335](#_Toc290028767)

[2.1.6.3.2.2.1 Index.phtml 335](#_Toc290028768)

[2.1.6.3.2.2.2 Password.phtml 335](#_Toc290028769)

[2.1.6.3.2.2.3 username.phtml 335](#_Toc290028770)

[2.1.6.3.2.3 timetable 336](#_Toc290028771)

[2.1.6.3.2.3.1 \_timetable-subtype-item.phtml 336](#_Toc290028772)

[2.1.6.3.2.3.2 timetable.phtml 336](#_Toc290028773)

[2.1.6.4 Teacher 337](#_Toc290028774)

[2.1.6.4.1 Controllers 337](#_Toc290028775)

[2.1.6.4.1.1 IndexController.php 337](#_Toc290028776)

[2.1.6.4.1.2 myaccountController.php 340](#_Toc290028777)

[2.1.6.4.1.3 timetableController.php 343](#_Toc290028778)

[2.1.6.4.2 Views 345](#_Toc290028779)

[2.1.6.4.2.1 Index 345](#_Toc290028780)

[2.1.6.4.2.1.1 Help.phtml 345](#_Toc290028781)

[2.1.6.4.2.1.2 \_news-item.phtml 347](#_Toc290028782)

[2.1.6.4.2.1.3 index.phtml 347](#_Toc290028783)

[2.1.6.4.2.2 Myaccount 348](#_Toc290028784)

[2.1.6.4.2.2.1 Index.phtml 348](#_Toc290028785)

[2.1.6.4.2.2.2 Password.phtml 348](#_Toc290028786)

[2.1.6.4.2.2.3 username.phtml 348](#_Toc290028787)

[2.1.6.4.2.3 Timetable 349](#_Toc290028788)

[2.1.6.4.2.3.1 Timetable.phtml 349](#_Toc290028789)

[2.1.6.5 AbstractController 350](#_Toc290028790)

[2.1.7 Validators 351](#_Toc290028791)

[2.1.7.1 Abstract.php 351](#_Toc290028792)

[2.1.7.2 ConfirmPassword.php 352](#_Toc290028793)

[2.1.7.3 Email.php 353](#_Toc290028794)

[2.1.7.4 EmailUnique.php 354](#_Toc290028795)

[2.1.7.5 Password.php 355](#_Toc290028796)

[2.1.7.6 Time.php 357](#_Toc290028797)

[2.1.7.7 User.php 359](#_Toc290028798)

[2.1.7.8 Username.php 361](#_Toc290028799)

[2.2 Public 363](#_Toc290028800)

[2.2.1 Js 363](#_Toc290028801)

[2.2.1.1 easySlider1.7.js 363](#_Toc290028802)

[2.2.2 styles 369](#_Toc290028803)

[2.2.2.1 menu.css 369](#_Toc290028804)

[2.2.2.2 msmstyles.css 372](#_Toc290028805)

[2.2.2.3 PIE.htc 375](#_Toc290028806)

[2.2.2.4 Screen.css 376](#_Toc290028807)

[2.2.3 Index.php 377](#_Toc290028808)

[2.2.4 Install\_2.php 378](#_Toc290028809)

[2.2.5 Install.php 382](#_Toc290028810)

[3 Technical Manual 388](#_Toc290028811)

[3.1 Software Requirements 388](#_Toc290028812)

[3.2 Install Instructions 388](#_Toc290028813)

# Introduction

What follows is the code listings for my project. I have listed the code in the same order as it would appear within the Zend editor.

At the end of the document there is a brief administration manual which demonstrates how to setup the system.

# Code

## Application

### configs

#### Application.ini

This is the file which stores the doctrine database information; it is the file we write to and read from during setup.

#### db.sql

The following dump file is what initially gets loaded into MySQL during database setup.

-- phpMyAdmin SQL Dump

-- version 3.3.3

-- http://www.phpmyadmin.net

--

-- Host: localhost

-- Generation Time: Apr 06, 2011 at 05:34 PM

-- Server version: 5.1.50

-- PHP Version: 5.3.3

**SET** SQL\_MODE="NO\_AUTO\_VALUE\_ON\_ZERO";

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!40101 SET NAMES utf8 \*/;

--

-- Database: `initial\_msm`

--

-- --------------------------------------------------------

--

-- Table structure for table `groupconstraints`

--

**CREATE** **TABLE** IF **NOT** **EXISTS** `groupconstraints` (

`id` **int**(11) **NOT** **NULL** AUTO\_INCREMENT,

`group\_id` **int**(11) **NOT** **NULL**,

`day` **char**(10) **CHARACTER** **SET** latin1 **COLLATE** latin1\_bin **NOT** **NULL**,

`**time**` **char**(15) **CHARACTER** **SET** latin1 **COLLATE** latin1\_bin **NOT** **NULL**,

**KEY** `id` (`id`),

**KEY** `group\_id` (`group\_id`)

) ENGINE=InnoDB **DEFAULT** CHARSET=latin1 AUTO\_INCREMENT=1 ;

--

-- Dumping data for table `groupconstraints`

--

-- --------------------------------------------------------

--

-- Table structure for table `groupparticipants`

--

**CREATE** **TABLE** IF **NOT** **EXISTS** `groupparticipants` (

`group\_id` **int**(11) **NOT** **NULL**,

`**user**` **int**(11) **NOT** **NULL**,

**KEY** `user\_id` (`**user**`),

**KEY** `group\_id` (`group\_id`)

) ENGINE=InnoDB **DEFAULT** CHARSET=latin1;

--

-- Dumping data for table `groupparticipants`

--

-- --------------------------------------------------------

--

-- Table structure for table `groups`

--

**CREATE** **TABLE** IF **NOT** **EXISTS** `groups` (

`group\_id` **int**(11) **NOT** **NULL** AUTO\_INCREMENT,

`group\_name` **char**(255) **CHARACTER** **SET** latin1 **COLLATE** latin1\_bin **NOT** **NULL**,

`group\_short\_name` **char**(100) **CHARACTER** **SET** latin1 **COLLATE** latin1\_bin **NOT** **NULL**,

`group\_type` **int**(11) **NULL**,

**PRIMARY** **KEY** (`group\_id`),

**KEY** `group\_type` (`group\_type`)

) ENGINE=InnoDB **DEFAULT** CHARSET=latin1 AUTO\_INCREMENT=1 ;

--

-- Dumping data for table `groups`

--

-- --------------------------------------------------------

--

-- Table structure for table `grouptype`

--

**CREATE** **TABLE** IF **NOT** **EXISTS** `grouptype` (

`grouptype\_id` **int**(11) **NOT** **NULL** AUTO\_INCREMENT,

`grouptype\_name` **char**(255) **CHARACTER** **SET** latin1 **COLLATE** latin1\_bin **NOT** **NULL**,

**PRIMARY** **KEY** (`grouptype\_id`),

**UNIQUE** **KEY** `grouptype\_name` (`grouptype\_name`)

) ENGINE=InnoDB **DEFAULT** CHARSET=latin1 AUTO\_INCREMENT=1 ;

--

-- Dumping data for table `grouptype`

--

-- --------------------------------------------------------

--

-- Table structure for table `instruments`

--

**CREATE** **TABLE** IF **NOT** **EXISTS** `instruments` (

`instrument\_id` **int**(11) **NOT** **NULL** AUTO\_INCREMENT,

`instrument\_name` **char**(100) **NOT** **NULL**,

`instrument\_used` tinyint(1) **NOT** **NULL**,

**PRIMARY** **KEY** (`instrument\_id`)

) ENGINE=InnoDB **DEFAULT** CHARSET=latin1 AUTO\_INCREMENT=1 ;

--

-- Dumping data for table `instruments`

--

-- --------------------------------------------------------

--

-- Table structure for table `lessons`

--

**CREATE** **TABLE** IF **NOT** **EXISTS** `lessons` (

`lesson\_id` **int**(11) **NOT** **NULL** AUTO\_INCREMENT,

`subject` **int**(11) **NOT** **NULL**,

`group\_id` **int**(11) **NOT** **NULL**,

`teacher` **int**(11) **NOT** **NULL**,

`room` **int**(11) **NOT** **NULL**,

`lesson\_length` **int**(11) **NOT** **NULL**,

`lesson\_count` **int**(11) **NOT** **NULL**,

**PRIMARY** **KEY** (`lesson\_id`),

**KEY** `subject` (`subject`),

**KEY** `**group**` (`group\_id`),

**KEY** `teacher` (`teacher`),

**KEY** `room` (`room`)

) ENGINE=InnoDB **DEFAULT** CHARSET=latin1 AUTO\_INCREMENT=1 ;

--

-- Dumping data for table `lessons`

--

-- --------------------------------------------------------

--

-- Table structure for table `news`

--

**CREATE** **TABLE** IF **NOT** **EXISTS** `news` (

`news\_id` **int**(11) **NOT** **NULL** AUTO\_INCREMENT,

`title` text **CHARACTER** **SET** latin1 **COLLATE** latin1\_bin **NOT** **NULL**,

`news\_item` text **CHARACTER** **SET** latin1 **COLLATE** latin1\_bin **NOT** **NULL**,

`published\_date` **varchar**(50) **NOT** **NULL**,

**PRIMARY** **KEY** (`news\_id`)

) ENGINE=InnoDB **DEFAULT** CHARSET=latin1 AUTO\_INCREMENT=1 ;

--

-- Dumping data for table `news`

--

-- --------------------------------------------------------

--

-- Table structure for table `personalisation`

--

**CREATE** **TABLE** IF **NOT** **EXISTS** `personalisation` (

`personalisation\_id` **int**(11) **NOT** **NULL** AUTO\_INCREMENT,

`image\_path` **char**(255) **CHARACTER** **SET** latin1 **COLLATE** latin1\_bin **NOT** **NULL**,

`image\_used` tinyint(1) **NOT** **NULL**,

**PRIMARY** **KEY** (`personalisation\_id`)

) ENGINE=InnoDB **DEFAULT** CHARSET=latin1 AUTO\_INCREMENT=1 ;

--

-- Dumping data for table `personalisation`

--

-- --------------------------------------------------------

--

-- Table structure for table `rooms`

--

**CREATE** **TABLE** IF **NOT** **EXISTS** `rooms` (

`room\_id` **int**(11) **NOT** **NULL** AUTO\_INCREMENT,

`room\_name` **char**(255) **COLLATE** latin1\_bin **NOT** **NULL**,

`room\_short\_name` **char**(100) **COLLATE** latin1\_bin **NOT** **NULL**,

`capacity` **int**(11) **NOT** **NULL**,

**PRIMARY** **KEY** (`room\_id`)

) ENGINE=InnoDB **DEFAULT** CHARSET=latin1 **COLLATE**=latin1\_bin AUTO\_INCREMENT=2 ;

--

-- Dumping data for table `rooms`

--

**INSERT** **INTO** `rooms` (`room\_id`, `room\_name`, `room\_short\_name`, `capacity`) **VALUES**

(1, 'Any', 'Any', 0);

-- --------------------------------------------------------

--

-- Table structure for table `schoolinfo`

--

**CREATE** **TABLE** IF **NOT** **EXISTS** `schoolinfo` (

`school\_id` **int**(11) **NOT** **NULL** AUTO\_INCREMENT,

`school\_name` **char**(255) **CHARACTER** **SET** latin1 **COLLATE** latin1\_bin **NOT** **NULL**,

`school\_address` **char**(255) **NOT** **NULL**,

`school\_phone` **char**(20) **NOT** **NULL**,

`school\_email` **char**(100) **NOT** **NULL**,

`school\_image` text **NOT** **NULL**,

**PRIMARY** **KEY** (`school\_id`)

) ENGINE=InnoDB **DEFAULT** CHARSET=latin1 AUTO\_INCREMENT=2 ;

--

-- Dumping data for table `schoolinfo`

--

**INSERT** **INTO** `schoolinfo` (`school\_id`, `school\_name`, `school\_address`, `school\_phone`, `school\_email`, `school\_image`) **VALUES**

(1, 'Carlow College of Music', '1 Larkfield, Green Lane, Carlow', '059 9140676', 'info@carlowcollegeofmusic.ie', 'msm\_logo.jpg');

-- --------------------------------------------------------

--

-- Table structure for table `session`

--

**CREATE** **TABLE** IF **NOT** **EXISTS** `session` (

`session\_id` **char**(32) **NOT** **NULL**,

`user\_id` **int**(11) **NOT** **NULL**,

`valid\_til` bigint(20) **NOT** **NULL**,

`created\_at` bigint(20) **NOT** **NULL**,

**PRIMARY** **KEY** (`session\_id`),

**KEY** `user\_id` (`user\_id`)

) ENGINE=InnoDB **DEFAULT** CHARSET=latin1;

--

-- Dumping data for table `session`

--

-- --------------------------------------------------------

--

-- Table structure for table `studentgrades`

--

**CREATE** **TABLE** IF **NOT** **EXISTS** `studentgrades` (

`studentgrade\_id` **int**(11) **NOT** **NULL** AUTO\_INCREMENT,

`student\_id` **int**(11) **NOT** **NULL**,

`instrument\_id` **int**(11) **NOT** **NULL**,

`grade` **int**(11) **NOT** **NULL**,

**PRIMARY** **KEY** (`studentgrade\_id`),

**KEY** `student\_id` (`student\_id`),

**KEY** `instrument\_id` (`instrument\_id`)

) ENGINE=InnoDB **DEFAULT** CHARSET=latin1 AUTO\_INCREMENT=1 ;

--

-- Dumping data for table `studentgrades`

--

-- --------------------------------------------------------

--

-- Table structure for table `subjects`

--

**CREATE** **TABLE** IF **NOT** **EXISTS** `subjects` (

`subject\_id` **int**(11) **NOT** **NULL** AUTO\_INCREMENT,

`subject` **char**(255) **CHARACTER** **SET** latin1 **COLLATE** latin1\_bin **NOT** **NULL**,

**PRIMARY** **KEY** (`subject\_id`),

**UNIQUE** **KEY** `subject` (`subject`)

) ENGINE=InnoDB **DEFAULT** CHARSET=latin1 AUTO\_INCREMENT=1 ;

--

-- Dumping data for table `subjects`

--

-- --------------------------------------------------------

--

-- Table structure for table `subjectstaught`

--

**CREATE** **TABLE** IF **NOT** **EXISTS** `subjectstaught` (

`subjectstaught\_id` **int**(11) **NOT** **NULL** AUTO\_INCREMENT,

`teacher\_id` **int**(11) **NOT** **NULL**,

`subjects\_id` **int**(11) **NOT** **NULL**,

**PRIMARY** **KEY** (`subjectstaught\_id`),

**KEY** `teacher\_id` (`teacher\_id`),

**KEY** `subjects\_id` (`subjects\_id`)

) ENGINE=InnoDB **DEFAULT** CHARSET=latin1 AUTO\_INCREMENT=1 ;

--

-- Dumping data for table `subjectstaught`

--

-- --------------------------------------------------------

--

-- Table structure for table `teacherconstraints`

--

**CREATE** **TABLE** IF **NOT** **EXISTS** `teacherconstraints` (

`id` **int**(11) **NOT** **NULL** AUTO\_INCREMENT,

`teacher\_id` **int**(11) **NOT** **NULL**,

`day` **char**(10) **CHARACTER** **SET** latin1 **COLLATE** latin1\_bin **NOT** **NULL**,

`**time**` **char**(15) **CHARACTER** **SET** latin1 **COLLATE** latin1\_bin **NOT** **NULL**,

**PRIMARY** **KEY** (`id`),

**KEY** `teacher\_id` (`teacher\_id`)

) ENGINE=InnoDB **DEFAULT** CHARSET=latin1 AUTO\_INCREMENT=1 ;

--

-- Dumping data for table `teacherconstraints`

--

-- --------------------------------------------------------

--

-- Table structure for table `timetableinfo`

--

**CREATE** **TABLE** IF **NOT** **EXISTS** `timetableinfo` (

`ttinfo\_id` **int**(11) **NOT** **NULL**,

`academic\_year` **char**(15) **COLLATE** latin1\_bin **NOT** **NULL**,

`days\_per\_week` tinyint(1) **NOT** **NULL**,

`lessons\_per\_day` tinyint(1) **NOT** **NULL**,

`start\_time` **varchar**(5) **COLLATE** latin1\_bin **NOT** **NULL**,

`lesson\_length` **varchar**(5) **COLLATE** latin1\_bin **NOT** **NULL**,

**PRIMARY** **KEY** (`ttinfo\_id`)

) ENGINE=InnoDB **DEFAULT** CHARSET=latin1 **COLLATE**=latin1\_bin;

--

-- Dumping data for table `timetableinfo`

--

-- --------------------------------------------------------

--

-- Table structure for table `users`

--

**CREATE** **TABLE** IF **NOT** **EXISTS** `users` (

`user\_id` **int**(11) **NOT** **NULL** AUTO\_INCREMENT,

`username` **varchar**(20) **COLLATE** latin1\_bin **NOT** **NULL** **DEFAULT** '',

`user\_password` **varchar**(32) **COLLATE** latin1\_bin **NOT** **NULL** **DEFAULT** '',

`user\_forename` **char**(255) **COLLATE** latin1\_bin **NOT** **NULL**,

`user\_surname` **char**(255) **COLLATE** latin1\_bin **NOT** **NULL**,

`user\_email` **varchar**(255) **COLLATE** latin1\_bin **NOT** **NULL** **DEFAULT** '',

`user\_address` **varchar**(255) **COLLATE** latin1\_bin **NOT** **NULL**,

`user\_telephone` **varchar**(50) **COLLATE** latin1\_bin **NOT** **NULL**,

`user\_mobile` **varchar**(50) **COLLATE** latin1\_bin **NOT** **NULL**,

`user\_type` **char**(10) **COLLATE** latin1\_bin **NOT** **NULL**,

`user\_verified` tinyint(1) **NOT** **NULL**,

**PRIMARY** **KEY** (`user\_id`),

**UNIQUE** **KEY** `user\_email` (`user\_email`)

) ENGINE=InnoDB **DEFAULT** CHARSET=latin1 **COLLATE**=latin1\_bin AUTO\_INCREMENT=2 ;

--

-- Dumping data for table `users`

--

**INSERT** **INTO** `users` (`user\_id`, `username`, `user\_password`, `user\_forename`, `user\_surname`, `user\_email`, `user\_address`, `user\_telephone`, `user\_mobile`, `user\_type`, `user\_verified`) **VALUES**

(1, 'administrator', MD5('administrator'), '', '', '', '', '', '', 'admin', 1);

--

-- Constraints for dumped tables

--

--

-- Constraints for table `groupconstraints`

--

**ALTER** **TABLE** `groupconstraints`

**ADD** **CONSTRAINT** `groupconstraints\_ibfk\_1` **FOREIGN** **KEY** (`group\_id`) **REFERENCES** `groups` (`group\_id`) **ON** **DELETE** **CASCADE** **ON** **UPDATE** **NO** **ACTION**;

--

-- Constraints for table `groupparticipants`

--

**ALTER** **TABLE** `groupparticipants`

**ADD** **CONSTRAINT** `groupparticipants\_ibfk\_4` **FOREIGN** **KEY** (`**user**`) **REFERENCES** `users` (`user\_id`) **ON** **DELETE** **CASCADE** **ON** **UPDATE** **NO** **ACTION**,

**ADD** **CONSTRAINT** `groupparticipants\_ibfk\_5` **FOREIGN** **KEY** (`group\_id`) **REFERENCES** `groups` (`group\_id`);

--

-- Constraints for table `groups`

--

**ALTER** **TABLE** `groups`

**ADD** **CONSTRAINT** `groups\_ibfk\_1` **FOREIGN** **KEY** (`group\_type`) **REFERENCES** `grouptype` (`grouptype\_id`) **ON** **DELETE** **CASCADE** **ON** **UPDATE** **CASCADE**;

--

-- Constraints for table `lessons`

--

**ALTER** **TABLE** `lessons`

**ADD** **CONSTRAINT** `lessons\_ibfk\_1` **FOREIGN** **KEY** (`subject`) **REFERENCES** `subjects` (`subject\_id`) **ON** **DELETE** **CASCADE** **ON** **UPDATE** **NO** **ACTION**,

**ADD** **CONSTRAINT** `lessons\_ibfk\_3` **FOREIGN** **KEY** (`teacher`) **REFERENCES** `users` (`user\_id`) **ON** **DELETE** **CASCADE** **ON** **UPDATE** **NO** **ACTION**,

**ADD** **CONSTRAINT** `lessons\_ibfk\_4` **FOREIGN** **KEY** (`room`) **REFERENCES** `rooms` (`room\_id`) **ON** **DELETE** **CASCADE** **ON** **UPDATE** **NO** **ACTION**,

**ADD** **CONSTRAINT** `lessons\_ibfk\_5` **FOREIGN** **KEY** (`group\_id`) **REFERENCES** `groups` (`group\_id`) **ON** **DELETE** **CASCADE** **ON** **UPDATE** **CASCADE**;

--

-- Constraints for table `session`

--

**ALTER** **TABLE** `session`

**ADD** **CONSTRAINT** `session\_ibfk\_1` **FOREIGN** **KEY** (`user\_id`) **REFERENCES** `users` (`user\_id`) **ON** **DELETE** **CASCADE** **ON** **UPDATE** **NO** **ACTION**;

--

-- Constraints for table `studentgrades`

--

**ALTER** **TABLE** `studentgrades`

**ADD** **CONSTRAINT** `studentgrades\_ibfk\_4` **FOREIGN** **KEY** (`student\_id`) **REFERENCES** `users` (`user\_id`) **ON** **DELETE** **CASCADE** **ON** **UPDATE** **NO** **ACTION**,

**ADD** **CONSTRAINT** `studentgrades\_ibfk\_5` **FOREIGN** **KEY** (`instrument\_id`) **REFERENCES** `instruments` (`instrument\_id`) **ON** **DELETE** **CASCADE** **ON** **UPDATE** **NO** **ACTION**;

--

-- Constraints for table `subjectstaught`

--

**ALTER** **TABLE** `subjectstaught`

**ADD** **CONSTRAINT** `subjectstaught\_ibfk\_1` **FOREIGN** **KEY** (`teacher\_id`) **REFERENCES** `users` (`user\_id`) **ON** **DELETE** **CASCADE** **ON** **UPDATE** **CASCADE**,

**ADD** **CONSTRAINT** `subjectstaught\_ibfk\_2` **FOREIGN** **KEY** (`subjects\_id`) **REFERENCES** `subjects` (`subject\_id`) **ON** **DELETE** **CASCADE** **ON** **UPDATE** **CASCADE**;

--

-- Constraints for table `teacherconstraints`

--

**ALTER** **TABLE** `teacherconstraints`

**ADD** **CONSTRAINT** `teacherconstraints\_ibfk\_1` **FOREIGN** **KEY** (`teacher\_id`) **REFERENCES** `users` (`user\_id`) **ON** **DELETE** **CASCADE** **ON** **UPDATE** **NO** **ACTION**;

#### installed.php

This is an empty placeholder files which is created once the system has been successfully installed.

### Forms

The following section contains all the applications form code, this is the code which creates the buttons, text boxes, etc.

#### Login.php

<?php

/\*\*

\*

\* Create the login form

\* @author Aaron Redmond

\* @version 1.0

\* @created 07-Feb-2011

\*/

class Form\_Login extends Zend\_Form

{

private $username;

private $password;

private $submitButton;

function \_\_construct ()

{

parent::\_\_construct();

$this->username = new Zend\_Form\_Element\_Text('username');

$this->username->setLabel('User Name');

$this->username->setAutoInsertNotEmptyValidator(false);

$this->username->setRequired();

$this->username->addValidator(new Validate\_User());

$this->addElement($this->username);

$this->password = new Zend\_Form\_Element\_Password('password');

$this->password->setLabel('Password');

$this->password->setAutoInsertNotEmptyValidator(false);

$this->password->setRequired();

$this->password->addValidator(new Validate\_Password());

$this->addElement($this->password);

$this->submitButton = new Zend\_Form\_Element\_Submit('submit');

$this->submitButton->setLabel('Login');

$this->submitButton->setIgnore(true);

$this->addElement($this->submitButton);

}

}

?>

#### Request.php

<?php

/\*\*

\*

\* The Request Username/Password Form

\* @author Aaron Redmond

\* @version 1.0

\* @created 10-Feb-2011

\*/

class Form\_Request extends Zend\_Form

{

private $email;

private $requestButton;

function \_\_construct()

{

parent::\_\_construct();

$this->email = new Zend\_Form\_Element\_Text('email');

$this->email->setAttrib('size', '40');

$this->email->setLabel('E-mail');

$this->email->setAutoInsertNotEmptyValidator(false)->setRequired();

$this->email->addValidator(new Validate\_Email());

$this->addElement($this->email);

$this->requestButton = new Zend\_Form\_Element\_Submit('request');

$this->requestButton->setLabel('Reqest')->setIgnore(true);

$this->addElement($this->requestButton);

}

}

?>

#### Admin

##### Manage

###### Cancelbutton.php

<?php

/\*\*

\* Create Cancel Buttomn for use on all Admin->Manage pages

\* @author Aaron Redmond

\* @version 1.0

\* @created 02-Mar-2011

\*

\*/

class Form\_Admin\_manage\_cancelbutton extends Zend\_Form

{

private $cancelButton;

function \_\_construct($urlLinkBack)

{

parent::\_\_construct();

$this->cancelButton = new Zend\_Form\_Element\_Button('cancelButton');

$this->cancelButton->setLabel('Back');

$this->cancelButton->setAttrib('onclick', 'location.href='.$urlLinkBack);

$this->addElement($this->cancelButton);

}

}

?>

###### editgroupstudentcheckboxes.php

This file is no longer in use.

<?php

/\*\*

\* NO LONGER USED

\*

\* Create Checkboxes for all Students

\* Only check for current students in group

\* for Admin->Manage->Edit Group->Edit Students page

\* @author Aaron Redmond

\* @version 1.0

\* @created 03-Mar-2011

\*

\*/

class Form\_Admin\_manage\_editgroupstudentcheckboxes extends Zend\_Form

{

private $studentCheckbox;

private $saveButton;

function \_\_construct ($studentsAll, $studentsInGroup)

{

parent::\_\_construct();

foreach ($studentsAll as $student)

{

$checked = false;

foreach ($studentsInGroup as $studentInGroup)

{

If ($studentInGroup->getUserId() == $student->getUserId())

{

$checked = true;

}

}

$this->studentCheckbox = new Zend\_Form\_Element\_Checkbox('studentCheckbox' . "\_" . $student->getUserId());

$this->studentCheckbox->setLabel($student->getUserForename() . ' ' . $student->getUserSurname());

$this->studentCheckbox->checked = $checked;

$this->addElement($this->studentCheckbox);

}

$this->saveButton = new Zend\_Form\_Element\_Submit('saveButton');

$this->saveButton->setLabel('Save');

$this->addElement($this->saveButton);

}

}

?>

###### Editroom.php

<?php

/\*\*

\* Create Edit button for Admin->Manage->editroom page

\* @author Aaron Redmond

\* @version 1.0

\* @created 24-Feb-2011

\* @version 1.1

\* @created 22-Mar-2011 - Added Capacity

\*

\*/

class Form\_Admin\_manage\_editroom extends Zend\_Form

{

private $roomNameEditBox;

private $shortNameEditBox;

private $capacityEditBox;

private $submitButton;

private $cancelButton;

function \_\_construct($roomName, $roomShortAddress, $capacity)

{

parent::\_\_construct();

$this->roomNameEditBox = new Zend\_Form\_Element\_Text('roomNameEditBox');

$this->roomNameEditBox->setLabel('Room Name')

->setAutoInsertNotEmptyValidator(true)

->setRequired()

->setValue($roomName)

->setAttrib('size', '40');

$this->addElement($this->roomNameEditBox);

$this->shortNameEditBox = new Zend\_Form\_Element\_Text('shortNameEditBox');

$this->shortNameEditBox->setLabel('Room Short Name')

->setAutoInsertNotEmptyValidator(true)

->setRequired()

->setValue($roomShortAddress)

->setAttrib('size', '40');

$this->addElement($this->shortNameEditBox);

$this->capacityEditBox = new Zend\_Form\_Element\_Text('capacityEditBox');

$this->capacityEditBox->setLabel('Capacity')

->setAutoInsertNotEmptyValidator(true)

->setRequired()

->setValue($capacity);

$this->addElement($this->capacityEditBox);

$this->submitButton = new Zend\_Form\_Element\_Submit('submit');

$this->submitButton->setLabel('Save')

->setIgnore(true);

$this->addElement($this->submitButton);

$this->cancelButton = new Zend\_Form\_Element\_Button('cancelButton');

$this->cancelButton->setLabel('Back')

->setAttrib('onclick', 'location.href="/admin/manage/rooms"');

$this->addElement($this->cancelButton);

}

}

?>

###### Editschool.php

<?php

/\*\*

\* Create Edit button for Admin->Manage->editSchool page

\* @author Aaron Redmond

\* @version 1.0

\* @created 22-Feb-2011

\*

\*/

class Form\_Admin\_manage\_editschool extends Zend\_Form

{

private $nameEditBox;

private $addressEditBox;

private $phoneEditBox;

private $emailEditBox;

private $submitButton;

private $cancelButton;

function \_\_construct($schoolName, $schoolAddress, $schoolPhone, $schoolEmail)

{

parent::\_\_construct();

$this->nameEditBox = new Zend\_Form\_Element\_Text('nameEditBox');

$this->nameEditBox->setLabel('Name')

->setAutoInsertNotEmptyValidator(true)

->setRequired()

->setValue($schoolName)

->setAttrib('size', '50');

$this->addElement($this->nameEditBox);

$this->addressEditBox = new Zend\_Form\_Element\_Text('addressEditBox');

$this->addressEditBox->setLabel('Address')

->setAutoInsertNotEmptyValidator(true)

->setRequired()

->setValue($schoolAddress)

->setAttrib('size', '50');

$this->addElement($this->addressEditBox);

$this->phoneEditBox = new Zend\_Form\_Element\_Text('phoneEditBox');

$this->phoneEditBox->setLabel('Telephone')

->setAutoInsertNotEmptyValidator(true)

->setRequired()

->setValue($schoolPhone)

->setAttrib('size', '50');

$this->addElement($this->phoneEditBox);

$this->emailEditBox = new Zend\_Form\_Element\_Text('emailEditBox');

$this->emailEditBox->setLabel('E-mail')

->setAutoInsertNotEmptyValidator(false)

->setRequired()

->addValidator(new Validate\_Email())

->setValue($schoolEmail)

->setAttrib('size', '50');

$this->addElement($this->emailEditBox);

$this->submitButton = new Zend\_Form\_Element\_Submit('submit');

$this->submitButton->setLabel('Save')

->setIgnore(true);

$this->addElement($this->submitButton);

$this->cancelButton = new Zend\_Form\_Element\_Button('cancelButton');

$this->cancelButton->setLabel('Back');

$this->cancelButton->setAttrib('onclick', 'location.href="/admin/manage/school"');

$this->addElement($this->cancelButton);

}

}

?>

###### Editstudent.php

<?php

/\*\*

\* Create Form for Admin->Manage->editteacher page

\* @author Aaron Redmond

\* @version 1.0

\* @created 26-Feb-2011

\*

\*/

class Form\_Admin\_manage\_editstudent extends Zend\_Form

{

private $forenameEditBox;

private $surnameEditBox;

private $emailEditBox;

private $addressEditBox;

private $phoneEditBox;

private $mobileEditBox;

private $submitButton;

private $cancelButton;

function \_\_construct($forename, $surname, $email, $address, $phone, $mobile)

{

parent::\_\_construct();

$this->forenameEditBox = new Zend\_Form\_Element\_Text('forenameEditBox');

$this->forenameEditBox->setLabel('Forename')

->setAutoInsertNotEmptyValidator(true)

->setRequired()

->setValue($forename)

->setAttrib('size', '50');

$this->addElement($this->forenameEditBox);

$this->surnameEditBox = new Zend\_Form\_Element\_Text('surnameEditBox');

$this->surnameEditBox->setLabel('Surname')

->setAutoInsertNotEmptyValidator(true)

->setRequired()

->setValue($surname)

->setAttrib('size', '50');

$this->addElement($this->surnameEditBox);

$this->emailEditBox = new Zend\_Form\_Element\_Text('emailEditBox');

$this->emailEditBox->setLabel('E-mail')

->setAutoInsertNotEmptyValidator(true)

->setRequired()

->setValue($email)

->setAttrib('size', '50');

$this->addElement($this->emailEditBox);

$this->addressEditBox = new Zend\_Form\_Element\_Text('addressEditBox');

$this->addressEditBox->setLabel('Address')

->setAutoInsertNotEmptyValidator(true)

->setRequired()

->setValue($address)

->setAttrib('size', '50');

$this->addElement($this->addressEditBox);

$this->phoneEditBox = new Zend\_Form\_Element\_Text('phoneEditBox');

$this->phoneEditBox->setLabel('Telephone')

->setAutoInsertNotEmptyValidator(false)

->setRequired()

->setValue($phone);

$this->addElement($this->phoneEditBox);

$this->mobileEditBox = new Zend\_Form\_Element\_Text('mobileEditBox');

$this->mobileEditBox->setLabel('Mobile')

->setAutoInsertNotEmptyValidator(false)

->setRequired()

->setValue($mobile);

$this->addElement($this->mobileEditBox);

$this->submitButton = new Zend\_Form\_Element\_Submit('submit');

$this->submitButton->setLabel('Save')

->setIgnore(true);

$this->addElement($this->submitButton);

$this->cancelButton = new Zend\_Form\_Element\_Button('cancelButton');

$this->cancelButton->setLabel('Back')

->setAttrib('onclick', 'location.href="/admin/manage/students"');

$this->addElement($this->cancelButton);

}

}

?>

###### Editsubject.php

<?php

/\*\*

\* Create Edit button for Admin->Manage->editsubject page

\* @author Aaron Redmond

\* @version 1.0

\* @created 25-Feb-2011

\*

\*/

class Form\_Admin\_manage\_editsubject extends Zend\_Form

{

private $subjectEditBox;

private $submitButton;

private $cancelButton;

function \_\_construct($subject)

{

parent::\_\_construct();

$this->subjectEditBox = new Zend\_Form\_Element\_Text('subjectEditBox');

$this->subjectEditBox->setLabel('Subject')

->setAutoInsertNotEmptyValidator(true)

->setRequired()

->setValue($subject)

->setAttrib('size', '40');

$this->addElement($this->subjectEditBox);

$this->submitButton = new Zend\_Form\_Element\_Submit('submit');

$this->submitButton->setLabel('Save')

->setIgnore(true);

$this->addElement($this->submitButton);

$this->cancelButton = new Zend\_Form\_Element\_Button('cancelButton');

$this->cancelButton->setLabel('Back')

->setAttrib('onclick', 'location.href="/admin/manage/subjects"');

$this->addElement($this->cancelButton);

}

}

?>

###### Editteacher.php

<?php

/\*\*

\* Create Form for Admin->Manage->editteacher page

\* @author Aaron Redmond

\* @version 1.0

\* @created 25-Feb-2011

\*

\*/

class Form\_Admin\_manage\_editteacher extends Zend\_Form

{

private $forenameEditBox;

private $surnameEditBox;

private $emailEditBox;

private $addressEditBox;

private $phoneEditBox;

private $mobileEditBox;

private $submitButton;

private $cancelButton;

function \_\_construct($forename, $surname, $email, $address, $phone, $mobile)

{

parent::\_\_construct();

$this->forenameEditBox = new Zend\_Form\_Element\_Text('forenameEditBox');

$this->forenameEditBox->setLabel('Forename')

->setAutoInsertNotEmptyValidator(true)

->setRequired()

->setValue($forename)

->setAttrib('size', '50');

$this->addElement($this->forenameEditBox);

$this->surnameEditBox = new Zend\_Form\_Element\_Text('surnameEditBox');

$this->surnameEditBox->setLabel('Surname')

->setAutoInsertNotEmptyValidator(true)

->setRequired()

->setValue($surname)

->setAttrib('size', '50');

$this->addElement($this->surnameEditBox);

$this->emailEditBox = new Zend\_Form\_Element\_Text('emailEditBox');

$this->emailEditBox->setLabel('E-mail')

->setAutoInsertNotEmptyValidator(true)

->setRequired()

->setValue($email)

->setAttrib('size', '50');

$this->addElement($this->emailEditBox);

$this->addressEditBox = new Zend\_Form\_Element\_Text('addressEditBox');

$this->addressEditBox->setLabel('Address')

->setAutoInsertNotEmptyValidator(true)

->setRequired()

->setValue($address)

->setAttrib('size', '50');

$this->addElement($this->addressEditBox);

$this->phoneEditBox = new Zend\_Form\_Element\_Text('phoneEditBox');

$this->phoneEditBox->setLabel('Telephone')

->setAutoInsertNotEmptyValidator(false)

->setRequired()

->setValue($phone);

$this->addElement($this->phoneEditBox);

$this->mobileEditBox = new Zend\_Form\_Element\_Text('mobileEditBox');

$this->mobileEditBox->setLabel('Mobile')

->setAutoInsertNotEmptyValidator(false)

->setRequired()

->setValue($mobile);

$this->addElement($this->mobileEditBox);

$this->submitButton = new Zend\_Form\_Element\_Submit('submit');

$this->submitButton->setLabel('Save')

->setIgnore(true);

$this->addElement($this->submitButton);

$this->cancelButton = new Zend\_Form\_Element\_Button('cancelButton');

$this->cancelButton->setLabel('Back')

->setAttrib('onclick', 'location.href="/admin/manage/teachers"');

$this->addElement($this->cancelButton);

}

}

?>

###### Gradeselect.php

<?php

/\*\*

\* Create Grade Options Select

\* @author Aaron Redmond

\* @version 1.0

\* @created 05-Mar-2011

\*

\*/

class Form\_Admin\_manage\_gradeselect extends Zend\_Form

{

private $optionSelect;

function \_\_construct($grades, $grade, $studentID, $instrumentid)

{

parent::\_\_construct();

$id = $studentID . '\_' . $instrumentid;

$this->optionSelect = new Zend\_Form\_Element\_Select('optionSelect' . $id);

$this->optionSelect->addMultiOptions($grades)

->setOptions(array('onChange' => 'submit()','id'=>$id))

->setValue($grade);

$this->addElement($this->optionSelect);

}

}

?>

###### Groupadd.php

<?php

/\*\*

\* Create Form for Admin->Manage->Groups->Add page

\* @author Aaron Redmond

\* @version 1.0

\* @created 29-Mar-2011

\*

\*/

class Form\_Admin\_manage\_groupadd extends Zend\_Form

{

private $groupName;

private $groupShortName;

private $groupType;

private $submitButton;

function \_\_construct($groupTypes)

{

parent::\_\_construct();

$this->groupName = new Zend\_Form\_Element\_Text('GroupName');

$this->groupName->setLabel('Group Name')

->setAutoInsertNotEmptyValidator(true)

->setRequired();

$this->addElement($this->groupName);

$this->groupShortName = new Zend\_Form\_Element\_Text('groupShortName');

$this->groupShortName->setLabel('Group Short Name')

->setAutoInsertNotEmptyValidator(true)

->setRequired();

$this->addElement($this->groupShortName);

$groupTypesArray = array();

foreach ($groupTypes as $groupType)

{

$groupTypesArray[$groupType->getGroupTypeId()] = $groupType->getGroupTypeName();

}

$this->groupType = new Zend\_Form\_Element\_Select('groupType');

$this->groupType->setLabel('Group Type');

$this->groupType->setAutoInsertNotEmptyValidator(false);

//$this->groupType->setRequired();

if ($groupTypes != null){

$this->groupType->addMultiOptions($groupTypesArray);

}

$this->addElement($this->groupType);

$this->submitButton = new Zend\_Form\_Element\_Submit('submit');

$this->submitButton->setLabel('Save')

->setIgnore(true);

$this->addElement($this->submitButton);

}

}

###### Groupconstraints.php

<?php

/\*\*

\* Create Add group constraints form

\*

\* @author Aaron Redmond

\* @version 1.0

\* @created 02-Apr-2011

\*

\*/

class Form\_Admin\_manage\_groupconstraints extends Zend\_Form

{

private $days;

private $times;

private $saveButton;

function \_\_construct ($daysArray, $timesArray)

{

parent::\_\_construct();

$this->days = new Zend\_Form\_Element\_Select('days');

$this->days->setLabel('Day');

$this->days->addMultiOptions($daysArray);

$this->addElement($this->days);

$this->times = new Zend\_Form\_Element\_Select('times');

$this->times->setLabel('Time');

$this->times->addMultiOptions($timesArray);

$this->addElement($this->times);

$this->saveButton = new Zend\_Form\_Element\_Submit('saveButton');

$this->saveButton->setLabel('Add');

$this->addElement($this->saveButton);

}

}

?>

###### Groupedit.php

<?php

/\*\*

\* Create Edit Students for Admin->Manage->editgroup page

\* @author Aaron Redmond

\* @version 1.0

\* @created 04-Mar-2011

\*

\*/

class Form\_Admin\_manage\_groupedit extends Zend\_Form

{

private $groupName;

private $groupShortName;

private $groupType;

private $groupTypeEdit;

private $groupStudents;

private $editStudentButton;

private $submitButton;

function \_\_construct($Name, $ShortName, $groupTypes, $groupTypeID, $students, $groupID)

{

parent::\_\_construct();

$this->groupName = new Zend\_Form\_Element\_Text('GroupName');

$this->groupName->setLabel('Group Name')

->setAutoInsertNotEmptyValidator(true)

->setRequired()

->setValue($Name);

$this->addElement($this->groupName);

$this->groupShortName = new Zend\_Form\_Element\_Text('groupShortName');

$this->groupShortName->setLabel('Group Short Name')

->setAutoInsertNotEmptyValidator(true)

->setRequired()

->setValue($ShortName);

$this->addElement($this->groupShortName);

//Select element containing the different group types

//The selected item is the group type associated with the current group being edited

$groupTypesArray = array();

foreach ($groupTypes as $groupType)

{

$groupTypesArray[$groupType->getGroupTypeId()] = $groupType->getGroupTypeName();

}

$this->groupType = new Zend\_Form\_Element\_Select('groupType');

$this->groupType->setLabel('Group Type')

->setAutoInsertNotEmptyValidator(false)

->setRequired()

->addMultiOptions($groupTypesArray)

->setValue($groupTypeID);

$this->addElement($this->groupType);

//Button for Editing Group Types

//Supply Linkback Url for group we are currently editing

$url = "\"/admin/manage/grouptype/id/". $groupID . "\"";

$this->groupTypeEdit = new Zend\_Form\_Element\_Button('editGroupType');

$this->groupTypeEdit->setLabel('Edit Group Type')

->setAttrib('onclick', 'location.href='.$url);

$this->addElement($this->groupTypeEdit);

//Students In Group

$studentArray = array();

foreach ($students as $student)

{

$studentArray[$student->getUserId()] = $student->getUserForename() . ' ' . $student->getUserSurname();

}

$this->groupStudents = new Zend\_Form\_Element\_Multiselect('groupStudents');

$this->groupStudents->setLabel('Students')

->addMultiOptions($studentArray);

$this->addElement($this->groupStudents);

//Button for Editing Students in Group

//Supply Linkback Url for group we are currently editing

$url2 = "\"/admin/manage/groupeditstudents/id/". $groupID . "\"";

$this->editStudentButton = new Zend\_Form\_Element\_Button('editStudentButton');

$this->editStudentButton->setLabel('Edit Students')

->setAttrib('onclick', 'location.href='.$url2);

$this->addElement($this->editStudentButton);

//Button for submitting the edited group

$this->submitButton = new Zend\_Form\_Element\_Submit('submit');

$this->submitButton->setLabel('Save')

->setIgnore(true);

$this->addElement($this->submitButton);

}

}

?>

###### Groups.php

<?php

/\*\*

\* Create form for Admin->Manage->groups page

\* @author Aaron Redmond

\* @version 1.0

\* @created 27-Feb-2011

\*

\*/

class Form\_Admin\_manage\_groups extends Zend\_Form

{

private $addButton;

function \_\_construct ()

{

parent::\_\_construct();

$this->addButton = new Zend\_Form\_Element\_Button('addButton');

$this->addButton->setLabel('Add')

->setAttrib('onclick', 'location.href="/admin/manage/groupadd"');

$this->addElement($this->addButton);

}

}

?>

###### Grouptypeedit.php

<?php

/\*\*

\* Create Edit Form for Admin->Manage->Groups->Group Type edit page

\* @author Aaron Redmond

\* @version 1.0

\* @created 14-Mar-2011

\*

\*/

class Form\_Admin\_manage\_grouptypeedit extends Zend\_Form

{

private $groupType;

private $submitButton;

private $cancelButton;

function \_\_construct($groupType, $groupId)

{

parent::\_\_construct();

$this->groupType = new Zend\_Form\_Element\_Text('groupType');

$this->groupType->setLabel('Group Type')

->setAutoInsertNotEmptyValidator(true)

->setRequired()

->setValue($groupType);

$this->addElement($this->groupType);

$this->submitButton = new Zend\_Form\_Element\_Submit('submit');

$this->submitButton->setLabel('Save')

->setIgnore(true);

$this->addElement($this->submitButton);

$url = "\"/admin/manage/grouptype/id/". $groupId . "\"";

$this->cancelButton = new Zend\_Form\_Element\_Button('cancelButton');

$this->cancelButton->setLabel('Back')

->setAttrib('onclick', 'location.href='.$url);

$this->addElement($this->cancelButton);

}

}

?>

###### Lessonadd.php

<?php

/\*\*

\* Create Add Form for Admin->Manage->Lessons add page

\* @author Aaron Redmond

\* @version 1.0

\* @created 13-Mar-2011

\*

\*/

class Form\_Admin\_manage\_lessonadd extends Zend\_Form

{

private $subject;

private $group;

private $teacher;

private $room;

private $lessonLength;

private $lessonCount;

private $save;

function \_\_construct($subjectsArray, $groupsArray, $teachersArray, $roomsArray)

{

parent::\_\_construct();

$this->subject = new Zend\_Form\_Element\_Select('subject');

$this->subject->setLabel('subject');

$this->subject->addMultiOptions($subjectsArray);

$this->addElement($this->subject);

$this->group = new Zend\_Form\_Element\_Select('group');

$this->group->setLabel('group');

$this->group->addMultiOptions($groupsArray);

$this->addElement($this->group);

$this->teacher = new Zend\_Form\_Element\_Select('teacher');

$this->teacher->setLabel('teacher');

$this->teacher->addMultiOptions($teachersArray);

$this->addElement($this->teacher);

$this->room = new Zend\_Form\_Element\_Select('room');

$this->room->setLabel('room');

$this->room->addMultiOptions($roomsArray);

$this->addElement($this->room);

$this->lessonLength = new Zend\_Form\_Element\_Text('lessonLength');

$this->lessonLength->setLabel('Lesson Length');

$this->lessonLength->autoInsertNotEmptyValidator(true);

$this->lessonLength->setRequired();

$this->addElement($this->lessonLength);

$this->lessonCount = new Zend\_Form\_Element\_Text('lessonCount');

$this->lessonCount->setLabel('Lesson Count');

$this->lessonCount->autoInsertNotEmptyValidator(true);

$this->lessonCount->setRequired();

$this->addElement($this->lessonCount);

$this->save = new Zend\_Form\_Element\_Submit('submit');

$this->save->setLabel('Save')

->setIgnore(true);

$this->addElement($this->save);

}

}

?>

###### Lessonedit.php

<?php

/\*\*

\* Create Edit Form for Admin->Manage->Lessons edit page

\* @author Aaron Redmond

\* @version 1.0

\* @created 11-Mar-2011

\*

\*/

class Form\_Admin\_manage\_lessonedit extends Zend\_Form

{

private $subject;

private $group;

private $teacher;

private $room;

private $lessonLength;

private $lessonCount;

private $save;

function \_\_construct($subjectsArray, $subjectId,

$groupsArray, $groupId,

$teachersArray, $teacherId,

$roomsArray, $roomId,

$lessonLength,

$lessonCount)

{

parent::\_\_construct();

$this->subject = new Zend\_Form\_Element\_Select('subject');

$this->subject->setLabel('subject');

$this->subject->addMultiOptions($subjectsArray);

$this->subject->setValue($subjectId);

$this->addElement($this->subject);

$this->group = new Zend\_Form\_Element\_Select('group');

$this->group->setLabel('group');

$this->group->addMultiOptions($groupsArray);

$this->group->setValue($groupId);

$this->addElement($this->group);

$this->teacher = new Zend\_Form\_Element\_Select('teacher');

$this->teacher->setLabel('teacher');

$this->teacher->addMultiOptions($teachersArray);

$this->teacher->setValue($teacherId);

$this->addElement($this->teacher);

$this->room = new Zend\_Form\_Element\_Select('room');

$this->room->setLabel('room');

$this->room->addMultiOptions($roomsArray);

$this->room->setValue($roomId);

$this->addElement($this->room);

$this->lessonLength = new Zend\_Form\_Element\_Text('lessonLength');

$this->lessonLength->setLabel('Lesson Length');

$this->lessonLength->setValue($lessonLength);

$this->addElement($this->lessonLength);

$this->lessonCount = new Zend\_Form\_Element\_Text('lessonCount');

$this->lessonCount->setLabel('Lesson Count');

$this->lessonCount->setValue($lessonCount);

$this->addElement($this->lessonCount);

$this->save = new Zend\_Form\_Element\_Submit('submit');

$this->save->setLabel('Save')

->setIgnore(true);

$this->addElement($this->save);

}

}

?>

###### Lessons.php

<?php

/\*\*

\* Create Add button for Admin->Manage->rooms page

\* @author Aaron Redmond

\* @version 1.0

\* @created 13-Mar-2011

\*

\*/

class Form\_admin\_manage\_lessons extends Zend\_Form

{

private $addButton;

function \_\_construct ()

{

parent::\_\_construct();

$this->addButton = new Zend\_Form\_Element\_Button('addButton');

$this->addButton->setLabel('Add Lesson')

->setAttrib('onclick', 'location.href="/admin/manage/lessonadd"');

$this->addElement($this->addButton);

}

}

?>

###### Roomadd.php

<?php

/\*\*

\* Create Add text elements for Admin->Manage->roomadd page

\* @author Aaron Redmond

\* @version 1.0

\* @created 24-Feb-2011

\* @version 1.1

\* @created 22-Mar-2011 - Added Capacity

\*

\*/

class Form\_Admin\_manage\_roomadd extends Zend\_Form

{

private $roomName;

private $roomShortName;

private $capacity;

private $submitButton;

private $cancelButton;

function \_\_construct ()

{

parent::\_\_construct();

$this->roomName = new Zend\_Form\_Element\_Text('roomName');

$this->roomName->setLabel('Room Name')

->setAutoInsertNotEmptyValidator(true)

->setRequired()

->setAttrib('size', '40');

$this->addElement($this->roomName);

$this->roomShortName = new Zend\_Form\_Element\_Text('roomShortName');

$this->roomShortName->setLabel('Room Short Name')

->setAutoInsertNotEmptyValidator(true)

->setRequired()

->setAttrib('size', '40');

$this->addElement($this->roomShortName);

$this->capacity = new Zend\_Form\_Element\_Text('capacity');

$this->capacity->setLabel('Capacity')

->setAutoInsertNotEmptyValidator(true)

->setRequired();

$this->addElement($this->capacity);

$this->submitButton = new Zend\_Form\_Element\_Submit('submit');

$this->submitButton->setLabel('Save')

->setIgnore(true);

$this->addElement($this->submitButton);

$this->cancelButton = new Zend\_Form\_Element\_Button('cancelButton');

$this->cancelButton->setLabel('Back');

$this->cancelButton->setAttrib('onclick', 'location.href="/admin/manage/rooms"');

$this->addElement($this->cancelButton);

}

}

?>

###### Rooms.php

<?php

/\*\*

\* Create Add button for Admin->Manage->rooms page

\* @author Aaron Redmond

\* @version 1.0

\* @created 24-Feb-2011

\*

\*/

class Form\_Admin\_manage\_rooms extends Zend\_Form

{

private $addButton;

function \_\_construct ()

{

parent::\_\_construct();

$this->addButton = new Zend\_Form\_Element\_Button('addButton');

$this->addButton->setLabel('Add')

->setAttrib('onclick', 'location.href="/admin/manage/roomadd"');

$this->addElement($this->addButton);

}

}

?>

###### School.php

<?php

/\*\*

\* Create Edit button for Admin->Manage->School page

\* @author Aaron Redmond

\* @version 1.0

\* @created 22-Feb-2011

\*

\*/

class Form\_Admin\_manage\_school extends Zend\_Form

{

private $editButton;

function \_\_construct()

{

parent::\_\_construct();

$this->editButton = new Zend\_Form\_Element\_Button('editButton');

$this->editButton->setLabel('edit');

$this->editButton->setAttrib('onclick', 'location.href="/admin/manage/editschool"');

$this->addElement($this->editButton);

}

}

?>

###### Studentadd.php

<?php

/\*\*

\* Create Form for Admin->Manage->studentadd page

\* @author Aaron Redmond

\* @version 1.0

\* @created 26-Feb-2011

\*

\*/

class Form\_Admin\_manage\_studentadd extends Zend\_Form

{

private $usernameEditBox;

private $passwordEditBox;

private $forenameEditBox;

private $surnameEditBox;

private $emailEditBox;

private $addressEditBox;

private $phoneEditBox;

private $mobileEditBox;

private $submitButton;

private $cancelButton;

function \_\_construct()

{

parent::\_\_construct();

$this->usernameEditBox = new Zend\_Form\_Element\_Text('usernameEditBox');

$this->usernameEditBox->setLabel('Username')

->setAutoInsertNotEmptyValidator(true)

->setRequired()

->addValidator(new Validate\_Username())

->setAttrib('size', '50');

$this->addElement($this->usernameEditBox);

$this->passwordEditBox = new Zend\_Form\_Element\_Password('passwordEditBox');

$this->passwordEditBox->setLabel('Password')

->setAutoInsertNotEmptyValidator(true)

->setRequired()

->addValidator(new Validate\_Password())

->setAttrib('size', '50');

$this->addElement($this->passwordEditBox);

$this->forenameEditBox = new Zend\_Form\_Element\_Text('forenameEditBox');

$this->forenameEditBox->setLabel('Forename')

->setAutoInsertNotEmptyValidator(true)

->setRequired()

->setAttrib('size', '50');

$this->addElement($this->forenameEditBox);

$this->surnameEditBox = new Zend\_Form\_Element\_Text('surnameEditBox');

$this->surnameEditBox->setLabel('Surname')

->setAutoInsertNotEmptyValidator(true)

->setRequired()

->setAttrib('size', '50');

$this->addElement($this->surnameEditBox);

$this->emailEditBox = new Zend\_Form\_Element\_Text('emailEditBox');

$this->emailEditBox->setLabel('E-mail')

->setAutoInsertNotEmptyValidator(true)

->setRequired()

->addValidator(new Validate\_Email())

->addValidator(new Validate\_EmailUnique())

->setAttrib('size', '50');

$this->addElement($this->emailEditBox);

$this->addressEditBox = new Zend\_Form\_Element\_Text('addressEditBox');

$this->addressEditBox->setLabel('Address')

->setAutoInsertNotEmptyValidator(true)

->setRequired()

->setAttrib('size', '50');

$this->addElement($this->addressEditBox);

$this->phoneEditBox = new Zend\_Form\_Element\_Text('phoneEditBox');

$this->phoneEditBox->setLabel('Telephone')

->setAutoInsertNotEmptyValidator(false)

->setRequired();

$this->addElement($this->phoneEditBox);

$this->mobileEditBox = new Zend\_Form\_Element\_Text('mobileEditBox');

$this->mobileEditBox->setLabel('Mobile')

->setAutoInsertNotEmptyValidator(false)

->setRequired();

$this->addElement($this->mobileEditBox);

$this->submitButton = new Zend\_Form\_Element\_Submit('submit');

$this->submitButton->setLabel('Save')

->setIgnore(true);

$this->addElement($this->submitButton);

$this->cancelButton = new Zend\_Form\_Element\_Button('cancelButton');

$this->cancelButton->setLabel('Back')

->setAttrib('onclick', 'location.href="/admin/manage/students"');

$this->addElement($this->cancelButton);

}

}

?>

###### Students.php

<?php

/\*\*

\* Create Form for Admin->Manage->students page

\* @author Aaron Redmond

\* @version 1.0

\* @created 26-Feb-2011

\*

\*/

class Form\_Admin\_manage\_students extends Zend\_Form

{

private $addButton;

function \_\_construct ()

{

parent::\_\_construct();

$this->addButton = new Zend\_Form\_Element\_Button('addButton');

$this->addButton->setLabel('Add')

->setAttrib('onclick', 'location.href="/admin/manage/studentadd"');

$this->addElement($this->addButton);

}

}

?>

###### Subjects.php

<?php

/\*\*

\* Create Fofm elements for Admin->Manage->subjectadd page

\* @author Aaron Redmond

\* @version 1.0

\* @created 25-Feb-2011

\*

\*/

class Form\_Admin\_manage\_subjectadd extends Zend\_Form

{

private $subject;

private $submitButton;

private $cancelButton;

function \_\_construct ()

{

parent::\_\_construct();

$this->subject = new Zend\_Form\_Element\_Text('subject');

$this->subject->setLabel('Subject')

->setAutoInsertNotEmptyValidator(true)

->setRequired()

->setAttrib('size', '40');

$this->addElement($this->subject);

$this->submitButton = new Zend\_Form\_Element\_Submit('submit');

$this->submitButton->setLabel('Save')

->setIgnore(true);

$this->addElement($this->submitButton);

$this->cancelButton = new Zend\_Form\_Element\_Button('cancelButton');

$this->cancelButton->setLabel('Back');

$this->cancelButton->setAttrib('onclick', 'location.href="/admin/manage/subjects"');

$this->addElement($this->cancelButton);

}

}

?>

###### Subjectadd.php

<?php

/\*\*

\* Create Fofm elements for Admin->Manage->subjectadd page

\* @author Aaron Redmond

\* @version 1.0

\* @created 25-Feb-2011

\*

\*/

class Form\_Admin\_manage\_subjectadd extends Zend\_Form

{

private $subject;

private $submitButton;

private $cancelButton;

function \_\_construct ()

{

parent::\_\_construct();

$this->subject = new Zend\_Form\_Element\_Text('subject');

$this->subject->setLabel('Subject')

->setAutoInsertNotEmptyValidator(true)

->setRequired()

->setAttrib('size', '40');

$this->addElement($this->subject);

$this->submitButton = new Zend\_Form\_Element\_Submit('submit');

$this->submitButton->setLabel('Save')

->setIgnore(true);

$this->addElement($this->submitButton);

$this->cancelButton = new Zend\_Form\_Element\_Button('cancelButton');

$this->cancelButton->setLabel('Back');

$this->cancelButton->setAttrib('onclick', 'location.href="/admin/manage/subjects"');

$this->addElement($this->cancelButton);

}

}

?>

###### Subjects.php

<?php

/\*\*

\* Create form for Admin->Manage->subjects page

\* @author Aaron Redmond

\* @version 1.0

\* @created 25-Feb-2011

\*

\*/

class Form\_Admin\_manage\_subjects extends Zend\_Form

{

private $addButton;

function \_\_construct ()

{

parent::\_\_construct();

$this->addButton = new Zend\_Form\_Element\_Button('addButton');

$this->addButton->setLabel('Add')

->setAttrib('onclick', 'location.href="/admin/manage/subjectadd"');

$this->addElement($this->addButton);

}

}

?>

###### Teacheradd.php

<?php

/\*\*

\* Create Form for Admin->Manage->teacheradd page

\* @author Aaron Redmond

\* @version 1.0

\* @created 25-Feb-2011

\*

\*/

class Form\_Admin\_manage\_teacheradd extends Zend\_Form

{

private $usernameEditBox;

private $passwordEditBox;

private $forenameEditBox;

private $surnameEditBox;

private $emailEditBox;

private $addressEditBox;

private $phoneEditBox;

private $mobileEditBox;

private $submitButton;

private $cancelButton;

function \_\_construct()

{

parent::\_\_construct();

$this->usernameEditBox = new Zend\_Form\_Element\_Text('usernameEditBox');

$this->usernameEditBox->setLabel('Username')

->setAutoInsertNotEmptyValidator(true)

->setRequired()

->addValidator(new Validate\_Username())

->setAttrib('size', '50');

$this->addElement($this->usernameEditBox);

$this->passwordEditBox = new Zend\_Form\_Element\_Password('passwordEditBox');

$this->passwordEditBox->setLabel('Password')

->setAutoInsertNotEmptyValidator(true)

->setRequired()

->addValidator(new Validate\_Password())

->setAttrib('size', '50');

$this->addElement($this->passwordEditBox);

$this->forenameEditBox = new Zend\_Form\_Element\_Text('forenameEditBox');

$this->forenameEditBox->setLabel('Forename')

->setAutoInsertNotEmptyValidator(true)

->setRequired()

->setAttrib('size', '50');

$this->addElement($this->forenameEditBox);

$this->surnameEditBox = new Zend\_Form\_Element\_Text('surnameEditBox');

$this->surnameEditBox->setLabel('Surname')

->setAutoInsertNotEmptyValidator(true)

->setRequired()

->setAttrib('size', '50');

$this->addElement($this->surnameEditBox);

$this->emailEditBox = new Zend\_Form\_Element\_Text('emailEditBox');

$this->emailEditBox->setLabel('E-mail')

->setAutoInsertNotEmptyValidator(true)

->setRequired()

->addValidator(new Validate\_Email())

->addValidator(new Validate\_EmailUnique())

->setAttrib('size', '50');

$this->addElement($this->emailEditBox);

$this->addressEditBox = new Zend\_Form\_Element\_Text('addressEditBox');

$this->addressEditBox->setLabel('Address')

->setAutoInsertNotEmptyValidator(true)

->setRequired()

->setAttrib('size', '50');

$this->addElement($this->addressEditBox);

$this->phoneEditBox = new Zend\_Form\_Element\_Text('phoneEditBox');

$this->phoneEditBox->setLabel('Telephone')

->setAutoInsertNotEmptyValidator(false)

->setRequired();

$this->addElement($this->phoneEditBox);

$this->mobileEditBox = new Zend\_Form\_Element\_Text('mobileEditBox');

$this->mobileEditBox->setLabel('Mobile')

->setAutoInsertNotEmptyValidator(false)

->setRequired();

$this->addElement($this->mobileEditBox);

$this->submitButton = new Zend\_Form\_Element\_Submit('submit');

$this->submitButton->setLabel('Save')

->setIgnore(true);

$this->addElement($this->submitButton);

$this->cancelButton = new Zend\_Form\_Element\_Button('cancelButton');

$this->cancelButton->setLabel('Back')

->setAttrib('onclick', 'location.href="/admin/manage/teachers"');

$this->addElement($this->cancelButton);

}

}

?>

###### Teacherconstraints.php

<?php

/\*\*

\* Create Add teacher constraints form

\*

\* @author Aaron Redmond

\* @version 1.0

\* @created 01-Apr-2011

\*

\*/

class Form\_Admin\_manage\_teacherconstraints extends Zend\_Form

{

private $days;

private $times;

private $saveButton;

function \_\_construct ($daysArray, $timesArray)

{

parent::\_\_construct();

$this->days = new Zend\_Form\_Element\_Select('days');

$this->days->setLabel('Day');

$this->days->addMultiOptions($daysArray);

$this->addElement($this->days);

$this->times = new Zend\_Form\_Element\_Select('times');

$this->times->setLabel('Time');

$this->times->addMultiOptions($timesArray);

$this->addElement($this->times);

$this->saveButton = new Zend\_Form\_Element\_Submit('saveButton');

$this->saveButton->setLabel('Add');

$this->addElement($this->saveButton);

}

}

?>

###### Teachers.php

<?php

/\*\*

\* Create Form for Admin->Manage->teachers page

\* @author Aaron Redmond

\* @version 1.0

\* @created 25-Feb-2011

\*

\*/

class Form\_Admin\_manage\_teachers extends Zend\_Form

{

private $addButton;

function \_\_construct ()

{

parent::\_\_construct();

$this->addButton = new Zend\_Form\_Element\_Button('addButton');

$this->addButton->setLabel('Add')

->setAttrib('onclick', 'location.href="/admin/manage/teacheradd"');

$this->addElement($this->addButton);

}

}

?>

##### Myaccount

###### Password.php

<?php

/\*\*

\*

\* Create Form for Admin->My Account->Change Password

\* @author Aaron Redmond

\* @version 1.0

\* @created 29-Mar-2011

\*/

class Form\_Admin\_myaccount\_password extends Zend\_Form

{

private $oldPassword;

private $newPassword;

private $verifyNewPassword;

private $submitButton;

function \_\_construct()

{

parent::\_\_construct();

$this->oldPassword = new Zend\_Form\_Element\_Password('oldPassword');

$this->oldPassword->setLabel('Old Password');

$this->oldPassword->autoInsertNotEmptyValidator(true);

$this->oldPassword->setRequired();

$this->addElement($this->oldPassword);

$this->newPassword = new Zend\_Form\_Element\_Password('newPassword');

$this->newPassword->setLabel('New Password');

$this->newPassword->autoInsertNotEmptyValidator(true);

$this->newPassword->setRequired();

$this->newPassword->addValidator(new Validate\_Password());

$this->addElement($this->newPassword);

$this->verifyNewPassword = new Zend\_Form\_Element\_Password('verifyNewPassword');

$this->verifyNewPassword->setLabel('Retype New Password');

$this->verifyNewPassword->autoInsertNotEmptyValidator(true);

$this->verifyNewPassword->setRequired();

$this->verifyNewPassword->addValidator(new Validate\_Password());

$this->addElement($this->verifyNewPassword);

$this->submitButton = new Zend\_Form\_Element\_Submit('submit');

$this->submitButton->setLabel('Reset')

->setIgnore(true);

$this->addElement($this->submitButton);

}

}

?>

###### Username.php

<?php

/\*\*

\*

\* Create Form for Admin->My Account->Change Username

\* @author Aaron Redmond

\* @version 1.0

\* @created 29-Mar-2011

\*/

class Form\_Admin\_myaccount\_username extends Zend\_Form

{

private $newUsername;

private $submitButton;

function \_\_construct()

{

parent::\_\_construct();

$this->newUsername = new Zend\_Form\_Element\_Text('newUsername');

$this->newUsername->setLabel('New Username');

$this->newUsername->autoInsertNotEmptyValidator(true);

$this->newUsername->setRequired();

$this->newUsername->addValidator(new Validate\_User());

$this->addElement($this->newUsername);

$this->submitButton = new Zend\_Form\_Element\_Submit('submit');

$this->submitButton->setLabel('Change')

->setIgnore(true);

$this->addElement($this->submitButton);

}

}

?>

##### Setup

###### Fileupload.php

<?php

/\*\*

\*

\* Create Form for Admin->setup->personalisation upload file

\* @author Aaron Redmond

\* @version 1.0

\* @created 31-Mar-2011

\*/

class Form\_Admin\_setup\_fileupload extends Zend\_Form

{

private $file;

private $submitButton;

function \_\_construct($path)

{

parent::\_\_construct();

$this->file = new Zend\_Form\_Element\_file('file');

$this->file->setLabel('File');

$this->file->setRequired(true);

$this->file->addValidator('NotEmpty');

$this->file->setDestination($path);

$this->file->addValidator(new Zend\_Validate\_File\_Upload());

$this->file->addValidator('MimeType', true, array('image/jpeg','image/gif','image/bmp'));

$this->addElement($this->file);

$this->submitButton = new Zend\_Form\_Element\_Submit('submit');

$this->submitButton->setLabel('Upload');

$this->submitButton->setIgnore(true);

$this->addElement($this->submitButton);

}

}

?>

###### Instrumentadd.php

<?php

/\*\*

\*

\* Create Form for Admin->setup->instruments->add page

\* @author Aaron Redmond

\* @version 1.0

\* @created 14-Mar-2011

\*/

class Form\_Admin\_setup\_instrumentadd extends Zend\_Form

{

private $instrument;

private $submitButton;

private $cancelButton;

function \_\_construct()

{

parent::\_\_construct();

$this->instrument = new Zend\_Form\_Element\_Text('instrument');

$this->instrument->setLabel('Instrument Name');

$this->instrument->autoInsertNotEmptyValidator(true);

$this->instrument->setRequired();

$this->addElement($this->instrument);

$this->submitButton = new Zend\_Form\_Element\_Submit('submit');

$this->submitButton->setLabel('Save')

->setIgnore(true);

$this->addElement($this->submitButton);

$this->cancelButton = new Zend\_Form\_Element\_Button('cancelButton');

$this->cancelButton->setLabel('Cancel')

->setAttrib('onclick', 'location.href="/admin/setup/instruments"');

$this->addElement($this->cancelButton);

}

}

?>

###### Instrumentaddbutton.php

<?php

/\*\*

\*

\* Create Add button for Admin->setup->instruments page

\* @author Aaron Redmond

\* @version 1.0

\* @created 14-Mar-2011

\*/

class Form\_Admin\_setup\_instrumentaddbutton extends Zend\_Form

{

private $addButton;

function \_\_construct()

{

parent::\_\_construct();

$this->addButton = new Zend\_Form\_Element\_Button('addButton');

$this->addButton->setLabel('Add Instrument')

->setAttrib('onclick', 'location.href="/admin/setup/instrumentadd"');

$this->addElement($this->addButton);

}

}

?>

###### Instrumentedit.php

<?php

/\*\*

\*

\* Create Form for Admin->setup->instruments edit page

\* @author Aaron Redmond

\* @version 1.0

\* @created 14-Mar-2011

\*/

class Form\_Admin\_setup\_instrumentedit extends Zend\_Form

{

private $instrument;

private $submitButton;

private $cancelButton;

function \_\_construct($instrument)

{

parent::\_\_construct();

$this->instrument = new Zend\_Form\_Element\_Text('instrument');

$this->instrument->setLabel('Instrument Name');

$this->instrument->autoInsertNotEmptyValidator(true);

$this->instrument->setRequired();

$this->instrument->setValue($instrument);

$this->addElement($this->instrument);

$this->submitButton = new Zend\_Form\_Element\_Submit('submit');

$this->submitButton->setLabel('Save')

->setIgnore(true);

$this->addElement($this->submitButton);

$this->cancelButton = new Zend\_Form\_Element\_Button('cancelButton');

$this->cancelButton->setLabel('Cancel')

->setAttrib('onclick', 'location.href="/admin/setup/instruments"');

$this->addElement($this->cancelButton);

}

}

?>

###### News.php

<?php

/\*\*

\*

\* Create Form for Admin->setup->News page

\* @author Aaron Redmond

\* @version 1.0

\* @created 29-Mar-2011

\*/

class Form\_Admin\_setup\_news extends Zend\_Form

{

private $articleTitle;

private $article;

private $publishButton;

function \_\_construct ()

{

parent::\_\_construct();

$this->articleTitle = new Zend\_Form\_Element\_Text('articleTitle');

$this->articleTitle->setLabel('Title');

$this->articleTitle->autoInsertNotEmptyValidator(true);

$this->articleTitle->setRequired();

$this->articleTitle->setAttrib('size', '50');

$this->addElement($this->articleTitle);

$this->article = new Zend\_Form\_Element\_Textarea('article');

$this->article->setLabel('Article');

$this->article->autoInsertNotEmptyValidator(true);

$this->article->setRequired();

$this->addElement($this->article);

$this->publishButton = new Zend\_Form\_Element\_Submit('submit');

$this->publishButton->setLabel('Publish');

$this->publishButton->setIgnore(true);

$this->addElement($this->publishButton);

}

}

?>

##### Timetabling

###### Generate.php

<?php

/\*\*

\*

\* Create Generate Button for Admin->Timetabling->Timetable page

\* @author Aaron Redmond

\* @version 1.0

\* @created 17-Mar-2011

\*/

class Form\_Admin\_timetabling\_generate extends Zend\_Form

{

private $submitButton;

function \_\_construct()

{

parent::\_\_construct();

$this->submitButton = new Zend\_Form\_Element\_Submit('submit');

$this->submitButton->setLabel('Generate')

->setIgnore(true);

$this->addElement($this->submitButton);

}

}

?>

###### Information.php

<?php

/\*\*

\*

\* Create Form for Admin->Timetabling->information page

\* @author Aaron Redmond

\* @version 1.0

\* @created 17-Mar-2011

\*/

class Form\_Admin\_timetabling\_information extends Zend\_Form

{

private $academicYear;

private $daysPerWeek;

private $lessonsPerDay;

private $startTime;

private $lessonLength;

private $submitButton;

private $cancelButton;

function \_\_construct($academicYear, $daysPerWeekArray, $daysPerWeek, $lessonsPerDayArray, $lessonsPerDay, $startTime, $lessonLength)

{

parent::\_\_construct();

$this->academicYear = new Zend\_Form\_Element\_Text('academicYear');

$this->academicYear->setLabel('Academic Year');

$this->academicYear->autoInsertNotEmptyValidator(true);

$this->academicYear->setRequired();

$this->academicYear->setValue($academicYear);

$this->addElement($this->academicYear);

$this->daysPerWeek = new Zend\_Form\_Element\_Select('daysPerWeek');

$this->daysPerWeek->setLabel('Working days per week');

$this->daysPerWeek->addMultiOptions($daysPerWeekArray);

$this->daysPerWeek->setValue($daysPerWeek);

$this->addElement($this->daysPerWeek);

$this->lessonsPerDay = new Zend\_Form\_Element\_Select('lessonsPerDay');

$this->lessonsPerDay->setLabel('Lessons per day');

$this->lessonsPerDay->addMultiOptions($lessonsPerDayArray);

$this->lessonsPerDay->setValue($lessonsPerDay);

$this->addElement($this->lessonsPerDay);

$this->startTime = new Zend\_Form\_Element\_Text('startTime');

$this->startTime->setLabel('Start Time (First Lesson)');

$this->startTime->setValue($startTime);

$this->startTime->autoInsertNotEmptyValidator(true);

$this->startTime->setRequired();

$this->startTime->addValidator(new Validate\_Time());

$this->addElement($this->startTime);

$this->lessonLength = new Zend\_Form\_Element\_Text('lessonLength');

$this->lessonLength->setLabel('Lesson Length (in minutes)');

$this->lessonLength->setValue($lessonLength);

$this->lessonLength->autoInsertNotEmptyValidator(true);

$this->lessonLength->setRequired();

$this->lessonLength->addValidator(new Validate\_Time());

$this->addElement($this->lessonLength);

$this->submitButton = new Zend\_Form\_Element\_Submit('submit');

$this->submitButton->setLabel('Save')

->setIgnore(true);

$this->addElement($this->submitButton);

$this->cancelButton = new Zend\_Form\_Element\_Button('cancelButton');

$this->cancelButton->setLabel('Back')

->setAttrib('onclick', 'location.href="/admin/index/timetabling"');

$this->addElement($this->cancelButton);

}

}

?>

###### Printbutton.php

<?php

/\*\*

\*

\* Create Print button for Printing timetables

\* @author Aaron Redmond

\* @version 1.0

\* @created 06-Apr-2011

\*/

class Form\_Admin\_timetabling\_printbutton extends Zend\_Form

{

private $printButton;

function \_\_construct($data)

{

parent::\_\_construct();

$this->printButton = new Zend\_Form\_Element\_Button('print');

$this->printButton->setLabel('Print');

$this->printButton->setIgnore(true);

$this->printButton->setAttribs(array('onclick' => 'printTT(' .$data .');'));

$this->addElement($this->printButton);

}

}

?>

###### Subtype.php

<?php

/\*\*

\*

\* Create Select Element for Timetabling->edit/add pages

\* @author Aaron Redmond

\* @version 1.0

\* @created 26-Mar-2011

\*/

class Form\_Admin\_timetabling\_subtype extends Zend\_Form

{

private $timetableSubType;

function \_\_construct ($timetableSubTypes, $type)

{

parent::\_\_construct();

$this->timetableSubType = new Zend\_Form\_Element\_Select('timetableSubType');

$this->timetableSubType->setLabel('Subtype');

$this->timetableSubType->addMultiOptions($timetableSubTypes);

$this->timetableSubType->setOptions(array('onChange' => 'submit()','id' => $type));

$this->addElement($this->timetableSubType);

}

}

?>

###### Timetables.php

<?php

/\*\*

\*

\* Create timetable Form for Admin->Timetabling->Timetable page

\* @author Aaron Redmond

\* @version 1.0

\* @created 26-Mar-2011

\*/

class Form\_Admin\_timetabling\_timetables extends Zend\_Form

{

private $timetableType;

function \_\_construct ($timetableTypes, $displayType)

{

parent::\_\_construct();

$this->timetableType = new Zend\_Form\_Element\_Select('timetableType');

$this->timetableType->setLabel('Time table type');

$this->timetableType->addMultiOptions($timetableTypes);

$this->timetableType->setValue($displayType);

$this->timetableType->setOptions(array('onChange' => 'submit()'));

$this->addElement($this->timetableType);

}

}

?>

###### newItemEdit.php

<?php

/\*\*

\*

\* Create Form for Admin->setup->News edit page

\* @author Aaron Redmond

\* @version 1.0

\* @created 29-Mar-2011

\*/

class Form\_Admin\_setup\_newsItemEdit extends Zend\_Form

{

private $articleTitle;

private $article;

private $publishButton;

function \_\_construct ($title, $articleText)

{

parent::\_\_construct();

$this->articleTitle = new Zend\_Form\_Element\_Text('articleTitle');

$this->articleTitle->setLabel('Title');

$this->articleTitle->autoInsertNotEmptyValidator(true);

$this->articleTitle->setRequired();

$this->articleTitle->setValue($title);

$this->addElement($this->articleTitle);

$this->article = new Zend\_Form\_Element\_Textarea('article');

$this->article->setLabel('Article');

$this->article->autoInsertNotEmptyValidator(true);

$this->article->setRequired();

$this->article->setValue($articleText);

$this->addElement($this->article);

$this->publishButton = new Zend\_Form\_Element\_Submit('submit');

$this->publishButton->setLabel('Save');

$this->publishButton->setIgnore(true);

$this->addElement($this->publishButton);

}

}

?>

### Layouts

The following files are layouts which are used to wrap the HTML content .

#### Layout.phtml

Empty File.

#### Login\_layout.phtml

<!DOCTYPE html>

<html>

<head>

<?php **echo** $this->headTitle();?>

<meta http-equiv=*"Content-Language"* content=*"en-us"*>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=utf-8"*>

<link href=*"/styles/msmstyles.css"* rel=*"stylesheet"* type=*"text/css"*>

</head>

<body>

<div id=*"wrap"*>

<div id=*"header"*>

<div>

<img alt=*"Music School Manager"* src="*/msm\_images/*<?php **echo** $this->layout()->image ?>">

</div>

</div>

<div id=*"insidewrap"*>

<div id=*"login"*>

<b>Login</b>

<?php **echo** $this->loginForm?>

<?php **echo** $this->errorMsg?>

<a href=*"/request"*>Forgotten your User Name/Password?</a>

</div>

<p><b><?php **echo** $this->newsTitle?></b></p>

<br>

<div>

<?php **echo** $this->news?>

</div>

</div>

<div id=*"footer"*>

Music School Manager v1.0

</div>

</div>

</body>

</html>

#### Request\_layout.phtml

<!DOCTYPE html>

<html>

<head>

<?php **echo** $this->headTitle();?>

<meta http-equiv=*"Content-Language"* content=*"en-us"*>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=utf-8"*>

<link href=*"/styles/msmstyles.css"* rel=*"stylesheet"* type=*"text/css"*>

</head>

<body>

<div id=*"wrap"*>

<div id=*"header"*>

<div>

<img alt=*"Music School Manager"* src="*/msm\_images/*<?php **echo** $this->layout()->image ?>">

</div>

</div>

<div id=*"insidewrap"*>

<p>Enter your email address and the system will send you your new login details.</p>

<?php **echo** $this->requestForm?>

<?php **echo** $this->errorMessage?>

</div>

<div id=*"footer"*>

Footer Stuff

</div>

</div>

</body>

</html>

#### Admin

##### Admin\_layout.phtml

<!DOCTYPE html>

<html>

<head>

<?php **echo** $this->headTitle();?>

<meta http-equiv=*"Content-Language"* content=*"en-us"*>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=utf-8"*>

<link href=*"/styles/msmstyles.css"* rel=*"stylesheet"* type=*"text/css"*>

<link href=*"/styles/menu.css"* rel=*"stylesheet"* type=*"text/css"*>

<?php **echo** $this->render('Admin/menu\_script.phtml');?>

</head>

<body>

<div id=*"wrap"*>

<div id=*"header"*>

<div>

<img alt=*"Music School Manager"* src="*/msm\_images/*<?php **echo** $this->layout()->image ?>">

</div>

<div>

<?php **echo** $this->render('Admin/admin\_menu.phtml');?>

</div>

</div>

<div id=*"insidewrap"*>

<?php **echo** $this->layout()->content?>

</div>

<div id=*"footer"*>

Music School Manager v1.0

</div>

</div>

</body>

</html>

##### Admin\_menu.phtml

<div>

<ul class=*"topnav"*>

<li>

<a href=*"/admin/index/index"*>Home</a>

</li>

<li>

<a href=*"/admin/index/manage"*>Manage</a>

<ul class=*"subnav"*>

<li><a href=*"/admin/manage/school"*>School</a></li>

<li><a href=*"/admin/manage/rooms"*>Rooms</a></li>

<li><a href=*"/admin/manage/subjects"*>Subjects</a></li>

<li><a href=*"/admin/manage/teachers"*>Teachers</a></li>

<li><a href=*"/admin/manage/students"*>Students</a></li>

<li><a href=*"/admin/manage/groups"*>Groups</a></li>

<li><a href=*"/admin/manage/lessons"*>Lessons</a></li>

</ul>

</li>

<li>

<a href=*"/admin/index/timetabling"*>Timetabling</a>

<ul class=*"subnav"*>

<li>

<a href=*"/admin/timetabling/information"*>Timetable Information</a>

<a href=*"/admin/timetabling/timetables/"*>Timetabling</a>

</li>

</ul>

</li>

<li>

<a href=*"/admin/index/userpanel"*>User Panel</a>

<ul class=*"subnav"*>

<li>

<a href=*"/admin/index/systemsetup"*>System Setup</a>

<ul class=*"subnav2"*>

<li>

<a href=*"/admin/setup/instruments"*>Instruments</a>

<a href=*"/admin/setup/personalisation"*>Personalisation</a>

<a href=*"/admin/setup/news"*>News</a>

</li>

</ul>

</li>

<li>

<a href=*"/admin/index/myaccount"*>My Account</a>

<ul class=*"subnav2"*>

<li>

<a href=*"/admin/myaccount/username"*>Change Username</a>

<a href=*"/admin/myaccount/password"*>Change Password</a>

</li>

</ul>

</li>

</ul>

</li>

<li>

<a href=*"/admin/index/logout"*>Log Out</a>

</li>

<li>

<a href=*"/admin/index/help"*>Help</a>

</li>

</ul>

</div>

##### Menu\_script.phtml

<script src=*"http://ajax.googleapis.com/ajax/libs/jquery/1.4.4/jquery.min.js"* type=*"text/javascript"*></script>

<script type=*"text/javascript"*>

$(document).ready(**function**(){

$("ul.topnav li a").mouseover(**function**() { //When trigger is clicked...

//Following events are applied to the subnav itself (moving subnav up and down)

$(**this**).parent().find("ul.subnav").slideDown('fast').show(); //Drop down the subnav on click

$(**this**).parent().hover(**function**() {

}, **function**(){

$(**this**).parent().find("ul.subnav").slideUp('slow'); //When the mouse hovers out of the subnav, move it back up

});

//Following events are applied to the trigger (Hover events for the trigger)

}).hover(**function**() {

$(**this**).addClass("subhover"); //On hover over, add class "subhover"

}, **function**(){ //On Hover Out

$(**this**).removeClass("subhover"); //On hover out, remove class "subhover"

});

$("ul.subnav li a").mouseover(**function**() { //When trigger is clicked...

//Following events are applied to the subnav itself (moving subnav up and down)

$(**this**).parent().find("ul.subnav2").slideDown('fast').show(); //Drop down the subnav on click

$(**this**).parent().hover(**function**() {

}, **function**(){

$(**this**).parent().find("ul.subnav2").slideUp('slow'); //When the mouse hovers out of the subnav, move it back up

});

//Following events are applied to the trigger (Hover events for the trigger)

}).hover(**function**() {

$(**this**).addClass("subhover"); //On hover over, add class "subhover"

}, **function**(){ //On Hover Out

$(**this**).removeClass("subhover"); //On hover out, remove class "subhover"

});

});

</script>

##### Personalisation\_layout.phtml

<!DOCTYPE html>

<html>

<head>

<?php **echo** $this->headTitle();?>

<meta http-equiv=*"Content-Language"* content=*"en-us"*>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=utf-8"*>

<link href=*"/styles/msmstyles.css"* rel=*"stylesheet"* type=*"text/css"*>

<link href=*"/styles/menu.css"* rel=*"stylesheet"* type=*"text/css"*>

<?php **echo** $this->render('Admin/menu\_script.phtml');?>

<link href=*"/styles/screen.css"* rel=*"stylesheet"* type=*"text/css"* media=*"screen"* />

<script src=*"http://ajax.googleapis.com/ajax/libs/jquery/1.4.4/jquery.min.js"* type=*"text/javascript"*></script>

<script type=*"text/javascript"* src=*"/js/easySlider1.7.js"*></script>

<script type=*"text/javascript"*>

$(document).ready(**function**(){

$("#slider").easySlider({

auto: **true**,

continuous: **true**

});

});

</script>

</head>

<body>

<div id=*"wrap"*>

<div id=*"header"*>

<div>

<img alt=*"Music School Manager"* src="*/msm\_images/*<?php **echo** $this->layout()->image ?>">

</div>

<div>

<?php **echo** $this->render('Admin/admin\_menu.phtml');?>

</div>

</div>

<div id=*"insidewrap"* align=*"center"*>

<?php **echo** $this->layout()->content?>

</div>

<div id=*"footer"*>

Music School Manager v1.0

</div>

</div>

</body>

</html>

#### Student

##### menu\_script.phtml

Same as 2.3.4.3

##### student\_layout.phtml

<!DOCTYPE html>

<html>

<head>

<?php **echo** $this->headTitle();?>

<meta http-equiv=*"Content-Language"* content=*"en-us"*>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=utf-8"*>

<link href=*"/styles/msmstyles.css"* rel=*"stylesheet"* type=*"text/css"*>

<link href=*"/styles/menu.css"* rel=*"stylesheet"* type=*"text/css"*>

<?php **echo** $this->render('student/menu\_script.phtml');?>

</head>

<body>

<div id=*"wrap"*>

<div id=*"header"*>

<div>

<img alt=*"Music School Manager"* src="*/msm\_images/*<?php **echo** $this->layout()->image ?>">

</div>

<div>

<?php **echo** $this->render('student/student\_menu.phtml');?>

</div>

</div>

<div id=*"insidewrap"*>

<?php **echo** $this->layout()->content?>

</div>

<div id=*"footer"*>

Music School Manager v1.0

</div>

</div>

</body>

</html>

##### student\_menu.phtml

<div>

<ul class=*"topnav"*>

<li>

<a href=*"/student/index/index"*>Home</a>

</li>

<li>

<a href=*"/student/timetable/timetable"*>Timetable</a>

</li>

<li>

<a href=*"#"*>User Panel</a>

<ul class=*"subnav"*>

<li>

<a href=*"/student/myaccount"*>My Account</a>

<ul class=*"subnav2"*>

<li>

<a href=*"/student/myaccount/username"*>Change Username</a>

<a href=*"/student/myaccount/password"*>Change Password</a>

</li>

</ul>

</li>

</ul>

</li>

<li>

<a href=*"/student/index/logout"*>Log Out</a>

</li>

<li>

<a href=*"/student/index/help "*>Help</a>

</li>

</ul>

</div>

#### Teacher

##### menu\_script.phtml

Same as 2.3.4.3

##### teacher\_layout.phtml

<!DOCTYPE html>

<html>

<head>

<?php **echo** $this->headTitle();?>

<meta http-equiv=*"Content-Language"* content=*"en-us"*>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=utf-8"*>

<link href=*"/styles/msmstyles.css"* rel=*"stylesheet"* type=*"text/css"*>

<link href=*"/styles/menu.css"* rel=*"stylesheet"* type=*"text/css"*>

<?php **echo** $this->render('Teacher/menu\_script.phtml');?>

</head>

<body>

<div id=*"wrap"*>

<div id=*"header"*>

<div>

<img alt=*"Music School Manager"* src="*/msm\_images/*<?php **echo** $this->layout()->image ?>">

</div>

<div>

<?php **echo** $this->render('Teacher/teacher\_menu.phtml');?>

</div>

</div>

<div id=*"insidewrap"*>

<?php **echo** $this->layout()->content?>

</div>

<div id=*"footer"*>

Music School Manager v1.0

</div>

</div>

</body>

</html>

##### teacher\_menu.phtml

<div>

<ul class=*"topnav"*>

<li>

<a href=*"/teacher/index/index"*>Home</a>

</li>

<li>

<a href=*"/teacher/timetable/timetable"*>Timetable</a>

</li>

<li>

<a href=*"#"*>User Panel</a>

<ul class=*"subnav"*>

<li>

<a href=*"/teacher/myaccount"*>My Account</a>

<ul class=*"subnav2"*>

<li>

<a href=*"/teacher/myaccount/username"*>Change Username</a>

<a href=*"/teacher/myaccount/password"*>Change Password</a>

</li>

</ul>

</li>

</ul>

</li>

<li>

<a href=*"/teacher/index/logout"*>Log Out</a>

</li>

<li>

<a href=*"/teacher/index/help"*>Help</a>

</li>

</ul>

</div>

### Models

The following classes represents the data.

#### DAO

##### Abstract.php

<?php

/\*\*

\*

\* @author Aaron Redmond

\* @version 1.0

\*/

abstract class DAO\_Abstract

{

/\*\*

\* Doctrine Entity Manager

\* @var Doctrine\ORM\EntityManager

\*/

protected $entityManager;

function \_\_construct()

{

$this->entityManager = Zend\_Registry::get('em');

}

}

?>

##### Groupconstraints.php

<?php

/\*\*

\* @author Aaron Redmond

\* @version 1.0

\* @created 02-Apr-2011

\*/

use models\entities\groupconstraints;

class DAO\_groupconstraints extends DAO\_Abstract

{

/\*\*

\* Retrieve all group constraints

\*/

public function retrieveConstraints ()

{

$dquery = 'SELECT u from models\entities\groupconstraints u';

return $this->entityManager->createQuery($dquery)->getResult();

}

/\*\*

\*Retrieve Single Constraint based upon Id

\*@param int $Id

\*/

public function retrieveConstraintsById ($Id)

{

try {

$dquery = $this->entityManager->createQueryBuilder()

->select('u')

->from('models\entities\groupconstraints', 'u')

->where('u.group\_constraint\_id=?1')

->setParameter(1, $Id);

return $dquery->getQuery()->getSingleResult();

} catch (Exception $e) {

return null;

}

}

/\*\*

\*Retrieve Constraints for specific group

\*@param int $groupId

\*/

public function retrieveConstraintsByGroup ($groupId)

{

try {

$dquery = $this->entityManager->createQueryBuilder()

->select('u')

->from('models\entities\groupconstraints', 'u')

->where('u.group=?1')

->setParameter(1, $groupId);

return $dquery->getQuery()->execute();

} catch (Exception $e) {

return null;

}

}

/\*\*

\* Add a new group Constriant

\* @param int $groupId

\* @param string $day

\* @param string $time

\*/

public function addConstraint($groupId, $day, $time)

{

$newConstraint = new groupconstraints();

$newConstraint->setGroup($groupId);

$newConstraint->setDay($day);

$newConstraint->setTime($time);

$this->entityManager->persist($newConstraint);

$this->entityManager->flush();

}

/\*\*

\*

\* Delete a constraint

\* @param models\entities\groupconstraints $groupconstraintsentity

\*/

public function deleteConstraint (groupconstraints $groupconstraintsentity)

{

$this->entityManager->remove($groupconstraintsentity);

$this->entityManager->flush();

}

}

?>

##### Groups.php

<?php

/\*\*

\* @author Aaron Redmond

\* @version 1.0

\* @created 26-Feb-2011

\*/

use models\entities\groups;

use Doctrine\ORM\Query\Expr\Func;

class DAO\_groups extends DAO\_Abstract

{

/\*\*

\* @return DAO object containing all groups

\*/

public function retrieveGroups ()

{

$dquery = 'SELECT u from models\entities\groups u';

return $this->entityManager->createQuery($dquery)->getResult();

}

/\*\*

\* Retrieve group based on id

\* @param $id

\* @return DAO object containing reuqested group

\*/

public function retrieveGroupByID ($id)

{

try {

$dquery = $this->entityManager->createQueryBuilder()

->select('u')

->from('models\entities\groups', 'u')

->where('u.groupid=?1')

->setParameter(1, $id);

return $dquery->getQuery()->getSingleResult();

} catch (Exception $e) {

return null;

}

}

/\*\*

\* Update Group Information

\* @param models\entities\groups $groupEntity

\* @param string $groupName,

\* @param string $groupShortName

\* @param models\entities\grouptype $groupType

\*/

public function updateGroup ($groupEntity, $groupName, $groupShortName, $groupType)

{

$groupEntity->setGroupName($groupName);

$groupEntity->setGroupShortName($groupShortName);

$groupEntity->setGroupType($groupType);

$this->entityManager->flush();

}

/\*\*

\* Update the users of a group

\* @param models\entities\groups $groupEntity

\* @param ArrayCollection

\*/

public function updateUsersOfGroup ($groupEntity, $users)

{

try {

$groupEntity->setUsers($users);

$this->entityManager->flush();

} catch (Exception $e) {

return null;

}

}

/\*\*

\*

\* Remove user of a group

\* @param models\entities\groups $groupEntity

\* @param ArrayCollection $users

\*/

public function removeUsersOfGroup ($groupEntity, $users)

{

try {

$groupEntity->removeUsers($users);

$this->entityManager->flush();

} catch (Exception $e) {

return null;

}

}

/\*\*

\*

\* Add a new Group

\* @param String $groupname

\* @param String $groupshortname

\* @param models\entities\grouptype $grouptype

\*/

public function AddGroup ($groupname, $groupshortname, $grouptype)

{

$newGroup = new groups();

$newGroup->setGroupName($groupname);

$newGroup->setGroupShortName($groupshortname);

if ($grouptype != null){

$newGroup->setGroupType($grouptype);

}

$this->entityManager->persist($newGroup);

$this->entityManager->flush();

}

/\*\*

\*

\* Delete a group

\* @param models\entities\groups $groupEntity

\*/

public function deleteGroup (groups $groupEntity)

{

$this->entityManager->remove($groupEntity);

$this->entityManager->flush();

}

}

?>

##### Grouptype.php

<?php

/\*\*

\* @author Aaron Redmond

\* @version 1.0

\* @created 26-Feb-2011

\*/

use models\entities\grouptype;

use Doctrine\ORM\Query\Expr\Func;

class DAO\_grouptype extends DAO\_Abstract

{

/\*\*

\*

\* Retrieve all grouptypes

\* @return models\entities\grouptype || Null

\*/

public function retrieveGroupTypes ()

{

try {

$dquery = 'SELECT u from models\entities\grouptype u';

return $this->entityManager->createQuery($dquery)->getResult();

} catch (Exception $e) {

return null;

}

}

/\*\*

\*

\* Retrieve a group by Id

\* @param int $id

\* @return models\entities\grouptype || Null

\*/

public function retrieveGroupTypeByID($id)

{

try {

$dquery = $this->entityManager->createQueryBuilder()

->select('u')

->from('models\entities\grouptype', 'u')

->where('u.grouptypeid=?1')

->setParameter(1, $id);

return $dquery->getQuery()->getSingleResult();

} catch (Exception $e) {

return null;

}

}

/\*\*

\*

\* Add GroupType

\* @param string $groupType

\*/

public function addGroupType ($groupType)

{

$newGroupType = new grouptype();

$newGroupType->setGroupTypeName($groupType);

$this->entityManager->persist($newGroupType);

$this->entityManager->flush();

}

/\*\*

\*

\* Check to see if a group type exists

\* @param string $grouptypename

\* @return models\entities\grouptype | null

\*/

public function checkTypeExists ($grouptypename)

{

try {

$dquery = $this->entityManager->createQueryBuilder()

->select('u')

->from('models\entities\grouptype', 'u')

->where('LOWER(u.grouptypename)=?1')

->setParameter(1, $grouptypename);

return $dquery->getQuery()->getSingleResult();

} catch (Exception $e) {

return null;

}

}

/\*\*

\*

\* Delete Group Type

\* @param models\entities\grouptype $grouptypeentity

\*/

public function deleteGroupType (grouptype $grouptypeentity)

{

$this->entityManager->remove($grouptypeentity);

$this->entityManager->flush();

}

/\*\*

\*

\* Update Group Type

\* @param models\entities\grouptype $grouptypeentity

\* @param $value

\*/

public function updateGroupType (grouptype $grouptypeentity, $value)

{

$grouptypeentity->setGroupTypeName($value);

$this->entityManager->flush();

}

}

?>

##### Instruments.php

<?php

/\*\*

\* @author Aaron Redmond

\* @version 1.0

\* @created 04-Mar-2011

\*/

use models\entities\instruments;

use Doctrine\ORM\Query\Expr\Func;

class DAO\_instruments extends DAO\_Abstract

{

/\*\*

\*

\* Retrieve all instruments

\* @return models\entities\instruments

\*/

public function retrieveInstruments ()

{

$dquery = 'SELECT u from models\entities\instruments u';

return $this->entityManager->createQuery($dquery)->getResult();

}

/\*\*

\*

\* Retrieve an instrument by Id

\* @param $id

\* @return models\entities\instruments

\*/

public function retrieveInstrumentByID ($id)

{

$dquery = $this->entityManager->createQueryBuilder()

->select('u')

->from('models\entities\instruments', 'u')

->where('u.instrumentid=?1')

->setParameter(1, $id);

return $dquery->getQuery()->getSingleResult();

}

/\*\*

\*

\* Update instrument used field

\* @param models\entities\instruments $instrumentEntity

\* @param int $used

\*/

public function updateInstrumentUsed (instruments $instrumentEntity, $used)

{

$instrumentEntity->setInstrumentused($used);

$this->entityManager->flush();

}

/\*\*

\*

\* Delete an Instrument

\* @param models\entities\instruments $instrumententity

\*/

public function deleteInstrument (instruments $instrumententity)

{

$this->entityManager->remove($instrumententity);

$this->entityManager->flush();

}

/\*\*

\*

\* Update instrument name

\* @param models\entities\instruments $instrumentEntity

\* @param string $name

\*/

public function updateInstrumentName (instruments $instrumentEntity, $name)

{

$instrumentEntity->setInstrumentname($name);

$this->entityManager->flush();

}

/\*\*

\*

\* Check to see if an instrument exists

\* @param string $instrumentName

\* @return models\entities\instruments | null

\*/

public function checkInstrumentExists ($instrumentName)

{

try {

$dquery = $this->entityManager->createQueryBuilder()

->select('u')

->from('models\entities\instruments', 'u')

->where('LOWER(u.instrumentname)=?1')

->setParameter(1, $instrumentName);

return $dquery->getQuery()->getSingleResult();

} catch (Exception $e) {

return null;

}

}

/\*\*

\*

\* Add an Instrument

\* @param string $instrumentName

\*/

public function addInstrument ($instrumentName)

{

$newInstrument = new instruments();

$newInstrument->setInstrumentname($instrumentName);

$newInstrument->setInstrumentused(0);

$this->entityManager->persist($newInstrument);

$this->entityManager->flush();

}

}

?>

##### Lessons.php

<?php

/\*\*

\* @author Aaron Redmond

\* @version 1.0

\* @created 07-Mar-2011

\*/

use models\entities\rooms;

use models\entities\users;

use models\entities\groups;

use models\entities\subjects;

use models\entities\lessons;

use Doctrine\ORM\Query\Expr\Func;

class DAO\_lessons extends DAO\_Abstract

{

/\*\*

\*

\* Retrieve all Lessons

\* @return models\entities\lessons

\*/

public function retrieveLessons ()

{

$dquery = 'SELECT u from models\entities\lessons u';

return $this->entityManager->createQuery($dquery)->getResult();

}

/\*\*

\*

\* Delete a Lesson

\* @param models\entities\lessons $lessonentity

\*/

public function deleteLesson (lessons $lessonentity)

{

$this->entityManager->remove($lessonentity);

$this->entityManager->flush();

}

/\*\*

\*

\* Retrieve lesson by ID

\* @param string $id

\* @return models\entities\lessons

\*/

public function retrieveLessonByID ($id)

{

try {

$dquery = $this->entityManager->createQueryBuilder()

->select('u')

->from('models\entities\lessons', 'u')

->where('u.lessonid=?1')

->setParameter(1, $id);

return $dquery->getQuery()->getSingleResult();

} catch (Exception $e) {

return null;

}

}

/\*\*

\*

\* Update subject for edited lesson

\* @param models\entities\lessons $lesson

\* @param models\entities\subjects $subject

\*/

public function updateLessonSubject (lessons $lesson, subjects $subject)

{

$lesson->setSubjects($subject);

$this->entityManager->flush();

}

/\*\*

\*

\* Update group for edited lesson

\* @param models\entities\lessons $lesson

\* @param models\entities\subjects $subject

\*/

public function updateLessonGroup (lessons $lesson, groups $group)

{

$lesson->setGroup($group);

$this->entityManager->flush();

}

/\*\*

\*

\* Update teacher for edited lesson

\* @param models\entities\lessons $lesson

\* @param models\entities\users $teacher

\*/

public function updateLessonTeacher (lessons $lesson, users $teacher)

{

$lesson->setTeacher($teacher);

$this->entityManager->flush();

}

/\*\*

\*

\* Update room for edited lesson

\* @param models\entities\lessons $lesson

\* @param models\entities\rooms $room

\*/

public function updateLessonRoom (lessons $lesson, rooms $room)

{

$lesson->setRoom($room);

$this->entityManager->flush();

}

/\*\*

\*

\* Update lessson length and count for edited lesson

\* @param models\entities\lessons $lesson

\* @param int $ll

\* @param int $lc

\*/

public function updateLessonLengthCount (lessons $lesson, $ll, $lc)

{

$lesson->setLessonlength($ll);

$lesson->setLessoncount($lc);

$this->entityManager->flush();

}

/\*\*

\*

\* Add New Lesson

\*

\* @param models\entities\subjects $subjects

\* @param models\entities\groups $group

\* @param models\entities\users $users

\* @param models\entities\rooms $room

\* @param int $lessonlength

\* @param int $lessoncount

\*/

public function addLesson($subjects, $group, $users, $room, $lessonlength, $lessoncount)

{

$newLesson = new lessons();

$newLesson->setSubjects($subjects);

$newLesson->setGroup($group);

$newLesson->setTeacher($users);

$newLesson->setRoom($room);

$newLesson->setLessonlength($lessonlength);

$newLesson->setLessoncount($lessoncount);

$this->entityManager->persist($newLesson);

$this->entityManager->flush();

}

}

?>

##### News.php

<?php

/\*\*

\* @author Aaron Redmond

\* @version 1.0

\* @created 29-Mar-2011

\*/

use models\entities\news;

use Doctrine\ORM\Query\Expr\Func;

class DAO\_news extends DAO\_Abstract

{

/\*\*

\*

\* Retrieve all news items

\* @return models\entities\news

\*/

public function retrieveNewsItems ()

{

$dquery = $this->entityManager->createQueryBuilder()

->select('u')

->from('models\entities\news', 'u')

->orderBy('u.publishedDate', 'DESC');

return $dquery->getQuery()->execute();

}

/\*\*

\*

\* Retrieve a news item by Id

\* @param int $id

\* @return models\entities\news || Null

\*/

public function retrieveNewsItemByID ($id)

{

try {

$dquery = $this->entityManager->createQueryBuilder()

->select('u')

->from('models\entities\news', 'u')

->where('u.newsid=?1')

->setParameter(1, $id);

return $dquery->getQuery()->getSingleResult();

} catch (Exception $e) {

return null;

}

}

/\*\*

\*

\* Add News Item

\* @param string $title

\* @param string $article

\* @param string $publishedDate

\* @return models\entities\news || Null

\*/

public function addNewsItem ($title ,$article, $publishedDate)

{

try {

$newNewsItem = new news();

$newNewsItem->setTitle($title);

$newNewsItem->setArticle($article);

$newNewsItem->setPublishedDate($publishedDate);

$this->entityManager->persist($newNewsItem);

$this->entityManager->flush();

} catch (Exception $e) {

return null;

}

}

/\*\*

\*

\* Delete News Item

\* @param models\entities\news $newsentity

\*/

public function deleteNewsItem (news $newsentity)

{

$this->entityManager->remove($newsentity);

$this->entityManager->flush();

}

/\*\*

\*

\* Update NewsItem

\* @param models\entities\news $newsItemEntity

\* @param string $title

\* @param string $article

\* @param string $publishedDate

\*/

public function updateNewsItem (news $newsItemEntity, $title, $article, $publishedDate)

{

try {

$newsItemEntity->setTitle($title);

$newsItemEntity->setArticle($article);

$newsItemEntity->setPublishedDate($publishedDate);

$this->entityManager->flush();

} catch (Exception $e) {

return null;

}

}

}

?>

##### Result.php

<?php

/\*\*

\*

\* A class that is passed back from all DAO Operations (CRUD)

\* It indication an outcome of the operation and carries possible error messages

\* and exceptions

\*/

class DAO\_Result

{

/\*\*

\*

\* Result of the a CRUD Operation

\* @var bool

\*/

private $outcome;

/\*\*

\*

\* Error message if any

\* @var string

\*/

private $message;

/\*\*

\*

\* Exception if any

\* @var Exception

\*/

private $exception;

/\*\*

\*

\* The object created during execution

\* @var unknown\_type

\*/

private $createdObject;

/\*\*

\*

\* @param bool outcome

\* @param Exception exception

\* @param string message

\*/

public function \_\_construct ($outcome = true, $exception = null, $message = null)

{

$this->setOutcome($outcome);

$this->setException($exception);

$this->setMessage($message);

}

/\*\*

\* @return bool $outcome

\*/

public function getOutcome ()

{

return $this->outcome;

}

/\*\*

\* @return string $message

\*/

public function getMessage ()

{

return $this->message;

}

/\*\*

\* @return Exception $exception

\*/

public function getException ()

{

return $this->exception;

}

/\*\*

\* @param bool $outcome

\*/

public function setOutcome ($outcome)

{

$this->outcome = $outcome;

}

/\*\*

\* @param string $message

\*/

public function setMessage ($message)

{

$this->message = $message;

}

/\*\*

\* @param Exception $exception

\*/

public function setException ( $exception)

{

$this->exception = $exception;

}

/\*\*

\* @return the $createdObject

\*/

public function getCreatedObject ()

{

return $this->createdObject;

}

/\*\*

\* @param unknown\_type $createdObject

\*/

public function setCreatedObject ($createdObject)

{

$this->createdObject = $createdObject;

}

}

?>

##### Rooms.php

<?php

/\*\*

\* @author Aaron Redmond

\* @version 1.0

\* @created 22-Feb-2011

\*/

use models\entities\rooms;

use Doctrine\ORM\Query\Expr\Func;

class DAO\_rooms extends DAO\_Abstract

{

/\*\*

\*

\* Retrieve all rooms

\* @return models\entities\rooms

\*/

public function retrieveRooms ()

{

$dquery = 'SELECT u from models\entities\rooms u';

return $this->entityManager->createQuery($dquery)->getResult();

}

/\*\*

\*

\* Retrieve a room by Id

\* @param $id

\* @return models\entities\rooms

\*/

public function retrieveRoomByID ($id)

{

try {

$dquery = $this->entityManager->createQueryBuilder()

->select('u')

->from('models\entities\rooms', 'u')

->where('u.roomid=?1')

->setParameter(1, $id);

return $dquery->getQuery()->getSingleResult();

} catch (Exception $e) {

return null;

}

}

/\*\*

\*

\* Update Room Information

\* @param models\entities\rooms $roomentity

\* @param string $roomName

\* @param string $roomShortName

\* @param int $capacity

\*/

public function updateRoomInfo ($roomentity, $roomName, $roomShortName, $capacity)

{

$roomentity->setRoomName($roomName);

$roomentity->setRoomShortName($roomShortName);

$roomentity->setCapacity($capacity);

$this->entityManager->flush();

}

/\*\*

\*

\* Delete Room

\* @param models\entities\rooms $roomentity

\*/

public function deleteRoom (rooms $roomentity)

{

$this->entityManager->remove($roomentity);

$this->entityManager->flush();

}

/\*\*

\*

\* Add Room

\* @param string $roomName

\* @param string $roomShortName

\* @param int $capacity

\*/

public function addRoom ($roomName, $roomShortName, $capacity)

{

$newRoom = new rooms();

$newRoom->setRoomName($roomName);

$newRoom->setRoomShortName($roomShortName);

$newRoom->setCapacity($capacity);

$this->entityManager->persist($newRoom);

$this->entityManager->flush();

}

}

?>

##### Schoolinfo.php

<?php

/\*\*

\* @author Aaron Redmond

\* @version 1.0

\* @created 21-Feb-2011

\*/

use models\entities\schoolinfo;

use Doctrine\ORM\Query\Expr\Func;

class DAO\_schoolinfo extends DAO\_Abstract

{

/\*\*

\*

\* Retrieve SchoolInfo

\* @return models\entities\schoolinfo

\*/

public function retriveSchoolInfo ()

{

try {

$dquery = $this->entityManager->createQueryBuilder()

->select('u')

->from('models\entities\schoolinfo', "u");

return $dquery->getQuery()->getSingleResult();

} catch (Exception $e) {

return null;

}

}

/\*\*

\*

\* Update School Information

\* @param models\entities\schoolinfo $schoolentity

\* @param string $schoolName

\* @param string $schoolAddress

\* @param string $schoolPhone

\* @param string $schoolEmail

\*/

public function updateSchoolInfo ($schoolentity, $schoolName, $schoolAddress,

$schoolPhone, $schoolEmail)

{

$schoolentity->setSchoolName($schoolName);

$schoolentity->setSchoolAddress($schoolAddress);

$schoolentity->setSchoolPhone($schoolPhone);

$schoolentity->setSchoolemail($schoolEmail);

$this->entityManager->flush();

}

/\*\*

\*

\* Update Banner image to be used

\* @param $schoolentity

\* @param string $image

\*/

public function changeImage($schoolentity, $image)

{

$schoolentity->setSchoolimage($image);

$this->entityManager->flush();

}

}

?>

##### Session.php

<?php

/\*\*

\*

\* Represents the session table in the database

\* @author Aaron Redmond

\* @version 1.0

\* @created 11-Feb-2011

\*/

use models\entities\users;

use models\entities\Session;

use models\entities\teachers;

class DAO\_Session extends DAO\_Abstract

{

/\*\*

\*

\* Server session variable

\*/

private $authSession;

/\*\*

\*

\* The sesseion entity

\* @var models\entities\Session

\*/

private $sessionEntity;

/\*\*

\*

\* The user entity associated with the session

\* @var models\entities\users

\*/

private $userEntity;

/\*\*

\*

\* Create new blank session

\*/

public function \_\_construct ()

{

parent::\_\_construct();

$this->authSession = new Zend\_Session\_Namespace('Zend\_Auth');

$this->init();

}

/\*\*

\*

\* Initilise the session

\*/

public function init ()

{

if (! isset($this->authSession->sessionId) || $this->authSession->sessionId == null) {

$this->authSession->sessionId = null;

$this->userEntity = null;

$this->sessionEntity = null;

} else {

$this->sessionEntity = $this->findSession($this->authSession->sessionId);

if ($this->sessionEntity && $this->sessionEntity->isValid()) {

$this->sessionEntity->extend();

$this->userEntity = $this->sessionEntity->getuserid();

$this->entityManager->persist($this->sessionEntity);

} else {

$this->destroySession();

}

}

}

/\*\*

\*

\* Create a session

\* @param users $users

\*/

public function create (users $users)

{

$session = new Session($users);

$users->setSession($session);

$this->entityManager->persist($session);

$this->entityManager->flush();

$this->sessionEntity = $session;

$this->userEntity = $users;

$sid = $session->getsessionid();

$this->authSession->sessionId = $this->sessionEntity->getsessionid();

}

/\*\*

\*

\* Destroys the session

\* @return DAO\_Result

\*/

public function destroySession ()

{

$Result = new DAO\_Result();

try {

$this->authSession->sessionId = null;

unset($this->authSession->sessionId);

$this->userEntity = null;

if ($this->sessionEntity) {

$this->entityManager->remove($this->sessionEntity);

}

$this->sessionEntity = null;

$this->userEntity = null;

} catch (Exception $e) {

$this->userEntity = null;

$this->sessionEntity = null;

$this->authSession->sessionId = null;

}

return $Result;

}

/\*\*

\*

\* Find Session by ID

\* @param string $sessionId

\* @return models\entities\Session | null

\*/

public function findSession ($sessionId)

{

try {

return $this->entityManager->find('models\entities\Session', $sessionId);

} catch (Exception $e) {

return null;

}

}

/\*\*

\*

\* Check to see if a user is logged in

\* @return bool

\*/

public function isLoggedIn ()

{

return isset($this->authSession->sessionId);

}

/\*\*

\*

\* Gets current user

\* @return models\entities\users

\*/

public function getUser ()

{

return $this->userEntity;

}

}

?>

##### Studentgrades.php

<?php

/\*\*

\* @author Aaron Redmond

\* @version 1.0

\* @created 05-Feb-2011

\*/

use models\entities\studentgrades;

use Doctrine\ORM\Query\Expr\Func;

class DAO\_studentgrades extends DAO\_Abstract

{

/\*\*

\*

\* Retrieve a student grade by Id

\* @param int $studentId

\* @param int $instrumentId

\* @return models\entities\studentgrades

\*/

public function retrieveStudentGradeByID ($studentId,$instrumentId)

{

try {

$dquery = $this->entityManager->createQueryBuilder()

->select('u')

->from('models\entities\studentgrades', 'u')

->where('u.studentid=?1')

->andWhere('u.instrumentid=?2')

->setParameter(1, $studentId)

->setParameter(2, $instrumentId);

return $dquery->getQuery()->getSingleResult();

} catch (Exception $e) {

return null;

}

}

/\*\*

\*

\* Add Student Instrument

\* @param int $studentID

\* @param int $instrumentID

\*/

public function addStudentGrade ($studentID, $instrumentID)

{

$newStudentGrades = new studentgrades();

$newStudentGrades->setStudentid($studentID);

$newStudentGrades->setInstrumentid($instrumentID);

$newStudentGrades->setGrade(0);

$this->entityManager->persist($newStudentGrades);

$this->entityManager->flush();

}

/\*\*

\*

\* Update Subject Information

\* @param models\entities\studentgrades $studentgradesentity

\* @param string $studentgrades

\*/

public function updateStudentGrade (studentgrades $studentgradesentity, $studentgrades)

{

$studentgradesentity->setGrade($studentgrades);

$this->entityManager->flush();

}

/\*\*

\*

\* Delete Student Grade

\* @param models\entities\studentgrades $studentgradeentitiy

\*/

public function removeStudentGrade (studentgrades $studentgradeentity)

{

$this->entityManager->remove($studentgradeentity);

$this->entityManager->flush();

}

}

?>

##### Subjects.php

<?php

/\*\*

\* @author Aaron Redmond

\* @version 1.0

\* @created 25-Feb-2011

\*/

use models\entities\subjects;

use Doctrine\ORM\Query\Expr\Func;

class DAO\_subjects extends DAO\_Abstract

{

/\*\*

\*

\* Retrieve all subjects

\* @return models\entities\subjects

\*/

public function retrieveSubjects ()

{

$dquery = 'SELECT u from models\entities\subjects u';

return $this->entityManager->createQuery($dquery)->getResult();

}

/\*\*

\*

\* Retrieve a subject by Id

\* @param int $id

\* @return models\entities\subjects

\*/

public function retrieveSubjectByID ($id)

{

try {

$dquery = $this->entityManager->createQueryBuilder()

->select('u')

->from('models\entities\subjects', 'u')

->where('u.subjectid=?1')

->setParameter(1, $id);

return $dquery->getQuery()->getSingleResult();

} catch (Exception $e) {

return null;

}

}

/\*\*

\*

\* Retrieve a subject by name

\* @param string $name

\* @return models\entities\subjects

\*/

public function retrieveSubjectByName ($name)

{

try {

$dquery = $this->entityManager->createQueryBuilder()

->select('u')

->from('models\entities\subjects', 'u')

->where('LOWER(u.subject)=?1')

->setParameter(1, $name);

return $dquery->getQuery()->getSingleResult();

} catch (Exception $e) {

return null;

}

}

/\*\*

\*

\* Update Subject Information

\* @param models\entities\subjects $subjectentity

\* @param string $subject

\*/

public function updateSubjectInfo ($subjectentity, $subject)

{

$subjectentity->setSubject($subject);

$this->entityManager->flush();

}

/\*\*

\*

\* Delete Subject

\* @param models\entities\subjects $subjectentity

\*/

public function deleteSubject (subjects $subjectentity)

{

$this->entityManager->remove($subjectentity);

$this->entityManager->flush();

}

/\*\*

\*

\* Add Subject

\* @param string $subject

\*/

public function addSubject ($subject)

{

$newSubject = new subjects();

$newSubject->setSubject($subject);

$this->entityManager->persist($newSubject);

$this->entityManager->flush();

}

}

?>

##### Teacherconstraints.php

<?php

/\*\*

\* @author Aaron Redmond

\* @version 1.0

\* @created 01-Apr-2011

\*/

use models\entities\teacherconstraints;

use Doctrine\ORM\Query\Expr\Func;

class DAO\_teacherconstraints extends DAO\_Abstract

{

/\*\*

\*

\* Retrieve all teacher constraints

\* @return models\entities\teacherconstraints

\*/

public function retrieveConstraints ()

{

$dquery = 'SELECT u from models\entities\teacherconstraints u';

return $this->entityManager->createQuery($dquery)->getResult();

}

/\*\*

\*

\* Retrieve Single Constraint based upon Id

\* @param int $Id

\* @return models\entities\teacherconstraints || Null

\*/

public function retrieveConstraintsById ($Id)

{

try {

$dquery = $this->entityManager->createQueryBuilder()

->select('u')

->from('models\entities\teacherconstraints', 'u')

->where('u.teacher\_constraint\_id=?1')

->setParameter(1, $Id);

return $dquery->getQuery()->getSingleResult();

} catch (Exception $e) {

return null;

}

}

/\*\*

\*

\* Retrieve Constraints for specific teacher

\* @param int $teacherId

\* @return models\entities\teacherconstraints || Null

\*/

public function retrieveConstraintsByTeacher ($teacherId)

{

try {

$dquery = $this->entityManager->createQueryBuilder()

->select('u')

->from('models\entities\teacherconstraints', 'u')

->where('u.teacher=?1')

->setParameter(1, $teacherId);

return $dquery->getQuery()->execute();

} catch (Exception $e) {

return null;

}

}

/\*\*

\*

\* Add an Individal Teacher Constraint

\* @param int $teacherId

\* @param string $day

\* @param string $time

\*/

public function addConstraint($teacherId, $day, $time)

{

$newConstraint = new teacherconstraints();

$newConstraint->setTeacher($teacherId);

$newConstraint->setDay($day);

$newConstraint->setTime($time);

$this->entityManager->persist($newConstraint);

$this->entityManager->flush();

}

/\*\*

\*

\* Delete a constraint

\* @param models\entities\teacherconstraints $teacherconstraintsentity

\*/

public function deleteConstraint (teacherconstraints $teacherconstraintsentity)

{

$this->entityManager->remove($teacherconstraintsentity);

$this->entityManager->flush();

}

}

?>

##### Timetableinfo.php

<?php

/\*\*

\* @author Aaron Redmond

\* @version 1.0

\* @created 17-Mar-2011

\*/

use models\entities\timetableinfo;

use Doctrine\ORM\Query\Expr\Func;

class DAO\_timetableinfo extends DAO\_Abstract

{

/\*\*

\*

\* Retrieve timetable Information

\* @return models\entities\timetableinfo

\*/

public function retrieveTimetableInfo ()

{

try {

$dquery = 'SELECT u from models\entities\timetableinfo u';

return $this->entityManager->createQuery($dquery)->getResult();

} catch (Exception $e) {

return null;

}

}

/\*\*

\*

\* Update Timetable Information

\* @param models/entities/timetableinfo $timetableInfoEntity

\* @param string $academicYear

\* @param int $daysPerWeek

\* @param int $lessonsPerDay

\* @param int $startTime

\* @param int $lessonLength

\*/

public function updateTimetableInfo ($timetableInfoEntity,

$academicYear,

$daysPerWeek,

$lessonsPerDay,

$startTime,

$lessonLength)

{

$timetableInfoEntity->setAcademicYear($academicYear);

$timetableInfoEntity->setDaysPerWeek($daysPerWeek);

$timetableInfoEntity->setLessonsPerDay($lessonsPerDay);

$timetableInfoEntity->setStartTime($startTime);

$timetableInfoEntity->setLessonLength($lessonLength);

$this->entityManager->flush();

}

/\*\*

\*

\* Add new Timetable Info

\* @param string $academicYear

\* @param int $daysPerWeek

\* @param int $lessonsPerDay

\* @param string $startTime

\* @param string $lessonLength

\*/

public function addTimetableInfo ($academicYear,

$daysPerWeek,

$lessonsPerDay,

$startTime,

$lessonLength)

{

$newTimetableInfo = new timetableinfo();

$newTimetableInfo->setAcademicYear($academicYear);

$newTimetableInfo->setDaysPerWeek($daysPerWeek);

$newTimetableInfo->setLessonsPerDay($lessonsPerDay);

$newTimetableInfo->setStartTime($startTime);

$newTimetableInfo->setLessonLength($lessonLength);

$this->entityManager->persist($newTimetableInfo);

$this->entityManager->flush();

}

}

?>

##### users.php

<?php

/\*\*

\* @author Aaron Redmond

\* @version 1.0

\* @created 11-Feb-2011

\*/

use models\entities\users;

use Doctrine\ORM\Query\Expr\Func;

class DAO\_users extends DAO\_Abstract

{

/\*\*

\*

\* Get User By Username/Password combination

\* @param string $username

\* @param string $userpassword

\* @return bool

\*/

public function checkLogin ($username, $userpassword)

{

$dquery = $this->entityManager->createQuery(

'select u from models\entities\users u where u.username = ?1 and u.userpassword =?2 ');

$result = $dquery->execute(array(1 => $username, 2 => md5($userpassword)));

if ($result) {

return true;

}

return false;

}

/\*\*

\*

\* Get User By Username/Password combination

\* @param string $username

\* @param string $userpassword

\* @return models\entities\users

\*/

public function authenticate ($username, $userpassword)

{

try {

$userValidator = new Validate\_User();

$passwordValidator = new Validate\_Password();

if (! $userValidator->isValid($username) || ! $passwordValidator->isValid($userpassword)) {

return null;

}

$query = $this->entityManager->createQueryBuilder()

->select('u')

->from('models\entities\users', 'u')

->where('u.username=?1')

->andWhere('u.userpassword=?2')

->setParameter(1, $username)

->setParameter(2, md5($userpassword));

return $query->getQuery()->getSingleResult();

} catch (Exception $e){

return null;

}

}

/\*\*

\*

\* Register User

\* @param string $username

\* @param string $email

\* @param string $password

\*/

public function registerUser ($username, $email, $password)

{

$newUser = new users();

$newUser->setUserName($username);

$newUser->setUserPassword($password);

$newUser->setUserForename('');

$newUser->setUserSurname('');

$newUser->setUserEmail($email);

$newUser->setUserAddress('');

$newUser->setUserTelephone('');

$newUser->setUserMobile('');

$newUser->setUserType(1);

$newUser->setUserVerified(1);

$this->entityManager->persist($newUser);

$this->entityManager->flush();

}

/\*\*

\*

\* Retrieve user of specified type admin|teacher|student

\* @param string $usertype

\* @return models\entities\users

\*/

public function retrieveSpecificUsers($usertype)

{

$dquery = $this->entityManager->createQueryBuilder()

->select('u')

->from('models\entities\users', 'u')

->where('u.usertype=?1')

->setParameter(1, $usertype);

return $dquery->getQuery()->execute();

}

/\*\*

\*

\* Retrieve user by ID

\* @param string $id

\* @return models\entities\users

\*/

public function retrieveUserByID ($id)

{

try {

$dquery = $this->entityManager->createQueryBuilder()

->select('u')

->from('models\entities\users', 'u')

->where('u.userid=?1')

->setParameter(1, $id);

return $dquery->getQuery()->getSingleResult();

} catch (Exception $e) {

return null;

}

}

/\*\*

\*

\* Retrieve User by Username

\* @param string $username

\* @return models\entities\users

\*/

public function retrieveUserByUsername ($username)

{

try {

$dquery = $this->entityManager->createQueryBuilder()

->select('u')

->from('models\entities\users', 'u')

->where('u.username=?1')

->setParameter(1, $username);

return $dquery->getQuery()->getSingleResult();

} catch (Exception $e) {

return null;

}

}

/\*\*

\*

\* Retrieve User by Email

\* @param string $email

\* @return models\entities\users

\*/

public function retrieveUserByUserEmail ($email)

{

try {

$dquery = $this->entityManager->createQueryBuilder()

->select('u')

->from('models\entities\users', 'u')

->where('u.useremail=?1')

->setParameter(1, $email);

return $dquery->getQuery()->getSingleResult();

} catch (Exception $e) {

return null;

}

}

/\*\*

\*

\* Update User Information

\* @param models\entities\users $usersentity

\* @param string $foreName

\* @param string $surName

\* @param string $email

\* @param string $address

\* @param string $phone

\* @param string $mobile

\*/

public function updateUserInfo ($usersentity, $foreName, $surName, $email, $address, $phone, $mobile)

{

$usersentity->setUserForename($foreName);

$usersentity->setUserSurname($surName);

$usersentity->setUserEmail($email);

$usersentity->setUserAddress($address);

$usersentity->setUserTelephone($phone);

$usersentity->setUserMobile($mobile);

$this->entityManager->flush();

}

/\*\*

\*

\* Update the users password

\* @param models/entities/users $usersentity

\* @param md5 string $password

\*/

public function updateUserPassword ($usersentity, $password)

{

$usersentity->setUserPassword($password);

$this->entityManager->flush();

}

/\*\*

\*

\* Update the users username

\* @param models/entities/users $usersentity

\* @param string $username

\*/

public function updateUserName ($usersentity, $username)

{

$usersentity->setUserName($username);

$this->entityManager->flush();

}

/\*\*

\*

\* Delete User

\* @param models/entities/users $userentity

\*/

public function deleteUser (users $userentity)

{

$this->entityManager->remove($userentity);

$this->entityManager->flush();

}

/\*\*

\*

\* Add a new user to the system

\* @param string $username

\* @param string $password

\* @param string $foreName

\* @param string $sureName

\* @param string $email

\* @param string $address

\* @param string $phone

\* @param string $mobile

\* @param string $type

\* @param int $verified

\*/

public function addUser ($username, $password, $foreName, $surName, $email,

$address, $phone, $mobile, $type, $verified)

{

$newUser = new users();

$newUser->setUserName($username);

$newUser->setUserPassword(md5($password));

$newUser->setUserForename($foreName);

$newUser->setUserSurname($surName);

$newUser->setUserEmail($email);

$newUser->setUserAddress($address);

$newUser->setUserTelephone($phone);

$newUser->setUserMobile($mobile);

$newUser->setUserType($type);

$newUser->setUserVerified($verified);

$this->entityManager->persist($newUser);

$this->entityManager->flush();

}

}

?>

#### Entities

##### Groupconstraints.php

<?php

/\*\*

\*

\* Represents the groupconstraints table in the database

\* @author Aaron Redmond

\* @version 1.0

\* @created 02-Apr-2011

\*/

namespace models\entities;

use Doctrine\Common\Collections\ArrayCollection;

/\*\*

\* @Entity

\* @Table(name="groupconstraints")

\*/

class groupconstraints

{

/\*\*

\* @var int

\* @Id @Column(name="id", type="integer")

\* @GeneratedValue(strategy="AUTO") \*/

protected $group\_constraint\_id;

/\*\*

\* @var int

\* @Column(name="group\_id", type="integer") \*/

protected $group;

/\*\*

\* @var string

\* @Column(name="day", type="string", length="100") \*/

protected $day;

/\*\*

\* @var string

\* @Column(name="time", type="string", length="100") \*/

protected $time;

/\*\*

\* @OneToMany(targetEntity="groups", mappedBy="groupconstraints") \*/

private $groups;

function \_\_construct()

{

$this->groups = new ArrayCollection();

}

/\*\*

\* @return the $group\_constraint\_id

\*/

public function getGroup\_constraint\_id() {

return $this->group\_constraint\_id;

}

/\*\*

\* @return the $group

\*/

public function getGroup() {

return $this->group;

}

/\*\*

\* @return the $day

\*/

public function getDay() {

return $this->day;

}

/\*\*

\* @return the $time

\*/

public function getTime() {

return $this->time;

}

/\*\*

\* @return the $groups

\*/

public function getGroups() {

return $this->groups;

}

/\*\*

\* @param int $group

\*/

public function setGroup($group) {

$this->group = $group;

}

/\*\*

\* @param string $day

\*/

public function setDay($day) {

$this->day = $day;

}

/\*\*

\* @param string $time

\*/

public function setTime($time) {

$this->time = $time;

}

/\*\*

\* @param field\_type $groups

\*/

public function setGroups($groups) {

$this->groups = $groups;

}

}

?>

##### Groups.php

<?php

/\*\*

\*

\* Represents the groups table in the database

\* @author Aaron Redmond

\* @version 1.0

\* @created 26-Feb-2011

\*/

namespace models\entities;

use Doctrine\Common\Collections\ArrayCollection;

/\*\*

\* @Entity

\* @Table(name="groups")

\*/

class groups

{

/\*\*

\* @var int

\* @Id @Column(name="group\_id", type="integer")

\* @GeneratedValue(strategy="AUTO") \*/

protected $groupid;

/\*\*

\* @var string

\* @Column(name="group\_name", type="string", length="255") \*/

protected $groupname;

/\*\*

\* @var string

\* @Column(name="group\_short\_name", type="string", length="100") \*/

protected $groupshortname;

/\*\*

\* @var models\entities\grouptype

\* @OneToOne(targetEntity="grouptype")

\* @JoinColumn(name="group\_type", referencedColumnName="grouptype\_id") \*/

private $grouptype;

/\*\*

\* @ManyToMany(targetEntity="users", mappedBy="groups") \*/

private $users;

///////////////////

//Get & Set Methods

///////////////////

function \_\_construct()

{

$this->users = new ArrayCollection();

$this->lesson = new ArrayCollection();

}

/\*\*

\* @return the $groupid

\*/

public function getGroupId ()

{

return $this->groupid;

}

/\*\*

\* @return the $groupname

\*/

public function getGroupName ()

{

return $this->groupname;

}

/\*\*

\* @return the $groupshortname

\*/

public function getGroupShortName ()

{

return $this->groupshortname;

}

/\*\*

\* @return models\entities\grouptype $grouptype

\*/

public function getGroupType ()

{

return $this->grouptype;

}

/\*\*

\*@return ArrayCollection

\*/

public function getUsers() {

return $this->users;

}

/\*\*

\* @param $groupname

\*/

public function setGroupName ($groupname)

{

$this->groupname = $groupname;

}

/\*\*

\* @param $groupshortname

\*/

public function setGroupShortName ($groupshortname)

{

$this->groupshortname = $groupshortname;

}

/\*\*

\* @param models\entities\grouptype $grouptype

\*/

public function setGroupType (grouptype $grouptype)

{

$this->grouptype = $grouptype;

}

/\*\*

\* @param ArrayCollection $users

\*/

public function setUsers ($users)

{

foreach ($users as $user) {

$user->addGroup($this);

$this->users->add($user);

}

}

/\*\*

\* @param ArrayCollection $users

\*/

public function removeUsers ($users)

{

foreach ($users as $user) {

$user->removeGroup($this);

}

}

}

?>

##### Grouptype.php

<?php

/\*\*

\*

\* Represents the grouptype table in the database

\* @author Aaron Redmond

\* @version 1.0

\* @created 26-Feb-2011

\*/

namespace models\entities;

use Doctrine\Common\Collections\ArrayCollection;

/\*\*

\* @Entity

\* @Table(name="grouptype")

\*/

class grouptype

{

/\*\*

\* @var int

\* @Id @Column(name="grouptype\_id", type="integer")

\* @GeneratedValue(strategy="AUTO") \*/

protected $grouptypeid;

/\*\*

\* @var string

\* @Column(name="grouptype\_name", type="string", length="255", unique=true) \*/

protected $grouptypename;

///////////////////

//Get & Set Methods

///////////////////

/\*\*

\* @return the $grouptypeid

\*/

public function getGroupTypeId ()

{

return $this->grouptypeid;

}

/\*\*

\* @return the $grouptypename

\*/

public function getGroupTypeName ()

{

return $this->grouptypename;

}

/\*\*

\* @param $grouptypename

\*/

public function setGroupTypeName ($grouptypename)

{

$this->grouptypename = $grouptypename;

}

}

?>

##### Instruments.php

<?php

/\*\*

\*

\* Represents the instruments table in the database

\* @author Aaron Redmond

\* @version 1.0

\* @created 04-Mar-2011

\*/

namespace models\entities;

use Doctrine\Common\Collections\ArrayCollection;

use models\entities\users;

/\*\*

\* @Entity

\* @Table(name="instruments")

\*/

class instruments

{

/\*\*

\* @var int

\* @Id @Column(type="integer", name="instrument\_id")

\* @GeneratedValue(strategy="AUTO") \*/

protected $instrumentid;

/\*\*

\* @var string

\* @Column(name="instrument\_name", type="string", length="100") \*/

protected $instrumentname;

/\*\*

\* @var int

\* @Column(name="instrument\_used", type="integer") \*/

protected $instrumentused;

/\*\*

\* @ManyToMany(targetEntity="users", mappedBy="instruments") \*/

private $users;

///////////////////

//Get & Set Methods

///////////////////

/\*\*

\* @return the $instrumentid

\*/

public function getInstrumentid() {

return $this->instrumentid;

}

/\*\*

\* @return the $instrumentname

\*/

public function getInstrumentname() {

return $this->instrumentname;

}

/\*\*

\* @return the $instrumentused

\*/

public function getInstrumentused() {

return $this->instrumentused;

}

/\*\*

\* @return the $users

\*/

public function getUsers() {

return $this->users;

}

/\*\*

\* @param int $instrumentid

\*/

public function setInstrumentid($instrumentid) {

$this->instrumentid = $instrumentid;

}

/\*\*

\* @param string $instrumentname

\*/

public function setInstrumentname($instrumentname) {

$this->instrumentname = $instrumentname;

}

/\*\*

\* @param int $instrumentused

\*/

public function setInstrumentused($instrumentused) {

$this->instrumentused = $instrumentused;

}

/\*\*

\* @param field\_type $users

\*/

public function setUsers($users) {

$this->users = $users;

}

}

?>

##### Lessons.php

<?php

/\*\*

\*

\* Represents the lessons table in the database

\* @author Aaron Redmond

\* @version 1.0

\* @created 07-Mar-2011

\*/

namespace models\entities;

use Doctrine\Common\Collections\ArrayCollection;

/\*\*

\* @Entity

\* @Table(name="lessons")

\*/

class lessons

{

/\*\*

\* @var int

\* @Id @Column(name="lesson\_id", type="integer")

\* @GeneratedValue(strategy="AUTO") \*/

protected $lessonid;

/\*\*

\* @OneToOne(targetEntity="subjects")

\* @JoinColumn(name="subject", referencedColumnName="subject\_id")

\*/

private $subjects;

/\*\*

\* @var entities\models\groups

\* @ManyToOne(targetEntity="groups")

\* @JoinColumn(name="group\_id", referencedColumnName="group\_id")

\*/

private $group;

/\*\*

\* @var entities\models\users

\* @ManyToOne(targetEntity="users", cascade={"persist"})

\* @JoinColumn(name="teacher", referencedColumnName="user\_id")

\*/

private $teacher;

/\*\*

\* @var entities\models\rooms

\* @ManyToOne(targetEntity="rooms", cascade={"persist"})

\* @JoinColumn(name="room", referencedColumnName="room\_id")

\*/

private $room;

/\*\*

\* @var int

\* @Column(name="lesson\_length", type="integer") \*/

protected $lessonlength;

/\*\*

\* @var int

\* @Column(name="lesson\_count", type="integer") \*/

protected $lessoncount;

///////////////////

//Get & Set Methods

///////////////////

/\*\*

\* @return the $lessonid

\*/

public function getLessonid() {

return $this->lessonid;

}

/\*\*

\* @return the $subject

\*/

public function getSubjects() {

return $this->subjects;

}

/\*\*

\* @return the $group

\*/

public function getGroup() {

return $this->group;

}

/\*\*

\* @return the $user

\*/

public function getTeacher() {

return $this->teacher;

}

/\*\*

\* @return the $room

\*/

public function getRoom() {

return $this->room;

}

/\*\*

\* @return the $lessonlength

\*/

public function getLessonlength() {

return $this->lessonlength;

}

/\*\*

\* @return the $lessoncount

\*/

public function getLessoncount() {

return $this->lessoncount;

}

/\*\*

\* @param field\_type $subject

\*/

public function setSubjects($subjects) {

$this->subjects = $subjects;

}

/\*\*

\* @param field\_type $group

\*/

public function setGroup($group) {

$this->group = $group;

}

/\*\*

\* @param field\_type $users

\*/

public function setTeacher($users) {

$this->teacher = $users;

}

/\*\*

\* @param field\_type $room

\*/

public function setRoom($room) {

$this->room = $room;

}

/\*\*

\* @param int $lessonlength

\*/

public function setLessonlength($lessonlength) {

$this->lessonlength = $lessonlength;

}

/\*\*

\* @param int $lessoncount

\*/

public function setLessoncount($lessoncount) {

$this->lessoncount = $lessoncount;

}

}

?>

##### News.php

<?php

/\*\*

\*

\* Represents the news table in the database

\* @author Aaron Redmond

\* @version 1.0

\* @created 29-Mar-2011

\*/

namespace models\entities;

use Doctrine\Common\Collections\ArrayCollection;

/\*\*

\* @Entity

\* @Table(name="news")

\*/

class news

{

/\*\*

\* @var int

\* @Id @Column(name="news\_id", type="integer")

\* @GeneratedValue(strategy="AUTO") \*/

protected $newsid;

/\*\*

\* @var string

\* @Column(name="title", type="string", length="255") \*/

protected $title;

/\*\*

\* @var string

\* @Column(name="news\_item", type="string", length="255") \*/

protected $article;

/\*\*

\* @var string

\* @Column(name="published\_date", type="string", length="50") \*/

protected $publishedDate;

///////////////////

//Get & Set Methods

///////////////////

/\*\*

\* @return the $newsid

\*/

public function getNewsid() {

return $this->newsid;

}

/\*\*

\* @return the $title

\*/

public function getTitle() {

return $this->title;

}

/\*\*

\* @return the $article

\*/

public function getArticle() {

return $this->article;

}

/\*\*

\* @return the $publishedDate

\*/

public function getPublishedDate() {

return $this->publishedDate;

}

/\*\*

\* @param string $title

\*/

public function setTitle($title) {

$this->title = $title;

}

/\*\*

\* @param string $article

\*/

public function setArticle($article) {

$this->article = $article;

}

/\*\*

\* @param string $publishedDate

\*/

public function setPublishedDate($publishedDate) {

$this->publishedDate = $publishedDate;

}

}

?>

##### Rooms.php

<?php

/\*\*

\*

\* Represents the room table in the database

\* @author Aaron Redmond

\* @version 1.0

\* @created 22-Feb-2011

\*/

namespace models\entities;

use Doctrine\Common\Collections\ArrayCollection;

/\*\*

\* @Entity

\* @Table(name="rooms")

\*/

class rooms

{

/\*\*

\* @var int

\* @Id @Column(name="room\_id", type="integer")

\* @GeneratedValue(strategy="AUTO") \*/

protected $roomid;

/\*\*

\* @var string

\* @Column(name="room\_name", type="string", length="255") \*/

protected $roomname;

/\*\*

\* @var string

\* @Column(name="room\_short\_name", type="string", length="100") \*/

protected $roomshortname;

/\*\*

\* @var int

\* @Column(name="capacity", type="integer") \*/

protected $capacity;

///////////////////

//Get & Set Methods

///////////////////

/\*\*

\* @return the $roomid

\*/

public function getRoomId ()

{

return $this->roomid;

}

/\*\*

\* @return the $roomname

\*/

public function getRoomName ()

{

return $this->roomname;

}

/\*\*

\* @return the $roomshortname

\*/

public function getRoomShortName ()

{

return $this->roomshortname;

}

/\*\*

\* @return the $capacity

\*/

public function getCapacity() {

return $this->capacity;

}

/\*\*

\* @param string $roomname

\*/

public function setRoomName ($roomname)

{

$this->roomname = $roomname;

}

/\*\*

\* @param string $roomshortname

\*/

public function setRoomShortName ($roomshortname)

{

$this->roomshortname = $roomshortname;

}

/\*\*

\* @param int $capacity

\*/

public function setCapacity($capacity) {

$this->capacity = $capacity;

}

}

?>

##### Schoolinfo.php

<?php

/\*\*

\*

\* Represents the schoolinfo table in the database

\* @author Aaron Redmond

\* @version 1.0

\* @created 21-Feb-2011

\*/

namespace models\entities;

use Doctrine\Common\Collections\ArrayCollection;

/\*\*

\* @Entity

\* @Table(name="schoolinfo")

\*/

class schoolinfo

{

/\*\*

\* @var int

\* @Id @Column(name="school\_id", type="integer")

\* @GeneratedValue(strategy="AUTO") \*/

protected $schoolid;

/\*\*

\* @var string

\* @Column(name="school\_name", type="string", length="255") \*/

protected $schoolname;

/\*\*

\* @var string

\* @Column(name="school\_address", type="string", length="255") \*/

protected $schooladdress;

/\*\*

\* @var string

\* @Column(name="school\_phone", type="string", length="20") \*/

protected $schoolphone;

/\*\*

\* @var string

\* @Column(name="school\_email", type="string", length="100") \*/

protected $schoolemail;

/\*\*

\* @var string

\* @Column(name="school\_image", type="string", length="255") \*/

protected $schoolimage;

///////////////////

//Get & Set Methods

///////////////////

/\*\*

\* @return the $schoolid

\*/

public function getSchoolId ()

{

return $this->schoolid;

}

/\*\*

\* @return the $schoolname

\*/

public function getSchoolName ()

{

return $this->schoolname;

}

/\*\*

\* @return the $schooladdress

\*/

public function getSchoolAddress ()

{

return $this->schooladdress;

}

/\*\*

\* @return the $schoolphone

\*/

public function getSchoolPhone ()

{

return $this->schoolphone;

}

/\*\*

\* @return the $schoolemail

\*/

public function getSchoolemail ()

{

return $this->schoolemail;

}

/\*\*

\* @return the $schoolimage

\*/

public function getSchoolimage() {

return $this->schoolimage;

}

/\*\*

\* @param string $schoolname

\*/

public function setSchoolName ($schoolname)

{

$this->schoolname = $schoolname;

}

/\*\*

\* @param string $schooladdress

\*/

public function setSchoolAddress ($schooladdress)

{

$this->schooladdress = $schooladdress;

}

/\*\*

\* @param string $schoolphone

\*/

public function setSchoolPhone ($schoolphone)

{

$this->schoolphone = $schoolphone;

}

/\*\*

\* @param string $schoolemail

\*/

public function setSchoolemail ($schoolemail)

{

$this->schoolemail = $schoolemail;

}

/\*\*

\* @param string $schoolimage

\*/

public function setSchoolimage($schoolimage) {

$this->schoolimage = $schoolimage;

}

}

?>

##### Session.php

<?php

/\*\*

\*

\* Represents the session table in the database

\* @author Aaron Redmond

\* @version 1.0

\* @created 11-Feb-2011

\*/

namespace models\entities;

use models\entities\users;

/\*\*

\* @Entity

\* @Table(name="session")

\*/

class Session

{

////////////////////////////////////////////////////////////////

//Variables which represent the session table in the database

////////////////////////////////////////////////////////////////

/\*\*

\* @var string

\* @Id @Column(type="string", name="session\_id") \*/

protected $sessionid;

/\*\*

\* @ManyToOne(targetEntity="users")

\* @JoinColumn(name="user\_id", referencedColumnName="user\_id")

\*/

protected $userid;

/\*\*

\* @var int

\* @Column(name="valid\_til", type="bigint") \*/

protected $validtil;

/\*\*

\* @var int

\* @Column(name="created\_at", type="bigint") \*/

protected $createdat;

/\*\*

\*

\* 1 Hour

\* @var int

\*/

private $sessionDuration = 3600;

///////////////////

//Get & Set Methods

///////////////////

/\*\*

\* @return the $sessionid

\*/

public function getsessionid ()

{

return $this->sessionid;

}

/\*\*

\* @return the $userid

\*/

public function getuserid ()

{

return $this->userid;

}

/\*\*

\* @return the $validtil

\*/

public function getvalidtil ()

{

return $this->validtil;

}

/\*\*

\* @return the $createdat

\*/

public function getcreatedat ()

{

return $this->createdat;

}

/\*\*

\* Set the sessionid

\*/

public function setsessionid ()

{

$this->sessionid = md5(rand(10000, 99999) . time() . rand(10000, 99999));

}

///////////////////

//Session Functions

///////////////////

/\*\*

\*

\* Create a Session

\*/

function \_\_construct (users $userid)

{

$this->setsessionid();

$this->createdat = time();

$this->extend();

$this->userid = $userid;

}

/\*\*

\*

\* Extend the session length by the time specified in sessionDuration

\*/

public function extend ()

{

$this->validtil = time() + $this->sessionDuration;

}

/\*\*

\*

\* Determine if the session is still valid

\* @return bool

\*/

public function isValid ()

{

return time() < $this->validtil;

}

}

?>

##### Studentgrades.php

<?php

/\*\*

\*

\* Represents the studentgrades table in the database

\* @author Aaron Redmond

\* @version 1.0

\* @created 05-Feb-2011

\*/

namespace models\entities;

use Doctrine\Common\Collections\ArrayCollection;

/\*\*

\* @Entity

\* @Table(name="studentgrades")

\*/

class studentgrades

{

/\*\*

\* @var int

\* @Id @Column(type="integer", name="studentgrade\_id")

\* @GeneratedValue(strategy="AUTO") \*/

protected $studentgradeid;

/\*\*

\* @var int

\* @Column(name="student\_id", type="integer") \*/

protected $studentid;

/\*\*

\* @var int

\* @Column(name="instrument\_id", type="integer") \*/

protected $instrumentid;

/\*\*

\* @var int

\* @Column(name="grade", type="integer") \*/

protected $grade;

///////////////////

//Get & Set Methods

///////////////////

/\*\*

\* @return the $studentid

\*/

public function getStudentid() {

return $this->studentid;

}

/\*\*

\* @return the $instrumentid

\*/

public function getInstrumentid() {

return $this->instrumentid;

}

/\*\*

\* @return the $grade

\*/

public function getGrade() {

return $this->grade;

}

/\*\*

\* @param int $studentid

\*/

public function setStudentid($studentid) {

$this->studentid = $studentid;

}

/\*\*

\* @param int $instrumentid

\*/

public function setInstrumentid($instrumentid) {

$this->instrumentid = $instrumentid;

}

/\*\*

\* @param int $grade

\*/

public function setGrade($grade) {

$this->grade = $grade;

}

}

?>

##### Subjects.php

<?php

/\*\*

\*

\* Represent the subjects table in the database

\* @author Aaron Redmond

\* @version 1.0

\* @created 25-Feb-2011

\*/

namespace models\entities;

use Doctrine\Common\Collections\ArrayCollection;

/\*\*

\* @Entity

\* @Table(name="subjects")

\*/

class subjects

{

/\*\*

\* @var int

\* @Id @Column(name="subject\_id", type="integer")

\* @GeneratedValue(strategy="AUTO") \*/

private $subjectid;

/\*\*

\* @var string

\* @Column(name="subject", type="string", length="255") \*/

private $subject;

///////////////////

//Get & Set Methods

///////////////////

/\*\*

\* @return the $subjectid

\*/

public function getSubjectid() {

return $this->subjectid;

}

/\*\*

\* @return the $subject

\*/

public function getSubject() {

return $this->subject;

}

/\*\*

\* @param string $subject

\*/

public function setSubject($subject) {

$this->subject = $subject;

}

}

?>

##### Teacherconstraints.php

<?php

/\*\*

\*

\* Represents the teacherconstraints Table in the Database

\* @author Aaron Redmond

\* @version 1.0

\* @created 01-Apr-2011

\*/

namespace models\entities;

use Doctrine\Common\Collections\ArrayCollection;

/\*\*

\* @Entity

\* @Table(name="teacherconstraints")

\*/

class teacherconstraints

{

/\*\*

\* @var int

\* @Id @Column(name="id", type="integer")

\* @GeneratedValue(strategy="AUTO") \*/

protected $teacher\_constraint\_id;

/\*\*

\* @var int

\* @Column(name="teacher\_id", type="integer") \*/

protected $teacher;

/\*\*

\* @var string

\* @Column(name="day", type="string", length="100") \*/

protected $day;

/\*\*

\* @var string

\* @Column(name="time", type="string", length="100") \*/

protected $time;

/\*\*

\* @OneToMany(targetEntity="users", mappedBy="teacherconstraints") \*/

private $users;

function \_\_construct()

{

$this->users = new ArrayCollection();

}

/\*\*

\* @return the $teacher\_constraint\_id

\*/

public function getTeacher\_constraint\_id() {

return $this->teacher\_constraint\_id;

}

/\*\*

\* @return the $teacher

\*/

public function getTeacher() {

return $this->teacher;

}

/\*\*

\* @return the $day

\*/

public function getDay() {

return $this->day;

}

/\*\*

\* @return the $time

\*/

public function getTime() {

return $this->time;

}

/\*\*

\* @return ArrayCollection

\*/

public function getUsers() {

return $this->users;

}

/\*\*

\* @param int $teacher

\*/

public function setTeacher($teacher) {

$this->teacher = $teacher;

}

/\*\*

\* @param string $day

\*/

public function setDay($day) {

$this->day = $day;

}

/\*\*

\* @param string $time

\*/

public function setTime($time) {

$this->time = $time;

}

/\*\*

\* @param ArrayCollection $users

\*/

public function setUsers ($users)

{

foreach ($users as $user) {

$user->addGroup($this);

$this->users->add($user);

}

}

/\*\*

\* @param ArrayCollection $users

\*/

public function removeUsers ($users)

{

foreach ($users as $user) {

$user->removeGroup($this);

}

}

}

?>

##### Timetableinfo.php

<?php

/\*\*

\*

\* Represents the Timetableinfo Table in the Database

\* @author Aaron Redmond

\* @version 1.0

\* @created 17-Mar-2011

\*/

namespace models\entities;

use Doctrine\Common\Collections\ArrayCollection;

/\*\*

\* @Entity

\* @Table(name="timetableinfo")

\*/

class timetableinfo

{

/\*\*

\* @var int

\* @Id @Column(name="ttinfo\_id", type="integer")

\* @GeneratedValue(strategy="AUTO") \*/

protected $ttInfoId;

/\*\*

\* @var string

\* @Column(name="academic\_year", type="string", length="15") \*/

protected $academicYear;

/\*\*

\* @var int

\* @Column(name="days\_per\_week", type="integer") \*/

protected $daysPerWeek;

/\*\*

\* @var int

\* @Column(name="lessons\_per\_day", type="integer") \*/

protected $lessonsPerDay;

/\*\*

\* @var string

\* @Column(name="start\_time", type="string") \*/

protected $startTime;

/\*\*

\* @var string

\* @Column(name="lesson\_length", type="string") \*/

protected $lessonLength;

///////////////////

//Get & Set Methods

///////////////////

/\*\*

\* @return the $ttInfoId

\*/

public function getTtInfoId() {

return $this->ttInfoId;

}

/\*\*

\* @return the $academicYear

\*/

public function getAcademicYear() {

return $this->academicYear;

}

/\*\*

\* @return the $daysPerWeek

\*/

public function getDaysPerWeek() {

return $this->daysPerWeek;

}

/\*\*

\* @return the $lessonsPerDay

\*/

public function getLessonsPerDay() {

return $this->lessonsPerDay;

}

/\*\*

\* @return the $startTime

\*/

public function getStartTime() {

return $this->startTime;

}

/\*\*

\* @return the $lessonLength

\*/

public function getLessonLength() {

return $this->lessonLength;

}

/\*\*

\* @param string $academicYear

\*/

public function setAcademicYear($academicYear) {

$this->academicYear = $academicYear;

}

/\*\*

\* @param int $daysPerWeek

\*/

public function setDaysPerWeek($daysPerWeek) {

$this->daysPerWeek = $daysPerWeek;

}

/\*\*

\* @param int $lessonsPerDay

\*/

public function setLessonsPerDay($lessonsPerDay) {

$this->lessonsPerDay = $lessonsPerDay;

}

/\*\*

\* @param time $startTime

\*/

public function setStartTime($startTime) {

$this->startTime = $startTime;

}

/\*\*

\* @param string $lessonLength

\*/

public function setLessonLength($lessonLength) {

$this->lessonLength = $lessonLength;

}

}

?>

##### users.php

<?php

/\*\*

\*

\* Represents the Users Table in the Database

\* @author Aaron Redmond

\* @version 1.0

\* @created 11-Feb-2011

\*/

namespace models\entities;

use Doctrine\Common\Collections\ArrayCollection;

use models\entities\groups;

/\*\*

\* @Entity

\* @Table(name="users")

\*/

class users

{

/\*\*

\* @var int

\* @Id @Column(type="integer", name="user\_id")

\* @GeneratedValue(strategy="AUTO") \*/

protected $userid;

/\*\*

\* @var string

\* @Column(type="string", length="20") \*/

protected $username;

/\*\*

\* @var string

\* @Column(name="user\_password", type="string", length="32") \*/

protected $userpassword;

/\*\*

\* @var string

\* @Column(name="user\_forename", type="string", length="255") \*/

protected $userforename;

/\*\*

\* @var string

\* @Column(name="user\_surname", type="string", length="255") \*/

protected $usersurname;

/\*\*

\* @var string

\* @Column(name="user\_email", type="string", length="255", unique="true") \*/

protected $useremail;

/\*\*

\* @var string

\* @Column(name="user\_address", type="string", length="255") \*/

protected $useraddress;

/\*\*

\* @var string

\* @Column(name="user\_telephone", type="string", length="50") \*/

protected $usertelephone;

/\*\*

\* @var string

\* @Column(name="user\_mobile", type="string", length="50") \*/

protected $usermobile;

/\*\*

\* @var string

\* @Column(name="user\_type", type="string", length="10") \*/

protected $usertype;

/\*\*

\* @var int

\* @Column(name="user\_verified", type="smallint", length="1") \*/

protected $userverified;

/\*\*

\* @OneToMany(targetEntity="Session", mappedBy="user")

\* @var ArrayCollection

\*/

protected $session;

/\*\*

\* List of groups the users belong to

\* @var ArrayCollection

\* @ManyToMany(targetEntity="groups", inversedBy="users", cascade={"all"}, fetch="EAGER")

\* @JoinTable(name="groupparticipants",

\* joinColumns={@JoinColumn(name="user", referencedColumnName="user\_id")},

\* inverseJoinColumns={@JoinColumn(name="group\_id", referencedColumnName="group\_id")}

\* )

\*/

private $groups;

/\*\*

\* List of users instruments

\* @var ArrayCollection

\* @ManyToMany(targetEntity="instruments", inversedBy="users", cascade={"all"})

\* @JoinTable(name="studentgrades",

\* joinColumns={@JoinColumn(name="student\_id", referencedColumnName="user\_id")},

\* inverseJoinColumns={@JoinColumn(name="instrument\_id", referencedColumnName="instrument\_id")}

\* )

\*/

private $instruments;

///////////////////

//Get & Set Methods

///////////////////

function \_\_construct ()

{

$this->session = new ArrayCollection();

$this->groups = new ArrayCollection();

$this->instruments = new ArrayCollection();

}

/\*\*

\* @return the $userid

\*/

public function getUserId ()

{

return $this->userid;

}

/\*\*

\* @return the $username

\*/

public function getUserName ()

{

return $this->username;

}

/\*\*

\* @return the $userpassword

\*/

public function getUserPassword ()

{

return $this->userpassword;

}

/\*\*

\* @return the $userforename

\*/

public function getUserForename ()

{

return $this->userforename;

}

/\*\*

\* @return the $usersurname

\*/

public function getUserSurname ()

{

return $this->usersurname;

}

/\*\*

\* @return the $useremail

\*/

public function getUserEmail ()

{

return $this->useremail;

}

/\*\*

\* @return the $useraddress

\*/

public function getUserAddress ()

{

return $this->useraddress;

}

/\*\*

\* @return the $usertelephone

\*/

public function getUserTelephone ()

{

return $this->usertelephone;

}

/\*\*

\* @return the $usermobile

\*/

public function getUserMobile ()

{

return $this->usermobile;

}

/\*\*

\* @return the $usertype

\*/

public function getUserType ()

{

return $this->usertype;

}

/\*\*

\* @return the $userverified

\*/

public function getUserVerified ()

{

return $this->userverified;

}

/\*\*

\* Get all users groups

\* @return ArrayCollection

\*/

public function getGroups ()

{

return $this->groups;

}

/\*\*

\* @return the $instruments

\*/

public function getInstruments() {

return $this->instruments;

}

/\*\*

\* @param string $username

\*/

public function setUserName ($username)

{

$this->username = $username;

}

/\*\*

\* @param string $userpassword

\*/

public function setUserPassword ($userpassword)

{

$this->userpassword = $userpassword;

}

/\*\*

\* @param string $userforename

\*/

public function setUserForename ($userforename)

{

$this->userforename = $userforename;

}

/\*\*

\* @param string $usersurname

\*/

public function setUserSurname ($usersurname)

{

$this->usersurname = $usersurname;

}

/\*\*

\* @param string $useremail

\*/

public function setUserEmail ($useremail)

{

$this->useremail = $useremail;

}

/\*\*

\* @param string $useraddress

\*/

public function setUserAddress ($useraddress)

{

$this->useraddress = $useraddress;

}

/\*\*

\* @param string $usertelephone

\*/

public function setUserTelephone ($usertelephone)

{

$this->usertelephone = $usertelephone;

}

/\*\*

\* @param string $usermobile

\*/

public function setUserMobile ($usermobile)

{

$this->usermobile = $usermobile;

}

/\*\*

\* @param string $usertype

\*/

public function setUserType ($usertype)

{

$this->usertype = $usertype;

}

/\*\*

\* @param tinyint $userverified

\*/

public function setUserVerified ($userverified)

{

$this->userverified = $userverified;

}

/\*\*

\*

\* @param session $session

\*/

public function setSession (Session $session)

{

$this->session->add($session);

}

/\*\*

\* @param ArrayCollection $groups

\*/

public function setGroups($groups) {

$this->groups = $groups;

}

/\*\*

\*

\* @param unknown\_type $group

\*/

public function addGroup($group) {

$this->groups->add($group);

}

/\*\*

\*

\* @param unknown\_type $group

\*/

public function removeGroup($group) {

$this->groups->removeElement($group);

}

/\*\*

\* @param ArrayCollection $instruments

\*/

public function setInstruments($instruments) {

$this->instruments = $instruments;

}

}

?>

### Proxies

These files are automatically generated by the Doctrine ORM, and so will not be listed here.

### Modules

This section contains the Controllers and their related views.

#### Admin

##### Controllers

###### IndexController

<?php

/\*\*

\* Admin IndexController

\*

\* @author Aaron Redmond

\* @version 1.0

\*/

use Doctrine\ORM\Query\Expr\Func;

require\_once APPLICATION\_PATH . '/modules/AbstractController.php';

use models\entities\users;

use models\entities\Session;

class Admin\_IndexController extends AbstractController

{

/\*\*

\*Check to see if the user is logged in

\*This function is applied to all pages

\*/

public function init ()

{

if (! ($this->DAOSession->isLoggedIn())) {

$this->\_redirect('/Login');

}

if (!($this->DAOSession->getUser()->getUserType() == 'admin')){

$this->\_redirect('/Login');

}

}

/\*\*

\* The default action - show the Admin home page

\*/

public function indexAction ()

{

//Get current users name

$userEntity = $this->DAOSession->getUser();

$userName = $userEntity->getUserForename();

$this->view->userName = $userName;

$this->\_helper->layout->setLayout('/admin/admin\_layout');

}

/\*\*

\* Show the Manage page

\*/

public function manageAction()

{

$this->\_helper->layout->setLayout('admin/admin\_layout');

}

/\*\*

\* Show the Timetabling page

\*/

public function timetablingAction()

{

$this->\_helper->layout->setLayout('admin/admin\_layout');

}

/\*\*

\* Show the User Panel page

\*/

public function userpanelAction()

{

$this->\_helper->layout->setLayout('admin/admin\_layout');

}

/\*\*

\* Show the System Setup page

\*/

public function systemsetupAction()

{

$this->\_helper->layout->setLayout('admin/admin\_layout');

}

/\*\*

\* Show the My Account page

\*/

public function myaccountAction()

{

$this->\_helper->layout->setLayout('admin/admin\_layout');

}

/\*\*

\* Show the Help page

\*/

**public function** helpAction()

{

$this->\_helper->layout->setLayout('admin/admin\_layout');

}

/\*\*

\* Destroy Session and redirect user to login page

\*/

public function logoutAction()

{

$this->DAOSession->destroySession();

$this->\_redirect('/Login');

}

}

###### manageController

<?php

/\*\*

\* Admin manageController

\*

\* @author Aaron Redmond

\* @version 1.0

\*/

use Doctrine\ORM\PersistentCollection;

use Doctrine\Common\Collections\ArrayCollection;

use Doctrine\DBAL\Types\StringType;

use Doctrine\ORM\Query\Expr\Func;

require\_once APPLICATION\_PATH . '/modules/AbstractController.php';

use models\entities\groupconstraints;

use models\entities\teacherconstraints;

use models\entities\grouptype;

use models\entities\lessons;

use models\entities\groups;

use models\entities\teachers;

use models\entities\users;

use models\entities\Session;

use models\entities\schoolinfo;

use models\entities\rooms;

use models\entities\subjects;

use models\entities\instruments;

class Admin\_ManageController extends AbstractController

{

/\*\*

\*Check to see if the user is logged in

\*This function is applied to all pages

\*/

public function init ()

{

if (! ($this->DAOSession->isLoggedIn())) {

$this->\_redirect('/Login');

}

if (!($this->DAOSession->getUser()->getUserType() == 'admin')){

$this->\_redirect('/Login');

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//School Information

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*\*

\*

\* Display School Information

\*/

public function schoolAction ()

{

//Retrieve all school Info

$DAOschoolInfo = new DAO\_schoolinfo();

$schoolinfo = $DAOschoolInfo->retriveSchoolInfo();

//Create New School Info Form

$editButton = new Form\_Admin\_manage\_school();

$this->view->schoolname = $schoolinfo->getSchoolName();

$this->view->schooladdress = $schoolinfo->getSchoolAddress();

$this->view->schoolPhone = $schoolinfo->getSchoolPhone();

$this->view->schoolEmail = $schoolinfo->getSchoolemail();

$this->view->editButton = $editButton;

$this->\_helper->layout->setLayout('/admin/admin\_layout');

}

/\*\*

\*

\* Show the Admin->Manage->EditSchool page

\*/

public function editschoolAction ()

{

//Retrieve School Info

$DAOschoolInfo = new DAO\_schoolinfo();

$schoolinfo = $DAOschoolInfo->retriveSchoolInfo();

//Create Edit School Form

$editSchoolForm = new Form\_Admin\_manage\_editschool($schoolinfo->getSchoolName(), $schoolinfo->getSchoolAddress(),

$schoolinfo->getSchoolPhone(),

$schoolinfo->getSchoolemail());

//Save Edited School Info

if ($this->\_request->isPost() && $editSchoolForm->isValid($\_POST)) {

$values = $editSchoolForm->getValues();

$updateschoolinfo = $DAOschoolInfo->updateSchoolInfo($schoolinfo,

$values['nameEditBox'],

$values['addressEditBox'],

$values['phoneEditBox'], $values['emailEditBox']);

$this->\_redirect('/admin/manage/school');

}

$this->view->schoolname = $schoolinfo->getSchoolName();

$this->view->schooladdress = $schoolinfo->getSchoolAddress();

$this->view->schoolPhone = $schoolinfo->getSchoolPhone();

$this->view->schoolEmail = $schoolinfo->getSchoolemail();

$this->view->editSchoolForm = $editSchoolForm;

$this->\_helper->layout->setLayout('/admin/admin\_layout');

}

/\*\*

\* Show the Admin->Manage->Rooms page

\* Retrieve Room data

\* Create Form containing rooms data

\*/

public function roomsAction ()

{

//Retrieve all Rooms

$DAOrooms = new DAO\_rooms();

$rooms = $DAOrooms->retrieveRooms();

//Render Rooms into table

$tableContent = "";

foreach ($rooms as $room) {

$tableContent .= $this->renderTable($room);

}

//Create New RoomsForm

$roomForm = new Form\_Admin\_manage\_rooms();

if ($tableContent == ""){

$tableContent = "<tr><td>No rooms available</td></tr>";

}

$this->view->tableContent = $tableContent;

$this->view->roomForm = $roomForm;

$this->\_helper->layout->setLayout('/admin/admin\_layout');

}

/\*\*

\* Renders a room into a \_room-item template

\* @param models\entities\rooms $rooms

\* @return /manage/\_room-item.phtml

\*/

private function renderTable (rooms $rooms)

{

$tableItem = new Zend\_View();

$tableItem->setScriptPath($this->view->getScriptPaths());

$tableItem->roomID = $rooms->getRoomId();

$tableItem->roomName = $rooms->getRoomName();

$tableItem->roomShortName = $rooms->getRoomShortName();

$tableItem->capacity = $rooms->getCapacity();

//Do not show the ANY room,this is used as a random room

if (!($rooms->getRoomName() == "Any")) {

return $tableItem->render('/manage/\_room-item.phtml');

}

}

/\*\*

\* Show the Admin->Manage->EditRoom page

\*/

public function roomeditAction ()

{

//Retrieve room

$roomID = $this->\_getParam('id');

$DAOrooms = new DAO\_rooms();

$room = $DAOrooms->retrieveRoomByID($roomID);

if ($room != null) {

//Create new Edit Room Form

$editRoomForm = new Form\_Admin\_manage\_editroom($room->getRoomName(),

$room->getRoomShortName(),

$room->getCapacity());

//Save Edited Room

if ($this->\_request->isPost() && $editRoomForm->isValid($\_POST)) {

$values = $editRoomForm->getValues();

$updateroominfo = $DAOrooms->updateRoomInfo($room,

$values['roomNameEditBox'],

$values['shortNameEditBox'],

$values['capacityEditBox']);

$this->\_redirect('/admin/manage/rooms');

}

$this->view->editRoomForm = $editRoomForm;

} else {

$this->view->errorMessage = "The requested room could not be found.";

}

$this->\_helper->layout->setLayout('/admin/admin\_layout');

}

/\*\*

\* Delete A Room

\*/

public function roomdeleteAction ()

{

$roomID = $this->\_getParam('id');

$DAOrooms = new DAO\_rooms();

$room = $DAOrooms->retrieveRoomByID($roomID);

if ($room != null){

$DAOrooms->deleteRoom($room);

}

$this->\_redirect('/admin/manage/rooms');

}

/\*\*

\* Add A Room

\*/

public function roomaddAction ()

{

//Create add room form

$addRoomForm = new Form\_Admin\_manage\_roomadd();

//Add the room

if ($this->\_request->isPost() && $addRoomForm->isValid($\_POST)) {

$values = $addRoomForm->getValues();

$DAOrooms = new DAO\_rooms();

$addroominfo = $DAOrooms->addRoom($values['roomName'],

$values['roomShortName'],

$values['capacity']);

$this->\_redirect('/admin/manage/roomadd');

}

$this->view->addRoom = $addRoomForm;

$this->\_helper->layout->setLayout('/admin/admin\_layout');

}

/\*\*

\*

\* Admin->Manage->Subjects Page

\*/

public function subjectsAction ()

{

//Retrieve all subjects

$DAOsubjects = new DAO\_subjects();

$subjects = $DAOsubjects->retrieveSubjects();

//Render subjects into table

$tableContent = "";

foreach ($subjects as $subject) {

$tableContent .= $this->renderSubjectsTable($subject);

}

//Create new subjects form

$subjectsForm = new Form\_Admin\_manage\_subjects();

if ($tableContent == ""){

$tableContent = "<tr><td>No subjects available</td></tr>";

}

$this->view->tableContent = $tableContent;

$this->view->subjectsForm = $subjectsForm;

$this->\_helper->layout->setLayout('/admin/admin\_layout');

}

/\*\*

\* Renders a subject into a \_subject-item template

\* @param models\entities\subjects $subjects

\* @return /manage/\_subject-item.phtml

\*/

private function renderSubjectsTable (subjects $subjects)

{

$subjectItem = new Zend\_View();

$subjectItem->setScriptPath($this->view->getScriptPaths());

$subjectItem->subjectID = $subjects->getSubjectId();

$subjectItem->subject = $subjects->getSubject();

return $subjectItem->render('/manage/\_subject-item.phtml');

}

/\*\*

\* Show the Admin->Manage->EditSubject page

\*/

public function subjecteditAction ()

{

//Retrieve Subject

$subjectID = $this->\_getParam('id');

$DAOsubjects = new DAO\_subjects();

$subject = $DAOsubjects->retrieveSubjectByID($subjectID);

if ($subject != null){

//Create New Edit Subject Form

$editSubjectForm = new Form\_Admin\_manage\_editsubject($subject->getSubject());

//Save Edited Subject

if ($this->\_request->isPost() && $editSubjectForm->isValid($\_POST)) {

$values = $editSubjectForm->getValues();

$updatesubjectinfo = $DAOsubjects->updateSubjectInfo($subject,

$values['subjectEditBox']);

$this->\_redirect('/admin/manage/subjects');

}

$this->view->editSubjectForm = $editSubjectForm;

}else{

$this->view->errorMessage = "The requested subject could not be found.";

}

$this->\_helper->layout->setLayout('/admin/admin\_layout');

}

/\*\*

\* Delete A Subject

\*/

public function subjectdeleteAction ()

{

$subjectID = $this->\_getParam('id');

$DAOsubjects = new DAO\_subjects();

$subject = $DAOsubjects->retrieveSubjectByID($subjectID);

if ($subject != null){

$DAOsubjects->deleteSubject($subject);

}

$this->\_redirect('/admin/manage/subjects');

}

/\*\*

\* Add A Subject

\*/

public function subjectaddAction ()

{

//Create Add Subject Form

$addSubjectForm = new Form\_Admin\_manage\_subjectadd();

//Add Subject

if ($this->\_request->isPost() && $addSubjectForm->isValid($\_POST)) {

$values = $addSubjectForm->getValues();

$DAOsubjects = new DAO\_subjects();

//Add the new subject if it is unique

$newSubjectName = trim(strtolower($values['subject']));

$subjectExisits = $DAOsubjects->retrieveSubjectByName($newSubjectName);

if ($subjectExisits == null) {

$addsubjectinfo = $DAOsubjects->addSubject($values['subject']);

$this->\_redirect('/admin/manage/subjectadd');

} else {

$this->view->errorMessage = "Subject already exists.";

}

}

$this->view->addSubject = $addSubjectForm;

$this->\_helper->layout->setLayout('/admin/admin\_layout');

}

/\*\*

\*

\* Admin->Manage->Teachers Page

\*/

public function teachersAction ()

{

//Retrieve all Teachers

$DAOteachers = new DAO\_users();

$teachers = $DAOteachers->retrieveSpecificUsers('teacher');

//Render teachers into table

$tableContent = "";

foreach ($teachers as $teacher) {

$tableContent .= $this->renderTeachersTable($teacher);

}

$this->view->tableContent = $tableContent;

$teachersForm = new Form\_Admin\_manage\_teachers();

$this->view->teachersForm = $teachersForm;

$this->\_helper->layout->setLayout('/admin/admin\_layout');

}

/\*\*

\*

\* Renders a teacher into a \_teacher-item template

\* @param models\entities\users $users

\* @return /manage/\_teacher-item.phtml

\*/

private function renderTeachersTable (users $teachers)

{

$teacherItem = new Zend\_View();

$teacherItem->setScriptPath($this->view->getScriptPaths());

$teacherItem->teacherID = $teachers->getUserId();

$teacherItem->forename = $teachers->getUserForename();

$teacherItem->surname = $teachers->getUserSurname();

$teacherItem->email = $teachers->getUserEmail();

$teacherItem->address = $teachers->getUserAddress();

$teacherItem->telephone = $teachers->getUserTelephone();

$teacherItem->mobile = $teachers->getUserMobile();

return $teacherItem->render('/manage/\_teacher-item.phtml');

}

/\*\*

\*

\* Add or delete Teacher Constraints

\*/

public function teacherconstraintsAction()

{

//Retrieve Teacher

$teacherID = $this->\_getParam('id');

$DAOusers = new DAO\_users();

$teacher = $DAOusers->retrieveUserByID($teacherID);

if ($teacher != null){

//Get Timetable Info

$DAOtimeTableInfo = new DAO\_timetableinfo();

$results = $DAOtimeTableInfo->retrieveTimetableInfo();

foreach ($results as $timetableInfo) {

$academicYear = $timetableInfo->getAcademicYear();

$daysPerWeek = $timetableInfo->getDaysPerWeek();

$lessonsPerDay = $timetableInfo->getLessonsPerDay();

$startTime = $timetableInfo->getStartTime();

$lessonLength = $timetableInfo->getLessonLength();

}

//Calculate Times

$startTimeArray = explode(":", $startTime);

$startHour = $startTimeArray[0];

$startMinutes = $startTimeArray[1];

$lessonLengthArray = explode(":", $lessonLength);

$lessonLengthHour = $lessonLengthArray[0];

$lessonLengthMinutes = $lessonLengthArray[1];

$endHour = $startHour + $lessonLengthHour;

$endMinutes = $startMinutes + $lessonLengthMinutes;

while ($endMinutes > 59) {

$endHour ++;

$endMinutes = $endMinutes - 60;

}

if ($startHour < 10 && strlen($startHour) < 2) {

$startHour = "0" . $startHour;

}

if ($endHour < 10 && strlen($endHour) < 2) {

$endHour = "0" . $endHour;

}

if ($startMinutes == 0 && strlen($startMinutes) < 2) {

$startMinutes = "0" . $startMinutes;

}

if ($endMinutes == 0 && strlen($endMinutes) < 2) {

$endMinutes = "0" . $endMinutes;

}

for ($i = 0; $i < $lessonsPerDay; $i++) {

$fileData = $startHour . "." . $startMinutes . " - " . $endHour . "." . $endMinutes;

$timesArray[] = $fileData;

$startHour = $endHour;

$startMinutes = $endMinutes;

$endHour = $endHour + $lessonLengthHour;

$endMinutes = $endMinutes + $lessonLengthMinutes;

while ($endMinutes > 59) {

$endHour++;

$endMinutes = $endMinutes - 60;

}

if ($startHour < 10 && strlen($startHour) < 2) {

$startHour = "0" . $startHour;

}

if ($endHour < 10 && strlen($endHour) < 2) {

$endHour = "0" . $endHour;

}

if ($startMinutes == 0 && strlen($startMinutes) < 2) {

$startMinutes = "0" . $startMinutes;

}

if ($endMinutes == 0 && strlen($endMinutes) < 2) {

$endMinutes = "0" . $endMinutes;

}

}

//Get working Days in Week based on Days Per Week Value

//submitted in timetabling information

for ($i = 0; $i < $daysPerWeek; $i++) {

switch ($i) {

case 0:

$daysarray[] = "Monday";

break;

case 1:

$daysarray[] = "Tuesday";

break;

case 2:

$daysarray[] = "Wednesday";

break;

case 3:

$daysarray[] = "Thursday";

break;

case 4:

$daysarray[] = "Friday";

break;

case 5:

$daysarray[] = "Saturday";

break;

case 6:

$daysarray[] = "Sunday";

break;

}

}

//Get and display exisiting Constraints

$DAO\_teacherConstraints = new DAO\_teacherconstraints();

$constraints = $DAO\_teacherConstraints->retrieveConstraintsByTeacher($teacherID);

$tableContent = "";

foreach ($constraints as $constraint) {

$tableContent .= $this->renderTeacherConstraints($constraint);

}

$constraintsForm = new Form\_Admin\_manage\_teacherconstraints($daysarray, $timesArray);

//Add new constraint and reload page

if ($this->\_request->isPost() && $constraintsForm->isValid($\_POST)) {

$values = $constraintsForm->getValues();

$dayKey = $values['days'];

$timeKey = $values['times'];

$DAO\_teacherConstraints->addConstraint($teacherID, $daysarray[$dayKey],

$timesArray[$timeKey]);

$this->\_redirect('/admin/manage/teacherconstraints/id/' . $teacherID);

}

$this->view->tableContent = $tableContent;

$this->view->constraintsForm = $constraintsForm;

$this->view->teacherName = $teacher->getUserForename() . ' '

. $teacher->getUserSurname();

} else {

$this->view->errorMessage = "Could not retrieve teacher, please go back and try again.";

}

$this->\_helper->layout->setLayout('/admin/admin\_layout');

}

/\*\*

\*

\* Create constraints table item

\* @param models\entities\teacherconstraints $constraint

\* @return /manage/\_teacher-constraint-item.phtml

\*/

private function renderTeacherConstraints (teacherconstraints $constraint)

{

$teacherConstraintItem = new Zend\_View();

$teacherConstraintItem->setScriptPath($this->view->getScriptPaths());

$teacherConstraintItem->id = $constraint->getTeacher\_constraint\_id();

$teacherConstraintItem->teacherId = $constraint->getTeacher();

$teacherConstraintItem->day = $constraint->getDay();

$teacherConstraintItem->time = $constraint->getTime();

return $teacherConstraintItem->render('/manage/\_teacher-constraint-item.phtml');

}

/\*\*

\*

\* Delete a teacher constraint

\*/

public function teacherconstraintdeleteAction()

{

$constraintID = $this->\_getParam('id');

$teacherID = $this->\_getParam('teacher');

$DAOteacherConstraints = new DAO\_teacherconstraints();

$toDelete = $DAOteacherConstraints->retrieveConstraintsById($constraintID);

if ($toDelete != null){

$DAOteacherConstraints->deleteConstraint($toDelete);

}

$this->\_redirect('/admin/manage/teacherconstraints/id/' . $teacherID);

}

/\*\*

\* Show the Admin->Manage->EditTeacher page

\*/

public function teachereditAction ()

{

try {

//Retrieve Teacher

$teacherID = $this->\_getParam('id');

$DAOteachers = new DAO\_users();

$teacher = $DAOteachers->retrieveUserByID($teacherID);

if ($teacher != null) {

//Create Edit Teacher Form

$editTeacherForm = new Form\_Admin\_manage\_editteacher($teacher->getUserForename(),

$teacher->getUserSurname(),

$teacher->getUserEmail(),

$teacher->getUserAddress(),

$teacher->getUserTelephone(),

$teacher->getUserMobile());

//Save Edited Teacher

if ($this->\_request->isPost() && $editTeacherForm->isValid($\_POST)) {

$values = $editTeacherForm->getValues();

$updateteacherinfo = $DAOteachers->updateUserInfo($teacher,

$values['forenameEditBox'],

$values['surnameEditBox'],

$values['emailEditBox'],

$values['addressEditBox'],

$values['phoneEditBox'],

$values['mobileEditBox']);

$this->\_redirect('/admin/manage/teachers');

}

$this->view->editTeacherForm = $editTeacherForm;

} else {

$this->view->errorMessage = "The requested teacher could not be found.";

}

} catch (Exception $e) {

$this->view->errorMessage = "The requested teacher could not be found.";

}

$this->\_helper->layout->setLayout('/admin/admin\_layout');

}

/\*\*

\* Delete A Teacher

\*/

public function teacherdeleteAction ()

{

$teacherID = $this->\_getParam('id');

$DAOteachers = new DAO\_users();

$teacher = $DAOteachers->retrieveUserByID($teacherID);

if ($teacher != null){

$DAOteachers->deleteUser($teacher);

}

$this->\_redirect('/admin/manage/teachers');

}

/\*\*

\* Add A Teacher

\* username and email must be unique

\*/

public function teacheraddAction ()

{

$addTeacherForm = new Form\_Admin\_manage\_teacheradd();

if ($this->\_request->isPost() && $addTeacherForm->isValid($\_POST)) {

$values = $addTeacherForm->getValues();

$DAOteachers = new DAO\_users();

$addteacherinfo = $DAOteachers->addUser($values['usernameEditBox'],

$values['passwordEditBox'],

$values['forenameEditBox'],

$values['surnameEditBox'],

$values['emailEditBox'],

$values['addressEditBox'],

$values['phoneEditBox'],

$values['mobileEditBox'],

'teacher',

true);

$DAOschoolInfo = new DAO\_schoolinfo();

$schoolInfo = $DAOschoolInfo->retriveSchoolInfo();

$schoolName = $schoolInfo->getSchoolName();

//Create Email message to be sent to new user

$senderEmail = $this->DAOSession->getUser()->getUserEmail();

$bodyText = 'You have been registered at ' . $schoolName . PHP\_EOL;

$bodyText .= 'Use the following details to login:' . PHP\_EOL;

$bodyText .= 'Username: ' . $values['usernameEditBox'] . PHP\_EOL;

$bodyText .= 'Password: ' . $values['passwordEditBox'] . PHP\_EOL;

$bodyText .= '' . PHP\_EOL;

$bodyText .= 'Please log in as soon as possible and change your password.' . PHP\_EOL;

$bodyText .= '' . PHP\_EOL;

$bodyText .= 'http://' . $\_SERVER['SERVER\_NAME'] . '/Login' . PHP\_EOL;

$recipient = $values['forenameEditBox'] . ' ' . $values['surnameEditBox'];

$subject = $schoolName. ' registration';

$mail = new Zend\_Mail();

$mail->setBodyText($bodyText);

$mail->setFrom($senderEmail);

$mail->addTo($values['emailEditBox'], $recipient);

$mail->setSubject($subject);

try {

$mail->send();

} catch (Exception $e) {

$msg = nl2br('Unable to send registration email to ' .

$recipient .

PHP\_EOL .

'Either manually send the registration email or ' .

'delete the teacher and start again.');

$this->view->errorMessage = $msg;

$messageFailure = true;

}

if ($messageFailure == false){

$this->\_redirect('/admin/manage/teacheradd');

}

}

$this->view->addTeacher = $addTeacherForm;

$this->\_helper->layout->setLayout('/admin/admin\_layout');

}

/\*\*

\*

\* Admin->Manage->Students

\*/

public function studentsAction ()

{

//Retrieve all Students

$DAOstudents = new DAO\_users();

$students = $DAOstudents->retrieveSpecificUsers('student');

//Render Student into table

$tableContent = "";

foreach ($students as $student) {

$tableContent .= $this->renderStudentsTable($student);

}

$this->view->tableContent = $tableContent;

$studentsForm = new Form\_Admin\_manage\_students();

$this->view->studentsForm = $studentsForm;

$this->\_helper->layout->setLayout('/admin/admin\_layout');

}

/\*\*

\* Renders a student into a \_student-item template

\* @param models\entities\users $users

\* @return /manage/\_student-item.phtml

\*/

private function renderStudentsTable (users $students)

{

$studentItem = new Zend\_View();

$studentItem->setScriptPath($this->view->getScriptPaths());

$studentItem->studentID = $students->getUserId();

$studentItem->forename = $students->getUserForename();

$studentItem->surname = $students->getUserSurname();

$studentItem->email = $students->getUserEmail();

$studentItem->address = $students->getUserAddress();

$studentItem->telephone = $students->getUserTelephone();

$studentItem->mobile = $students->getUserMobile();

return $studentItem->render('/manage/\_student-item.phtml');

}

/\*\*

\* Show the Admin->Manage->EditStudent page

\*/

public function studenteditAction ()

{

try {

//Retrieve Student

$studentID = $this->\_getParam('id');

$DAOstudents = new DAO\_users();

$student = $DAOstudents->retrieveUserByID($studentID);

if ($student != null) {

//Create Edit Student Form

$editStudentForm = new Form\_Admin\_manage\_editstudent($student->getUserForename(),

$student->getUserSurname(),

$student->getUserEmail(),

$student->getUserAddress(),

$student->getUserTelephone(),

$student->getUserMobile());

//Save Edited Student

if ($this->\_request->isPost() && $editStudentForm->isValid($\_POST)) {

$values = $editStudentForm->getValues();

$updateStudentinfo = $DAOstudents->updateUserInfo($student,

$values['forenameEditBox'],

$values['surnameEditBox'],

$values['emailEditBox'],

$values['addressEditBox'],

$values['phoneEditBox'],

$values['mobileEditBox']);

$this->\_redirect('/admin/manage/students');

}

$this->view->editStudentForm = $editStudentForm;

} else {

$this->view->errorMessage = "The requested student could not be found.";

}

} catch (Exception $e) {

$this->view->errorMessage = "The requested student could not be found.";

}

$this->\_helper->layout->setLayout('/admin/admin\_layout');

}

/\*\*

\* Delete A Student

\*/

public function studentdeleteAction ()

{

$studentID = $this->\_getParam('id');

$DAOstudents = new DAO\_users();

$student = $DAOstudents->retrieveUserByID($studentID);

if ($student != null){

$DAOstudents->deleteUser($student);

}

$this->\_redirect('/admin/manage/students');

}

/\*\*

\* Add A Student

\*/

public function studentaddAction ()

{

//Create Add Student Form

$addStudentForm = new Form\_Admin\_manage\_studentadd();

$messageFailure = false;

//Save button

if ($this->\_request->isPost() && $addStudentForm->isValid($\_POST)) {

$values = $addStudentForm->getValues();

$DAOstudents = new DAO\_users();

$addStudentInfo = $DAOstudents->addUser($values['usernameEditBox'],

$values['passwordEditBox'],

$values['forenameEditBox'],

$values['surnameEditBox'],

$values['emailEditBox'],

$values['addressEditBox'],

$values['phoneEditBox'],

$values['mobileEditBox'],

'student', true);

//Create Mail

$DAOschoolInfo = new DAO\_schoolinfo();

$schoolInfo = $DAOschoolInfo->retriveSchoolInfo();

$schoolName = $schoolInfo->getSchoolName();

$senderEmail = $this->DAOSession->getUser()->getUserEmail();

$bodyText = 'You have been registered at ' . $schoolName . PHP\_EOL;

$bodyText .= 'Use the following details to login:' . PHP\_EOL;

$bodyText .= 'Username: ' . $values['usernameEditBox'] . PHP\_EOL;

$bodyText .= 'Password: ' . $values['passwordEditBox'] . PHP\_EOL;

$bodyText .= '' . PHP\_EOL;

$bodyText .= 'Please log in as soon as possible and change your password.' . PHP\_EOL;

$bodyText .= '' . PHP\_EOL;

$bodyText .= 'http://' . $\_SERVER['SERVER\_NAME'] . '/Login' . PHP\_EOL;

$recipient = $values['forenameEditBox'] . ' ' . $values['surnameEditBox'];

$subject = $schoolName. ' registration';

$mail = new Zend\_Mail();

$mail->setBodyText($bodyText);

$mail->setFrom($senderEmail);

$mail->addTo($values['emailEditBox'], $recipient);

$mail->setSubject($subject);

try {

$mail->send();

} catch (Exception $e) {

$msg = nl2br('Unable to send registration email to ' .

$recipient .

PHP\_EOL .

'Either manually send the registration email or

delete the student and start again.');

$this->view->errorMessage = $msg;

$messageFailure = true;

}

if ($messageFailure == false){

$this->\_redirect('/admin/manage/studentadd');

}

}

$this->view->addStudent = $addStudentForm;

$this->\_helper->layout->setLayout('/admin/admin\_layout');

}

/\*\*

\*

\* This section deals with the instruments a student is

\* associated with and their grade with that instrument.

\*/

**public function** studenteditinstrumentsAction ()

{

//Retrieve Student

$studentID = $this->\_getParam('id');

$DAOstudents = **new** DAO\_users();

$student = $DAOstudents->retrieveUserByID($studentID);

**if** ($student != **null**) {

//Get Instruments used by student

$studentInstruments = $student->getInstruments();

//Get All Instruments

$DAOinstruments = **new** DAO\_instruments();

$allInstruments = $DAOinstruments->retrieveInstruments();

$DAOstudentgrades = **new** DAO\_studentgrades();

//Initialise Table Content

$tableContent = "";

//For every Instrument

**foreach** ($allInstruments **as** $instrument) {

$used = **false**;

$studentgrade = **null**;

//If the student uses the instrument set used to true

**foreach** ($studentInstruments **as** $studentInstrument) {

**if** ($instrument->getInstrumentname() == $studentInstrument->getInstrumentname()) {

$used = **true**;

}

}

//Get Curent Grade of Instrument

$studentgrade = $DAOstudentgrades->retrieveStudentGradeByID($studentID, $instrument->getInstrumentid());

//Create Grade Option Select

**if** ($studentgrade == **null**) {

$grade = 0;

} **else** {

$grade = strval($studentgrade->getGrade());

}

$grades = **array**(0 => 0,

1 => 1,

2 => 2,

3 => 3,

4 => 4,

5 => 5,

6 => 6,

7 => 7,

8 => 8);

$optionSelect = **new** Form\_Admin\_manage\_gradeselect($grades,$grade,$studentID, $instrument->getInstrumentid());

**if** ($used == **true**){

**if** ($this->\_request->isPost() && $optionSelect->isValid($\_POST)) {

$elements = $optionSelect->getElements();

$newGrade = $optionSelect->getValues();

**foreach** ($newGrade **as** $r) {

$newGrade = $r;

}

**if** (! ($newGrade == **null**)) {

**foreach** ($elements **as** $element) {

$idString = $element->id;

}

$pos = strpos($idString, "\_") + 1;

$iId = substr($idString, $pos);

$pos = strpos($idString, "\_");

$sId = substr($idString, 0, $pos);

$DAOstudentGrades = **new** DAO\_studentgrades();

$studentGradeEntity = $DAOstudentGrades->retrieveStudentGradeByID($sId, $iId);

**if** ($studentGradeEntity != **null**){

$addGrade = $DAOstudentGrades->updateStudentGrade($studentGradeEntity,

intval($newGrade));

}

}

}

}

$tableContent .= $this->renderInstrumentsTable($instrument.$used, $studentID,$optionSelect);

}

//Cancel Button

$urlLinkBack = "\"/admin/manage/students" . "\"";

$cancelButton = **new** Form\_Admin\_manage\_cancelbutton($urlLinkBack);

//Render View

$this->view->studentId = $studentID;

$this->view->tableContent = $tableContent;

$this->view->cancelButton = $cancelButton;

}**else** {

$this->view->errorMessage = "The selected student cannot be found.";

}

$this->\_helper->layout->setLayout('/admin/admin\_layout');

}

/\*\*

\*

\* Renders an instrument into a \_instrument-item template

\* @param models\entities\instruments $instrument

\* @param bool $instrumentUsed

\* @param int $studentID

\* @param Form\_Admin\_manage\_gradeselect

\* @return /manage/\_instrument-item.phtml

\*/

private function renderInstrumentsTable (instruments $instrument,

$instrumentUsed, $studentID,

$optionSelect)

{

$instrumentItem = new Zend\_View();

$instrumentItem->setScriptPath($this->view->getScriptPaths());

$instrumentItem->studentID = $studentID;

$instrumentItem->instrumentID = $instrument->getInstrumentid();

$instrumentItem->instrumentName = $instrument->getInstrumentname();

$instrumentItem->optionSelect = $optionSelect;

if ($instrumentUsed == true) {

$instrumentItem->instrumentUsed = 'used';

} else {

$instrumentItem->instrumentUsed = 'not used';

}

return $instrumentItem->render('/manage/\_instrument-item.phtml');

}

/\*\*

\* Adds or deletes an instrument used by a student

\*/

public function studenteditinstusedAction ()

{

$studentID = $this->\_getParam('id');

$instrumentID = $this->\_getParam('instrument');

//Retrieve Student Grade For Instrument

$DAOstudentGrade = new DAO\_studentgrades();

$studentgrade = $DAOstudentGrade->retrieveStudentGradeByID($studentID, $instrumentID);

//If the student currently has no grade then the instrument is not being use

if ($studentgrade == null) {

$addInstrument = $DAOstudentGrade->addStudentGrade($studentID, $instrumentID);

} else {

$removeInstrument = $DAOstudentGrade->removeStudentGrade($studentgrade);

}

$this->\_redirect('/admin/manage/studenteditinstruments/id/' . $studentID);

}

/\*\*

\*

\* Admin->Manage\_Groups

\*/

public function groupsAction ()

{

//Retrieve all Groups

$DAOgroups = new DAO\_groups();

$groups = $DAOgroups->retrieveGroups();

//Render Groups into table

$tableContent = "";

foreach ($groups as $group) {

$tableContent .= $this->renderGroupsTable($group);

}

$this->view->tableContent = $tableContent;

$groupsForm = new Form\_Admin\_manage\_groups();

$this->view->groupsForm = $groupsForm;

$this->\_helper->layout->setLayout('/admin/admin\_layout');

}

/\*\*

\* Renders a student into a \_group-item template

\* @param models\entities\groups $group

\* @return /manage/\_group-item.phtml

\*/

private function renderGroupsTable (groups $group)

{

$groupItem = new Zend\_View();

$groupItem->setScriptPath($this->view->getScriptPaths());

$groupItem->groupID = $group->getGroupId();

$groupItem->groupName = $group->getGroupName();

$groupItem->groupShortName = $group->getGroupShortName();

if ($group->getGroupType() != null){

$groupTypeName = $group->getGroupType()->getGroupTypeName();

} else {

$groupTypeName = null;

}

$groupItem->groupType = $groupTypeName;

//Get Group Users

$users = $group->getUsers();

$participantList = null;

foreach ($users as $user) {

$userForename = $user->getUserForename();

$userSurname = $user->getUserSurname();

if ($participantList == null) {

$participantList = $userForename . ' ' . $userSurname;

} else {

$participantList = $participantList . "<BR>" . $userForename .

' ' . $userSurname;

}

}

$groupItem->participants = $participantList;

return $groupItem->render('/manage/\_group-item.phtml');

}

/\*\*

\*

\* Group Constraint

\*/

public function groupconstraintsAction()

{

//Retrieve Group

$groupID = $this->\_getParam('id');

$DAOgroups = new DAO\_groups();

$group = $DAOgroups->retrieveGroupByID($groupID);

//If Group exists

if ($group != null){

//Get Timetable Info

$DAOtimeTableInfo = new DAO\_timetableinfo();

$results = $DAOtimeTableInfo->retrieveTimetableInfo();

foreach ($results as $timetableInfo) {

$academicYear = $timetableInfo->getAcademicYear();

$daysPerWeek = $timetableInfo->getDaysPerWeek();

$lessonsPerDay = $timetableInfo->getLessonsPerDay();

$startTime = $timetableInfo->getStartTime();

$lessonLength = $timetableInfo->getLessonLength();

}

//Calculate Times

$startTimeArray = explode(":", $startTime);

$startHour = $startTimeArray[0];

$startMinutes = $startTimeArray[1];

$lessonLengthArray = explode(":", $lessonLength);

$lessonLengthHour = $lessonLengthArray[0];

$lessonLengthMinutes = $lessonLengthArray[1];

$endHour = $startHour + $lessonLengthHour;

$endMinutes = $startMinutes + $lessonLengthMinutes;

while ($endMinutes > 59) {

$endHour ++;

$endMinutes = $endMinutes - 60;

}

if ($startHour < 10 && strlen($startHour) < 2) {

$startHour = "0" . $startHour;

}

if ($endHour < 10 && strlen($endHour) < 2) {

$endHour = "0" . $endHour;

}

if ($startMinutes == 0 && strlen($startMinutes) < 2) {

$startMinutes = "0" . $startMinutes;

}

if ($endMinutes == 0 && strlen($endMinutes) < 2) {

$endMinutes = "0" . $endMinutes;

}

for ($i = 0; $i < $lessonsPerDay; $i++) {

$fileData = $startHour . "." . $startMinutes . " - " . $endHour . "." . $endMinutes;

$timesArray[] = $fileData;

$startHour = $endHour;

$startMinutes = $endMinutes;

$endHour = $endHour + $lessonLengthHour;

$endMinutes = $endMinutes + $lessonLengthMinutes;

while ($endMinutes > 59) {

$endHour++;

$endMinutes = $endMinutes - 60;

}

if ($startHour < 10 && strlen($startHour) < 2) {

$startHour = "0" . $startHour;

}

if ($endHour < 10 && strlen($endHour) < 2) {

$endHour = "0" . $endHour;

}

if ($startMinutes == 0 && strlen($startMinutes) < 2) {

$startMinutes = "0" . $startMinutes;

}

if ($endMinutes == 0 && strlen($endMinutes) < 2) {

$endMinutes = "0" . $endMinutes;

}

}

//Get working Days in Week based on Days Per Week Value

//submitted in timetabling information

for ($i = 0; $i < $daysPerWeek; $i++) {

switch ($i) {

case 0:

$daysarray[] = "Monday";

break;

case 1:

$daysarray[] = "Tuesday";

break;

case 2:

$daysarray[] = "Wednesday";

break;

case 3:

$daysarray[] = "Thursday";

break;

case 4:

$daysarray[] = "Friday";

break;

case 5:

$daysarray[] = "Saturday";

break;

case 6:

$daysarray[] = "Sunday";

break;

}

}

//Get and display exisiting Constraints

$DAO\_groupConstraints = new DAO\_groupconstraints();

$constraints = $DAO\_groupConstraints->retrieveConstraintsByGroup($groupID);

$tableContent = "";

foreach ($constraints as $constraint) {

$tableContent .= $this->renderGroupConstraints($constraint);

}

$constraintsForm = new Form\_Admin\_manage\_groupconstraints($daysarray, $timesArray);

//Add new constraint and reload page

if ($this->\_request->isPost() && $constraintsForm->isValid($\_POST)) {

$values = $constraintsForm->getValues();

$dayKey = $values['days'];

$timeKey = $values['times'];

$DAO\_groupConstraints->addConstraint($groupID, $daysarray[$dayKey],

$timesArray[$timeKey]);

$this->\_redirect('/admin/manage/groupconstraints/id/' . $groupID);

}

$this->view->tableContent = $tableContent;

$this->view->constraintsForm = $constraintsForm;

$this->view->groupName = $group->getGroupName();

}

$this->\_helper->layout->setLayout('/admin/admin\_layout');

}

/\*\*

\*

\* Create constraints table item

\* @param models\entities\groupconstraints $constraint

\* @return /manage/\_group-constraint-item.phtml

\*/

private function renderGroupConstraints (groupconstraints $constraint)

{

$groupConstraintItem = new Zend\_View();

$groupConstraintItem->setScriptPath($this->view->getScriptPaths());

$groupConstraintItem->id = $constraint->getGroup\_constraint\_id();

$groupConstraintItem->groupId = $constraint->getGroup();

$groupConstraintItem->day = $constraint->getDay();

$groupConstraintItem->time = $constraint->getTime();

return $groupConstraintItem->render('/manage/\_group-constraint-item.phtml');

}

/\*\*

\*

\* Delete a Group Constraint

\*/

public function groupconstraintdeleteAction()

{

$constraintID = $this->\_getParam('id');

$groupID = $this->\_getParam('group');

$DAOgroupConstraints = new DAO\_groupconstraints();

$toDelete = $DAOgroupConstraints->retrieveConstraintsById($constraintID);

if ($toDelete != null){

$DAOgroupConstraints->deleteConstraint($toDelete);

}

$this->\_redirect('/admin/manage/groupconstraints/id/' . $groupID);

}

/\*\*

\* Show the Admin->Manage->EditGroup page

\*/

public function groupeditAction ()

{

try {

//Retrieve Grou

$groupID = $this->\_getParam('id');

$DAOgroups = new DAO\_groups();

$group = $DAOgroups->retrieveGroupByID($groupID);

//If group exists

if ($group != null) {

//Retrieve Group Types

$DAOgroupTypes = new DAO\_grouptype();

$groupTypes = $DAOgroupTypes->retrieveGroupTypes();

//Retrieve Users of group

$students = $group->getUsers();

//Retrieve Id of Group Type

if ($group->getGroupType() != null){

$groupTypeId = $group->getGroupType()->getGroupTypeId();

} else{

$groupTypeId = null;

}

//Create Group Edit Form

$groupEdit = new Form\_Admin\_manage\_groupedit($group->getGroupName(),

$group->getGroupShortName(),

$groupTypes,

$groupTypeId,

$students,

$groupID);

//Save Edited Group

if ($this->\_request->isPost() && $groupEdit->isValid($\_POST)) {

$values = $groupEdit->getValues();

$groupType = $DAOgroupTypes->retrieveGroupTypeByID($values['groupType']);

$newGroup = $DAOgroups->updateGroup($group, $values['GroupName'],

$values['groupShortName'

$groupType);

$this->\_redirect('/admin/manage/groups');

}

$this->view->groupID = $groupID;

$this->view->groupEdit = $groupEdit;

$urlLinkBack = '"/admin/manage/groups"';

$cancelButton = new Form\_Admin\_manage\_cancelbutton($urlLinkBack);

$this->view->cancelButton = $cancelButton;

}else {

$this->view->errorMessage = "The requested group could not be found.";

}

} catch (Exception $e) {

$this->view->errorMessage = "The requested group could not be found.";

}

$this->\_helper->layout->setLayout('/admin/admin\_layout');

}

/\*\*

\* Add new grouptype and reload edit page

\*/

public function grouptypeaddAction ()

{

//Add new group type if it is unique

$newGroupType = $this->\_getParam('id');

$groupID = $this->\_getParam('groupid');

$DAOgroupType = new DAO\_grouptype();

$typeExists = $DAOgroupType->checkTypeExists(trim(strtolower($newGroupType)));

if ($typeExists == null) {

$addGroupType = $DAOgroupType->addGroupType($newGroupType);

} else {

$this->view->errorMessage = "Type already exists.";

}

$url = '/admin/manage/grouptype/id/' . trim($groupID);

$this->\_redirect($url);

}

/\*\*

\* Edit Students in group

\*/

public function groupeditstudentsAction ()

{

//Retrieve Group

$groupID = $this->\_getParam('id');

$DAOgroups = new DAO\_groups();

$group = $DAOgroups->retrieveGroupByID($groupID);

//If group exists

if ($group != null) {

//Get Student in Group

$StudentsInGroup = $group->getUsers();

//Get all Students

$DAOstudents = new DAO\_users();

$allStudents = $DAOstudents->retrieveSpecificUsers('student');

//Render Students list

$tableContent = "";

foreach ($allStudents as $student) {

$tableContent .= $this->renderGroupStudentsTable($student,

$StudentsInGroup,

$groupID);

}

$this->view->studentsTable = $tableContent;

$urlLinkBack = "\"/admin/manage/groupedit/id/" . trim($groupID) . "\"";

$cancelButton = new Form\_Admin\_manage\_cancelbutton($urlLinkBack);

$this->view->cancelButton = $cancelButton;

} else {

$this->view->errorMessage = "Invalid Id supplied.";

}

$this->\_helper->layout->setLayout('/admin/admin\_layout');

}

/\*\*

\* Renders a student into a \_student-item template

\* @param models\entities\users $student

\* @param models\entities\users $GroupStudents

\* @param int $groupID

\* @return /manage/\_group-student-item.phtml

\*/

private function renderGroupStudentsTable (users $student, $GroupStudents, $groupID)

{

$studentItem = new Zend\_View();

$studentItem->setScriptPath($this->view->getScriptPaths());

$studentItem->studentID = $student->getUserId();

$studentItem->forename = $student->getUserForename();

$studentItem->surname = $student->getUserSurname();

$studentItem->email = $student->getUserEmail();

$studentItem->address = $student->getUserAddress();

$studentItem->groupId = $groupID;

$studentInGroup = false;

foreach ($GroupStudents as $gs) {

if ($student->getUserId() == $gs->getUserId()){

$studentInGroup = true;

}

}

if ($studentInGroup == true) {

$studentItem->studentInGroup = 'Yes';

} else {

$studentItem->studentInGroup = 'No';

}

return $studentItem->render('/manage/\_group-student-item.phtml');

}

/\*\*

\*

\* Add or remove a student from a group

\*/

public function studentingroupAction()

{

//Get Params

$studentID = $this->\_getParam('id');

$groupID = $this->\_getParam('groupid');

//Retrieve Student by Id

$DAOstudents = new DAO\_users();

$student = $DAOstudents->retrieveUserByID($studentID);

//Retrieve Group By Id

//Get Students in Group

$DAOgroups = new DAO\_groups();

$group = $DAOgroups->retrieveGroupByID($groupID);

$StudentsInGroup = $group->getUsers();

//Determine if student is in group

$studentInGroup = false;

foreach ($StudentsInGroup as $sg) {

if ($sg->getUserId() == $student->getUserId()) {

$studentInGroup = true;

}

}

//If student is in group remove them and vice versa

if ($studentInGroup == false) {

$newGroup[] = $student;

$DAOgroups->updateUsersOfGroup($group, $newGroup);

}else {

$newGroup[] = $student;

$DAOgroups->removeUsersOfGroup($group, $newGroup);

}

$this->\_redirect('/admin/manage/groupeditstudents/id/' . $groupID);

}

/\*\*

\*

\* Display Group Types page

\*/

public function grouptypeAction()

{

//Retrieve Group Types

$groupID = $this->\_getParam('id');

$DAOgroupTypes = new DAO\_grouptype();

$groupTypes = $DAOgroupTypes->retrieveGroupTypes();

//Render Group Types Into a table

$tableContent = "";

foreach ($groupTypes as $groupType){

$tableContent .= $this->renderGroupTypeTable($groupType, $groupID);

}

//Cancel Button

$urlLinkBack = "\"/admin/manage/groupedit/id/" . $groupID . "\"";

$cancelButton = new Form\_Admin\_manage\_cancelbutton($urlLinkBack);

$this->view->groupID = $groupID;

$this->view->cancelButton = $cancelButton;

$this->view->tableContent = $tableContent;

$this->\_helper->layout->setLayout('/admin/admin\_layout');

}

/\*\*

\* Render a Group Type into a Group Type Item

\*

\* @param $groupTypeEntity

\* @param $groupID

\* @return /manage/\_grouptype-item.phtml

\*/

private function renderGroupTypeTable (grouptype $groupTypeEntity,$groupID)

{

$groupTypeItem = new Zend\_View();

$groupTypeItem->setScriptPath($this->view->getScriptPaths());

$groupTypeItem->groupId = $groupID;

$groupTypeItem->groupTypeId = $groupTypeEntity->getGroupTypeId();

$groupTypeItem->groupTypeName = $groupTypeEntity->getGroupTypeName();

return $groupTypeItem->render('/manage/\_grouptype-item.phtml');

}

/\*\*

\*

\* Delete a Group Type and reload page

\*/

public function grouptpyedeleteAction()

{

$groupTypeID = $this->\_getParam('type');

$groupID = $this->\_getParam('group');

$DAOgroupTypes = new DAO\_grouptype();

$groupType = $DAOgroupTypes->retrieveGroupTypeByID($groupTypeID);

if ($groupType != null){

$DAOgroupTypes->deleteGroupType($groupType);

}

$this->\_redirect('/admin/manage/grouptype/id/' . $groupID);

}

/\*\*

\*

\* Update a Group Type

\*/

public function grouptypeeditAction()

{

$groupTypeID = $this->\_getParam('type');

$groupID = $this->\_getParam('group');

$DAOgrouptype = new DAO\_grouptype;

$groupType = $DAOgrouptype->retrieveGroupTypeByID($groupTypeID);

$groupTypeForm = new Form\_Admin\_manage\_grouptypeedit($groupType->getGroupTypeName(),

$groupID);

if ($this->\_request->isPost() && $groupTypeForm->isvalid($\_POST))

{

$values = $groupTypeForm->getValues();

$DAOgrouptype->updateGroupType($groupType, $values['groupType']);

$this->\_redirect('/admin/manage/grouptype/id/' . $groupID);

}

$this->view->groupTypeForm = $groupTypeForm;

$this->\_helper->layout->setLayout('/admin/admin\_layout');

}

/\*\*

\*

\* Add a group

\*/

public function groupaddAction()

{

$DAOgrouptype = new DAO\_grouptype;

$groupTypes = $DAOgrouptype->retrieveGroupTypes();

$groupAddForm = new Form\_Admin\_manage\_groupadd($groupTypes);

if ($this->\_request->isPost() && $groupAddForm->isValid($\_POST)) {

$values = $groupAddForm->getValues();

$groupTypeId = $values['groupType'];

if ($groupTypeId != null){

$groupTypeEntity = $DAOgrouptype->retrieveGroupTypeByID($groupTypeId);

}else{

$groupTypeEntity = null;

}

$DAOgroups = new DAO\_groups();

$DAOgroups->AddGroup($values['GroupName'],

$values['groupShortName'],

$groupTypeEntity);

$this->\_redirect('/admin/manage/groupadd');

}

$this->view->groupAddForm = $groupAddForm;

$this->\_helper->layout->setLayout('/admin/admin\_layout');

}

/\*\*

\*

\* Delete a group

\*/

public function groupdeleteAction()

{

//Get group Id

$groupID = $this->\_getParam('id');

$DAOgroups = new DAO\_groups();

$groupToDelete = $DAOgroups->retrieveGroupByID($groupID);

if ($groupToDelete != null){

$DAOgroups->deleteGroup($groupToDelete);

}

$this->\_redirect('/admin/manage/groups');

}

/\*\*

\*

\* Admin->Manage->Lesson Page

\* Display all available lessons

\*/

public function lessonsAction ()

{

if ($this->\_request->isGet()) {

$DAOlessons = new DAO\_lessons();

$lessons = $DAOlessons->retrieveLessons();

$tableContent = "";

foreach ($lessons as $lesson) {

$tableContent .= $this->renderLessonsTable($lesson);

}

$lessonsForm = new Form\_admin\_manage\_lessons();

$this->view->addButton = $lessonsForm;

$this->view->tableContent = $tableContent;

$this->\_helper->layout->setLayout('/admin/admin\_layout');

}

}

/\*\*

\* Render a Lessons into a lesson Item

\* @param models\entities\lessons $lessonEntity

\* @return /manage/\_lesson-item.phtml

\*/

private function renderLessonsTable (lessons $lessonEntity)

{

$lessonItem = new Zend\_View();

$lessonItem->setScriptPath($this->view->getScriptPaths());

$lessonItem->lessonId = $lessonEntity->getLessonid();

$lessonItem->subject = $lessonEntity->getSubjects()->getSubject();

$lessonItem->group = $lessonEntity->getGroup()->getGroupName();

$lessonItem->teacher = $lessonEntity->getTeacher()->getUserForename() .

' ' . $lessonEntity->getTeacher()->getUserSurname();

$lessonItem->room = $lessonEntity->getRoom()->getRoomName();

$lessonItem->lessonLength = $lessonEntity->getLessonlength();

$lessonItem->lessonCount = $lessonEntity->getLessoncount();

return $lessonItem->render('/manage/\_lesson-item.phtml');

}

/\*\*

\* Delete A Lesson

\*/

public function lessondeleteAction ()

{

//Get Lesson Id

$lessonID = $this->\_getParam('id');

$DAOlessons = new DAO\_lessons();

$lesson = $DAOlessons->retrieveLessonByID($lessonID);

if ($lesson !=null){

$DAOlessons->deleteLesson($lesson);

}

$this->\_redirect('/admin/manage/lessons');

}

/\*\*

\*

\* Edit a Lesson

\*/

public function lessoneditAction ()

{

//Get lesson Id

$lessonID = $this->\_getParam('id');

//Retrieve Lesson Details

$DAOlessons = new DAO\_lessons();

$lesson = $DAOlessons->retrieveLessonByID($lessonID);

//If the lesson exists

if ($lesson != null) {

//Get subject Id for lesson to edit

//Get List of subjects

$subjectId = $lesson->getSubjects()->getSubjectid();

$DAOsubjects = new DAO\_subjects();

$subjects = $DAOsubjects->retrieveSubjects();

$subjectsArray = array();

foreach ($subjects as $subject) {

$id = $subject->getSubjectid();

$sub = $subject->getSubject();

$subjectsArray[$id] = $sub;

}

//Get group id for lesson to edit

//Get list of groups

$groupId = $lesson->getGroup()->getGroupId();

$DAOgroups = new DAO\_groups();

$groups = $DAOgroups->retrieveGroups();

$groupsArray = array();

foreach ($groups as $group) {

$id = $group->getGroupId();

$gro = $group->getGroupName();

$groupsArray[$id] = $gro;

}

//Get teacher id for lesson to edit

//Get List of teachers

$teacherId = $lesson->getTeacher()->getUserId();

$DAOteachers = new DAO\_users();

$teachers = $DAOteachers->retrieveSpecificUsers('teacher');

$teachersArray = array();

foreach ($teachers as $teacher) {

$id = $teacher->getUserId();

$teach = $teacher->getUserForename() . ' ' . $teacher->getUserSurname();

$teachersArray[$id] = $teach;

}

//Get room id for the lesson to edit

//Get list of rooms

$roomId = $lesson->getRoom()->getRoomId();

$DAOrooms = new DAO\_rooms();

$rooms = $DAOrooms->retrieveRooms();

$roomsArray = array();

foreach ($rooms as $room) {

$id = $room->getRoomId();

$r = $room->getRoomName();

$roomsArray[$id] = $r;

}

//Get lesson length for the lesson to edit

//Get lesson Count for the lesson to edit

$lessonLength = $lesson->getLessonlength();

$lessonCount = $lesson->getLessonCount();

//Create Edit Lesson Form

$lessonEditForm = new Form\_Admin\_manage\_lessonedit($subjectsArray,

$subjectId,

$groupsArray,

$groupId,

$teachersArray,

$teacherId,

$roomsArray,

$roomId,

$lessonLength,

$lessonCount);

//Update Lesson

if ($this->\_request->IsPost() && $lessonEditForm->isValid($\_POST)) {

$values = $lessonEditForm->getValues();

$editedSubject = $values['subject'];

$editedSubjectEntity = $DAOsubjects->retrieveSubjectByID( $editedSubject);

$DAOlessons->updateLessonSubject($lesson, $editedSubjectEntity);

$editedGroup = $values['group'];

$editedGroupEntity = $DAOgroups->retrieveGroupByID($editedGroup);

$DAOlessons->updateLessonGroup($lesson, $editedGroupEntity);

$editedTeacher = $values['teacher'];

$editedTeacherEntity = $DAOteachers->retrieveUserByID($editedTeacher);

$DAOlessons->updateLessonTeacher($lesson, $editedTeacherEntity);

$editedRoom = $values['room'];

$editedRoomEntity = $DAOrooms->retrieveRoomByID($editedRoom);

$DAOlessons->updateLessonRoom($lesson, $editedRoomEntity);

$editedLessonLength = $values['lessonLength'];

$editedLessonCount = $values['lessonCount'];

$DAOlessons->updateLessonLengthCount($lesson, $editedLessonLength,

$editedLessonCount);

$this->\_redirect('/admin/manage/lessons');

}

$this->view->lessonEditForm = $lessonEditForm;

}else {

$this->view->errorMessage = "The requested lesson could not be found.";

}

$this->\_helper->layout->setLayout('/admin/admin\_layout');

}

/\*\*

\*

\* Add a New Lesson

\*/

public function lessonaddAction ()

{

//Get List of subjects, groups, teachers, rooms

$DAOsubjects = new DAO\_subjects();

$subjects = $DAOsubjects->retrieveSubjects();

$subjectsArray = array();

foreach ($subjects as $subject) {

$id = $subject->getSubjectid();

$sub = $subject->getSubject();

$subjectsArray[$id] = $sub;

}

$DAOgroups = new DAO\_groups();

$groups = $DAOgroups->retrieveGroups();

$groupsArray = array();

foreach ($groups as $group) {

$id = $group->getGroupId();

$gro = $group->getGroupName();

$groupsArray[$id] = $gro;

}

$DAOteachers = new DAO\_users();

$teachers = $DAOteachers->retrieveSpecificUsers('teacher');

$teachersArray = array();

foreach ($teachers as $teacher) {

$id = $teacher->getUserId();

$teach = $teacher->getUserForename() . ' ' .

$teacher->getUserSurname();

$teachersArray[$id] = $teach;

}

$DAOrooms = new DAO\_rooms();

$rooms = $DAOrooms->retrieveRooms();

$roomsArray = array();

foreach ($rooms as $room) {

$id = $room->getRoomId();

$r = $room->getRoomName();

$roomsArray[$id] = $r;

}

//Create Lesson Add Form

$lessonAddForm = new Form\_Admin\_manage\_lessonadd($subjectsArray,

$groupsArray,

$teachersArray,

$roomsArray);

If ($this->\_request->isPost() && $lessonAddForm->isvalid($\_POST)) {

//Get New lesson Data

$values = $lessonAddForm->getValues();

$newSubject = $DAOsubjects->retrieveSubjectByID($values['subject']);

$newGroup = $DAOgroups->retrieveGroupByID($values['group']);

$newTeacher = $DAOteachers->retrieveUserByID($values['teacher']);

$newRoom = $DAOrooms->retrieveRoomByID($values['room']);

//Add New Lesson

$DAOlessons = new DAO\_lessons();

$DAOlessons->addLesson($newSubject, $newGroup,

$newTeacher, $newRoom,

$values['lessonLength'],

$values['lessonCount']);

$this->\_redirect('/admin/manage/lessons');

}

$this->view->lessonForm = $lessonAddForm;

$this->\_helper->layout->setLayout('/admin/admin\_layout');

}

}

###### myaccountController

<?php

/\*\*

\*

\* Admin myaccountController

\* @author Aaron Redmond

\* @version 1.0

\*/

require\_once APPLICATION\_PATH . '/modules/AbstractController.php';

use models\entities\Session;

use models\entities\users;

class Admin\_myaccountController extends AbstractController

{

/\*\*

\*Check to see if the user is logged in

\*This function is applied to all pages

\*/

public function init ()

{

if (! ($this->DAOSession->isLoggedIn())) {

$this->\_redirect('/Login');

}

if (!($this->DAOSession->getUser()->getUserType() == 'admin')){

$this->\_redirect('/Login');

}

}

/\*\*

\* Show the reset password page

\* Allows the user to change their password

\*/

public function passwordAction ()

{

$passwordForm = new Form\_Admin\_myaccount\_password();

if ($this->\_request->isPost() && $passwordForm->isValid($\_POST)) {

$values = $passwordForm->getValues();

$oldPassword = $values['oldPassword'];

$newPassword = $values['newPassword'];

$verifyNewPassword = $values['verifyNewPassword'];

$user = $this->DAOSession->getUser();

$existingPassword = $user->getUserPassword();

//if the password entered in the old password

//field match the password in the database

if (md5($oldPassword) == $existingPassword) {

//if new password matches verifynew password

//change the password

if ($newPassword == $verifyNewPassword) {

$DAOusers = new DAO\_users();

$DAOusers->updateUserPassword($user, md5($newPassword));

$this->view->errorMessage = "Password successfully changed";

}else {

$this->view->errorMessage = "New password and verify new password must match";

}

} else {

$this->view->errorMessage = "Incorrect password entered.";

}

}

$this->view->passwordForm = $passwordForm;

$this->\_helper->layout->setLayout('/admin/admin\_layout');

}

/\*\*

\*

\* Show the change username page

\* Allows the user to change their username

\* to a new unique username

\*/

public function usernameAction()

{

$usernameForm = new Form\_Admin\_myaccount\_username();

$user = $this->DAOSession->getUser();

if ($this->\_request->isPost() && $usernameForm->isValid($\_POST)) {

$values = $usernameForm->getValues();

$newUsername = $values['newUsername'];

$DAOusers = new DAO\_users();

$usernameexists = $DAOusers->retrieveUserByUsername($newUsername);

if ($usernameexists == null) {

$DAOusers->updateUserName($user, $newUsername);

$this->view->errorMessage = "Username successfully changed";

}else {

$this->view->errorMessage = "Username already exists";

}

}

$this->view->usernameForm = $usernameForm;

$this->\_helper->layout->setLayout('/admin/admin\_layout');

}

}

###### setupController

<?php

/\*\*

\*

\* Admin setupController

\* @author Aaron Redmond

\* @version 1.0

\*/

require\_once APPLICATION\_PATH . '/modules/AbstractController.php';

use models\entities\instruments;

use models\entities\Session;

class Admin\_setupController extends AbstractController

{

/\*\*

\*Check to see if the user is logged in

\*This function is applied to all pages

\*Check to see if the user is an admin user

\*/

public function init ()

{

if (! ($this->DAOSession->isLoggedIn())) {

$this->\_redirect('/Login');

}

if (!($this->DAOSession->getUser()->getUserType() == 'admin')){

$this->\_redirect('/Login');

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Instruments

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*\*

\*

\* The default instruments page

\*/

public function instrumentsAction ()

{

$DAOinstruments = new DAO\_instruments();

$instruments = $DAOinstruments->retrieveInstruments();

$tableContent = "";

foreach ($instruments as $instrument){

$tableContent .= $this->renderInstrumentsTable($instrument);

}

$addButton = new Form\_Admin\_setup\_instrumentaddbutton();

$this->view->addButton = $addButton;

$this->view->tableContent = $tableContent;

$this->\_helper->layout->setLayout('/admin/admin\_layout');

}

/\*\*

\* Renders an instrument into an \_instrument-item template

\* @param instruments $instrument

\* @return /setup/\_instrument-item.phtml

\*/

private function renderInstrumentsTable (instruments $instrument)

{

$tableItem = new Zend\_View();

$tableItem->setScriptPath($this->view->getScriptPaths());

$tableItem->instrumentID = $instrument->getInstrumentid();

$tableItem->instrumentName = $instrument->getInstrumentname();

$used = $instrument->getInstrumentused();

if ($used == 1){

$instrumentUsed = "used";

} else {

$instrumentUsed = "not used";

}

$tableItem->instrumentUsed = $instrumentUsed;

return $tableItem->render('/setup/\_instrument-item.phtml');

}

/\*\*

\* Updates Instrument used values

\*/

public function instrumentusedAction ()

{

$instrumentID = $this->\_getParam('id');

$DAOinstruments = new DAO\_instruments();

$instrument = $DAOinstruments->retrieveInstrumentByID($instrumentID);

$used = $instrument->getInstrumentused();

if ($used == 1) {

$used = 0;

} else {

$used = 1;

}

$DAOinstruments->updateInstrumentUsed($instrument, $used);

$this->\_redirect('/admin/setup/instruments');

}

/\*\*

\*

\* Delete an Instrument

\*/

public function instrumentdeleteAction()

{

$instrumentID = $this->\_getParam('id');

$DAOinstruments = new DAO\_instruments();

$instrument = $DAOinstruments->retrieveInstrumentByID($instrumentID);

$DAOinstruments->deleteInstrument($instrument);

$this->\_redirect('/admin/setup/instruments');

}

/\*\*

\*

\*Edit Instrument

\*/

public function instrumenteditAction()

{

$instrumentID = $this->\_getParam('id');

$DAOinstruments = new DAO\_instruments();

$instrument = $DAOinstruments->retrieveInstrumentByID($instrumentID);

$instrumentEditForm = new Form\_Admin\_setup\_instrumentedit($instrument->getInstrumentname());

if ($this->\_request->isPost() && $instrumentEditForm->isValid($\_POST)){

$values = $instrumentEditForm->getValues();

$newInstrumentName = $values['instrument'];

$newInstrumentName = trim(strtolower($newInstrumentName));

$instrumentExisits = $DAOinstruments->checkInstrumentExists($newInstrumentName);

if ($instrumentExisits == null){

$DAOinstruments->updateInstrumentName($instrument, $values['instrument']);

$this->\_redirect('/admin/setup/instruments');

} else {

$this->view->errorMessage = "Instrument already exists.";

}

}

$this->view->instrumentEditForm = $instrumentEditForm;

$this->\_helper->layout->setLayout('/admin/admin\_layout');

}

/\*\*

\*

\* Add Instrument providing it does not already exist

\*/

public function instrumentaddAction()

{

$instrumentAddForm = new Form\_Admin\_setup\_instrumentadd();

$DAOinstruments = new DAO\_instruments();

if ($this->\_request->isPost() && $instrumentAddForm->isValid($\_POST)){

$values = $instrumentAddForm->getValues();

$newInstrumentName = $values['instrument'];

$newInstrumentName = trim(strtolower($newInstrumentName));

$instrumentExisits = $DAOinstruments->checkInstrumentExists($newInstrumentName);

if ($instrumentExisits == null){

$DAOinstruments->addInstrument($newInstrumentName);

$this->\_redirect('/admin/setup/instruments');

} else {

$this->view->errorMessage = "Instrument already exists.";

}

}

$this->view->instrumentAddForm = $instrumentAddForm;

$this->\_helper->layout->setLayout('/admin/admin\_layout');

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//News

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*\*

\*

\* Render existing News Articles

\* Save New News Articles

\*/

public function newsAction()

{

$DAOnews = new DAO\_news();

$newsArticles = $DAOnews->retrieveNewsItems();

//Render news items which each have an edit delete option

$tableContent = "";

foreach ($newsArticles as $newsArticle){

$tableContent .= $this->renderNewsTable($newsArticle);

}

//Text area form with publish button

$newsForm = new Form\_Admin\_setup\_news();

if ($this->\_request->isPost() && $newsForm->isValid($\_POST)) {

$values = $newsForm->getValues();

$currentDateTime = date("d/m/y h:i:s a", time());

$DAOnews->addNewsItem($values['articleTitle'], $values['article'], $currentDateTime);

$this->\_redirect('/admin/setup/news');

}

$this->view->newsForm = $newsForm;

$this->view->newsItems = $tableContent;

$this->\_helper->layout->setLayout('/admin/admin\_layout');

}

/\*\*

\*

\* Render a news item into a \_news-item table

\* @param models/entities/news $newsArticle

\* @return /setup/\_news-item.phtml

\*/

private function renderNewsTable($newsArticle)

{

$tableItem = new Zend\_View();

$tableItem->setScriptPath($this->view->getScriptPaths());

$tableItem->newsId = $newsArticle->getNewsid();

$tableItem->title = $newsArticle->getTitle();

$tableItem->article = $newsArticle->getArticle();

$tableItem->publishedDate = $newsArticle->getPublishedDate();

return $tableItem->render('/setup/\_news-item.phtml');

}

/\*\*

\*

\* Delete a news Item

\*/

public function newsarticledeleteAction()

{

$newsId = $this->\_getParam('id');

$DAOnews = new DAO\_news();

$newsItem = $DAOnews->retrieveNewsItemByID($newsId);

$DAOnews->deleteNewsItem($newsItem);

$this->\_redirect('/admin/setup/news');

}

/\*\*

\*

\* Edit a News Item

\*/

public function newsitemeditAction() {

$newsItemID = $this->\_getParam('id');

$DAOnews = new DAO\_news();

$newsItem = $DAOnews->retrieveNewsItemByID($newsItemID);

$newsItemEditForm = new Form\_Admin\_setup\_newsItemEdit($newsItem->getTitle(), $newsItem->getArticle());

if ($this->\_request->isPost() && $newsItemEditForm->isValid($\_POST)){

$values = $newsItemEditForm->getValues();

$newTitle = $values['articleTitle'];

$newArticle = $values['article'];

$currentDateTime = date("d/m/y h:i:s a", time());

$DAOnews->updateNewsItem($newsItem, $newTitle, $newArticle, $currentDateTime);

$this->\_redirect('/admin/setup/news');

}

$this->view->newsItemEditForm = $newsItemEditForm;

$this->\_helper->layout->setLayout('/admin/admin\_layout');

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Personalisation

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*\*

\*

\* Personalisation Page

\* Create List of images from images in the public folder "msm\_images"

\*

\* Allow user to upload images

\*/

public function personalisationAction()

{

//Get jpg|gif|bmp Image Files stored

//in msm\_images directory

$rootPath = $\_SERVER['DOCUMENT\_ROOT'];

$imagesFolder = '/msm\_images';

$handle = null;

$listContent= "";

if (file\_exists($rootPath . $imagesFolder)) {

if ($handle = opendir($rootPath . $imagesFolder)) {

while (false !== ($file = readdir($handle))) {

$ext = strchr($file,'.');

switch (strtolower($ext)){

case '.jpg':

$listContent .= $this->renderImagesList($imagesFolder, $file);

break;

case '.gif':

$listContent .= $this->renderImagesList($imagesFolder, $file);

break;

case '.bmp';

$listContent .= $this->renderImagesList($imagesFolder, $file);

break;

}

}

closedir($handle);

}

}

//Allow user to upload file

$path = $rootPath . '/' . $imagesFolder;

$uploadForm = new Form\_Admin\_setup\_fileupload($path);

if ($this->\_request->isPost() && $uploadForm->isvalid($\_POST)){

$values = $uploadForm->getValues();

$uploadedImage = $values['file'];

$this->\_redirect('/admin/setup/personalisation');

}

$this->view->listContent = $listContent;

$this->view->uploadForm = $uploadForm;

$this->\_helper->layout->setLayout('/admin/personalisation\_layout');

}

/\*\*

\*

\* Create image ListItem

\* @param string $folderName

\* @param string $fileName

\* @return /setup/\_image-list-item.phtml

\*/

private function renderImagesList($folderName, $fileName)

{

$tableItem = new Zend\_View();

$tableItem->setScriptPath($this->view->getScriptPaths());

$imageItem = $folderName . '/' . $fileName;

$tableItem->imageItem = $imageItem;

$tableItem->fileName = $fileName;

return $tableItem->render('/setup/\_image-list-item.phtml');

}

/\*\*

\*

\* Sets the choosen image as the new banner

\*/

public function useimageAction()

{

$imageToUse = $this->\_getParam('id');

if (file\_exists('msm\_images/' . $imageToUse)){

$DAOschoolInfo = new DAO\_schoolinfo();

$school = $DAOschoolInfo->retriveSchoolInfo();

$DAOschoolInfo->changeImage($school, $imageToUse);

}

$this->\_redirect('/admin/setup/personalisation');

}

}

###### timetablingController

<?php

/\*\*

\* Admin timetablingController

\*

\* @author Aaron Redmond

\* @version 1.0

\*

\*/

require\_once APPLICATION\_PATH . '/modules/AbstractController.php';

use models\entities\groups;

use models\entities\Session;

use models\entities\timetableinfo;

use models\entities\schoolinfo;

use models\entities\users;

use models\entities\rooms;

class Admin\_timetablingController extends AbstractController

{

/\*\*

\*Check to see if the user is logged in

\*This function is applied to all pages

\*/

public function init ()

{

if (! ($this->DAOSession->isLoggedIn())) {

$this->\_redirect('/Login');

}

if (!($this->DAOSession->getUser()->getUserType() == 'admin')){

$this->\_redirect('/Login');

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Timetabling Information

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*\*

\* Timetabling Information

\* Allow user to view/change the timetable information

\* Also delete all time constraints if times are changed

\*/

public function informationAction ()

{

$DAOtimetableInfo = new DAO\_timetableinfo;

$results = $DAOtimetableInfo->retrieveTimetableInfo();

foreach ($results as $timetableInfo) {

$academicYear = $timetableInfo->getAcademicYear();

$daysPerWeek = $timetableInfo->getDaysPerWeek();

$lessonsPerDay = $timetableInfo->getLessonsPerDay();

$startTime = $timetableInfo->getStartTime();

$lessonLength = $timetableInfo->getLessonLength();

}

$daysPerWeekArray = array(1 => 1, 2 => 2, 3 => 3, 4 => 4, 5 => 5, 6 => 6, 7 => 7);

$lessonsPerDayArray = array();

for ($i = 1; $i < 25; $i++) {

$lessonsPerDayArray[$i] = $i;

}

$timetableInfoForm = new Form\_Admin\_timetabling\_information($academicYear,

$daysPerWeekArray,

$daysPerWeek,

$lessonsPerDayArray,

$lessonsPerDay,

$startTime,

$lessonLength);

if ($this->\_request->isPost() && $timetableInfoForm->isValid($\_POST)) {

$values = $timetableInfoForm->getValues();

//put Start Time in format we want HH:MM

$startTimeArray = preg\_split("/:/", $values['startTime']);

while (sizeof($startTimeArray) < 2) {

$startTimeArray[] = '00';

}

if (strlen($startTimeArray[0]) == 1){

$startTimeArray[0] = '0' . $startTimeArray[0];

}

if (strlen($startTimeArray[1]) == 1){

$startTimeArray[1] = '0' . $startTimeArray[1];

}

$startTime = $startTimeArray[0] . ':' . $startTimeArray[1];

//put Lesson Length in format we want HH:MM

$lessonLengthArray = preg\_split("/:/", $values['lessonLength']);

while (sizeof($lessonLengthArray) < 2) {

$lessonLengthArray[] = '00';

}

if (strlen($lessonLengthArray[0]) == 1){

$lessonLengthArray[0] = '0' . $lessonLengthArray[0];

}

if (strlen($lessonLengthArray[1]) == 1){

$lessonLengthArray[1] = '0' . $lessonLengthArray[1];

}

$lessonLength = $lessonLengthArray[0] . ':' . $lessonLengthArray[1];

//If a timetable info record exists

// If either Start Time or Lesson length is changed delete Constraints

// update Timetable Info

//else

// Create new timetable Info

if ($timetableInfo != null){

$origional\_startTime = $timetableInfo->getStartTime();

$origionalLessonLength = $timetableInfo->getLessonLength();

if (($startTime != $origional\_startTime) || ($lessonLength != $origionalLessonLength))

{

$DAOteacherConstraints = new DAO\_teacherconstraints();

$allTeacherConstraints = $DAOteacherConstraints->retrieveConstraints();

foreach ($allTeacherConstraints as $teacherConstraint)

{

$DAOteacherConstraints->deleteConstraint($teacherConstraint);

}

$DAOgroupConstrints = new DAO\_groupconstraints();

$allGroupConstraints = $DAOgroupConstrints->retrieveConstraints();

foreach ($allGroupConstraints as $groupConstraint)

{

$DAOgroupConstrints->deleteConstraint($groupConstraint);

}

}

$DAOtimetableInfo->updateTimetableInfo($timetableInfo,

$values['academicYear'],

$values['daysPerWeek'],

$values['lessonsPerDay'],

$startTime,

$lessonLength);

} else{

$DAOtimetableInfo->addTimetableInfo($values['academicYear'],

$values['daysPerWeek'],

$values['lessonsPerDay'],

$startTime,

$lessonLength);

}

$this->\_redirect('/admin/index/timetabling');

}

$this->view->timetableInfoForm = $timetableInfoForm;

$this->\_helper->layout->setLayout('/admin/admin\_layout');

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Timetables

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public function timetablesAction()

{

//Get User, Group and Room data

//Need to create list menu

$DAOusers = new DAO\_users();

$teachers = $DAOusers->retrieveSpecificUsers('teacher');

$teacherlist = "";

foreach ($teachers as $teacher) {

$teacherlist .= $this->renderTeacherList($teacher);

}

$this->view->teacherList = $teacherlist;

$DAOgroups = new DAO\_groups();

$groups = $DAOgroups->retrieveGroups();

$grouplist = "";

foreach ($groups as $group) {

$grouplist .= $this->renderGroupList($group);

}

$this->view->groupList = $grouplist;

$DAOrooms = new DAO\_rooms();

$rooms = $DAOrooms->retrieveRooms();

$roomlist = "";

foreach ($rooms as $room) {

if ($room->getRoomName() != 'Any'){

$roomlist .= $this->renderRoomList($room);

}

}

$this->view->roomList = $roomlist;

//Get Type (Teacher, Group, Room) and Id

//Load the the correct timetable

$type = $this->\_getParam('type');

$id = $this->\_getParam('id');

switch ($type) {

case "teacher":

$TeacherData = $DAOusers->retrieveUserByID($id);

if ($TeacherData != null) {

$selectedTeacher = $TeacherData->getUserForename() . ' ' . $TeacherData->getUserSurname();

$selectedTeacher = str\_replace(' ', '\_', $selectedTeacher);

$timetable = $this->loadTimetable('teachers', $selectedTeacher);

} else {

$timetable = null;

$this->view->errorMessage = 'The ' . $type . ' with the Id ' . $id . ' does not exist.';

}

break;

case "group":

$groupData = $DAOgroups->retrieveGroupByID($id);

if ($groupData != null) {

$selectedGroup = $groupData->getGroupName();

$selectedGroup = str\_replace(' ', '\_', $selectedGroup);

$timetable = $this->loadTimetable('groups', $selectedGroup);

} else {

$timetable = null;

$this->view->errorMessage = 'The ' . $type . ' with the Id ' . $id . ' does not exist.';

}

break;

case "room":

$roomData = $DAOrooms->retrieveRoomByID($id);

if ($roomData != null) {

$selectedRoom = $roomData->getRoomName();

$selectedRoom = str\_replace(' ', '\_', $selectedRoom);

$timetable = $this->loadTimetable('rooms', $selectedRoom);

} else {

$timetable = null;

$this->view->errorMessage = 'The ' . $type . ' with the Id ' . $id . ' does not exist.';

}

break;

case null: //redirect to first teacher

if ($teachers != null) {

$t = $teachers[0];

$firstTeacherId = $t->getUserId();

$this->\_redirect('/admin/timetabling/timetables/type/teacher/id/' . $firstTeacherId);

} else {

$timetable = null;

}

break;

default:

$timetable = null;

$this->view->errorMessage = 'The timetabling type ' . $type . ' does not exist.';

}

$generateForm = new Form\_Admin\_timetabling\_generate();

//Generate button was pushed so generate timetables

//Create FET input file

//Delete old Timetables

//If input file was succesfully completed run the FET application

//if FET completed successfully create Individual Timetables for teachers, groups and rooms

if ($this->\_request->getPost('submit')=='Generate' && $generateForm->isValid($\_POST))

{

$rootPath = $\_SERVER['DOCUMENT\_ROOT'];

$createFETinput = $this->createTimeTableFile();

$this->deleteTimetables();

if ($createFETinput == true) {

$fetCompleted = $this->runFET();

if ($fetCompleted == true) {

$file = $rootPath . '/fet/generated/timetables/msm/msm\_teachers\_days\_horizontal.html';

$timetableType = 'teachers';

$teacherTimetables = $this->parseFETtimetables($file, $timetableType);

$file = $rootPath . '/fet/generated/timetables/msm/msm\_groups\_days\_horizontal.html';

$timetableType = 'groups';

$groupTimetables = $this->parseFETtimetables($file, $timetableType);

$file = $rootPath . '/fet/generated/timetables/msm/msm\_rooms\_days\_horizontal.html';

$timetableType = 'rooms';

$roomTimetables = $this->parseFETtimetables($file, $timetableType);

if ($teacherTimetables == false || $groupTimetables ==false || $roomTimetables == false) {

$this->view->errorMessage = 'Error encountered while trying create individual timetables';

}

} else {

$this->view->errorMessage = 'Error encountered while trying to run FET.';

}

}else {

$this->view->errorMessage = 'Error creating FET Input file.';

}

$this->\_redirect($this->view->url());

}

//The timetable will be null if there is no timetable for the selected item

if ($timetable != null){

$this->view->timetable = $timetable;

$this->view->printTimetable = new Form\_Admin\_timetabling\_printbutton($timetable);

} else {

$timetable = "To create the timetable please click the Generate button.";

$this->view->timetable = $timetable;

$this->view->printTimetable = null;

}

$this->view->generateButton = $generateForm;

$this->\_helper->layout->setLayout('/admin/admin\_layout');

}

/\*\*

\* Renders a teacher into a \_timetable-subtype-item list template

\* @param models\entities\users $teacher

\* @return /timetabling/\_timetable-subtype-item.phtml

\*/

private function renderTeacherList (users $teacher)

{

$tableItem = new Zend\_View();

$tableItem->setScriptPath($this->view->getScriptPaths());

$tableItem->Id = $teacher->getUserId();

$userForename = $teacher->getUserForename();

$userSurname = $teacher->getUserSurname();

$tableItem->name = $userForename . ' ' . $userSurname;

$tableItem->type = "teacher";

return $tableItem->render('/timetabling/\_timetable-subtype-item.phtml');

}

/\*\*

\* Renders a group into a \_timetable-subtype-item list template

\* @param models\entities\groups $group

\* @return /timetabling/\_timetable-subtype-item.phtml

\*/

private function renderGroupList (groups $group)

{

$tableItem = new Zend\_View();

$tableItem->setScriptPath($this->view->getScriptPaths());

$tableItem->Id = $group->getGroupId();

$tableItem->name = $group->getGroupName();

$tableItem->type = "group";

return $tableItem->render('/timetabling/\_timetable-subtype-item.phtml');

}

/\*\*

\* Renders a room into a \_timetable-subtype-item list template

\* @param models\entities\rooms $room

\* @return /timetabling/\_timetable-subtype-item.phtml

\*/

private function renderRoomList (rooms $room)

{

$tableItem = new Zend\_View();

$tableItem->setScriptPath($this->view->getScriptPaths());

$tableItem->Id = $room->getRoomId();

$tableItem->name = $room->getRoomName();

$tableItem->type = "room";

return $tableItem->render('/timetabling/\_timetable-subtype-item.phtml');

}

/\*\*

\*

\* Create the XML input file which FET will

\* use to generate the timetables

\* @return bool

\*/

private function createTimeTableFile()

{

$DAOschoolInfo = new DAO\_schoolinfo();

$schoolInfo = $DAOschoolInfo->retriveSchoolInfo();

$schoolName = $schoolInfo->getSchoolName();

$DAOtimeTableInfo = new DAO\_timetableinfo();

$results = $DAOtimeTableInfo->retrieveTimetableInfo();

foreach ($results as $timetableInfo) {

$academicYear = $timetableInfo->getAcademicYear();

$daysPerWeek = $timetableInfo->getDaysPerWeek();

$lessonsPerDay = $timetableInfo->getLessonsPerDay();

$startTime = $timetableInfo->getStartTime();

$lessonLength = $timetableInfo->getLessonLength();

}

$DAOusers = new DAO\_users();

$students = $DAOusers->retrieveSpecificUsers('student');

$totalStudents = count($students);

$teachers =$DAOusers->retrieveSpecificUsers('teacher');

$DAOgroups = new DAO\_groups();

$groups = $DAOgroups->retrieveGroups();

$DAOsubjects = new DAO\_subjects();

$subjects = $DAOsubjects->retrieveSubjects();

$DAOlessons = new DAO\_lessons();

$lessons = $DAOlessons->retrieveLessons();

$DAOrooms = new DAO\_rooms();

$rooms = $DAOrooms->retrieveRooms();

//make timetable file

$newLine= "\r\n";

$tab = "\t";

$fileData = "<?xml version=\"1.0\" encoding=\"utf-8\"?>" . $newLine;

$fileData .= "<fet version=\"5.13.0\">";

$fileData .= "<Institution\_Name>" . $schoolName . "</Institution\_Name>" . $newLine;

$fileData .= "<Comments>Default comments</Comments>" . $newLine;

//-----------------

//Hours

//-----------------

$fileData .= "<Hours\_List>" . $newLine;

$fileData .= $tab . "<Number>" . $lessonsPerDay . "</Number>" . $newLine;

$startTimeArray = explode(":", $startTime);

$startHour = $startTimeArray[0];

$startMinutes = $startTimeArray[1];

$lessonLengthArray = explode(":", $lessonLength);

$lessonLengthHour = $lessonLengthArray[0];

$lessonLengthMinutes = $lessonLengthArray[1];

$endHour = $startHour + $lessonLengthHour;

$endMinutes = $startMinutes + $lessonLengthMinutes;

while ($endMinutes > 59) {

$endHour ++;

$endMinutes = $endMinutes - 60;

}

if ($startHour < 10 && strlen($startHour) < 2) {

$startHour = "0" . $startHour;

}

if ($endHour < 10 && strlen($endHour) < 2) {

$endHour = "0" . $endHour;

}

if ($startMinutes == 0 && strlen($startMinutes) < 2) {

$startMinutes = "0" . $startMinutes;

}

if ($endMinutes == 0 && strlen($endMinutes) < 2) {

$endMinutes = "0" . $endMinutes;

}

for ($i = 0; $i < $lessonsPerDay; $i++) {

$fileData .= $tab . "<Name>" . $startHour . "." . $startMinutes . " - " . $endHour . "." . $endMinutes . "</Name>" . $newLine;

$startHour = $endHour;

$startMinutes = $endMinutes;

$endHour = $endHour + $lessonLengthHour;

$endMinutes = $endMinutes + $lessonLengthMinutes;

while ($endMinutes > 59) {

$endHour++;

$endMinutes = $endMinutes - 60;

}

if ($startHour < 10 && strlen($startHour) < 2) {

$startHour = "0" . $startHour;

}

if ($endHour < 10 && strlen($endHour) < 2) {

$endHour = "0" . $endHour;

}

if ($startMinutes == 0 && strlen($startMinutes) < 2) {

$startMinutes = "0" . $startMinutes;

}

if ($endMinutes == 0 && strlen($endMinutes) < 2) {

$endMinutes = "0" . $endMinutes;

}

}

$fileData .= "</Hours\_List>" . $newLine;

//-----------------

//Days

//-----------------

$fileData .= "<Days\_List>" . $newLine;

$fileData .= $tab . "<Number>" . $daysPerWeek . "</Number>" . $newLine;

for ($i = 0; $i < $daysPerWeek; $i++) {

switch ($i) {

case 0:

$day = "Monday";

break;

case 1:

$day = "Tuesday";

break;

case 2:

$day = "Wednesday";

break;

case 3:

$day = "Thursday";

break;

case 4:

$day = "Friday";

break;

case 5:

$day = "Saturday";

break;

case 6:

$day = "Sunday";

break;

}

$fileData.= $tab . "<Name>" . $day . "</Name>" . $newLine;

}

$fileData .= "</Days\_List>" . $newLine;

//-----------------

//Students

//-----------------

$fileData .= "<Students\_List>" . $newLine;

$fileData .= $tab . "<Year>" . $newLine;

$fileData .= $tab . "<Name>Students</Name>" . $newLine;

$fileData .= $tab . "<Number\_of\_Students>" . $totalStudents . "</Number\_of\_Students>" . $newLine;

foreach ($groups as $group) {

$groupName = $group->getGroupName();

$users = $group->getUsers();

$fileData .= $tab . "<Group>" . $newLine;

$fileData .= $tab . $tab . "<Name>". $groupName ."</Name>" . $newLine;

$fileData .= $tab . $tab . "<Number\_of\_Students>" . count($users) . "</Number\_of\_Students>" . $newLine;

$fileData .= $tab . "</Group>" . $newLine;

}

$fileData .= "$tab . </Year>" . $newLine;

$fileData .= "</Students\_List>" . $newLine;

//-----------------

//Teachers

//-----------------

$fileData .= "<Teachers\_List>" . $newLine;

foreach ($teachers as $teacher) {

$fileData .= $tab . "<Teacher>" . $newLine;

$fileData .= $tab . $tab . "<Name>" . $teacher->getUserForename() . " " . $teacher->getUserSurname() . "</Name>" . $newLine;

$fileData .= $tab . "</Teacher>" . $newLine;

}

$fileData .= "</Teachers\_List>" . $newLine;

//-----------------

//Subjects

//-----------------

$fileData .= "<Subjects\_List>" . $newLine;

foreach ($subjects as $subject) {

$fileData .= $tab . "<Subject>" . $newLine;

$fileData .= $tab . $tab . "<Name>" . $subject->getSubject() . "</Name>" . $newLine;

$fileData .= $tab . "</Subject>" . $newLine;

}

$fileData .= "</Subjects\_List>" . $newLine;

//-----------------

//Activity Tags

//-----------------

$fileData .= "<Activity\_Tags\_List>" . $newLine;

$fileData .= "</Activity\_Tags\_List>" . $newLine;

//-----------------

//Activities

//-----------------

$fileData .= "<Activities\_List>" . $newLine;

foreach ($lessons as $lesson) {

$activityTeacher = $lesson->getTeacher()->getUserForename() . ' ' . $lesson->getTeacher()->getUserSurname();;

$activitySubject = $lesson->getSubjects()->getSubject();

$activityDuration = $lesson->getLessonlength();

$activityId = $lesson->getLessonid();

$activityGroup = $lesson->getGroup()->getGroupName();

$activityCount = $lesson->getLessoncount();

for ($i = 0; $i < $activityCount; $i++) {

$fileData .= $tab . "<Activity>" . $newLine;

$fileData .= $tab . $tab . "<Teacher>" . $activityTeacher . "</Teacher>" . $newLine;

$fileData .= $tab . $tab . "<Subject>" . $activitySubject . "</Subject>" . $newLine;

$fileData .= $tab . $tab . "<Duration>" . $activityDuration . "</Duration>" . $newLine;

$fileData .= $tab . $tab . "<Total\_Duration>" . $activityDuration . "</Total\_Duration>" . $newLine;

$fileData .= $tab . $tab . "<Id>" . $activityId . "</Id>" . $newLine;

$fileData .= $tab . $tab . "<Activity\_Group\_Id>" . "0" . "</Activity\_Group\_Id>" . $newLine;

$fileData .= $tab . $tab . "<Active>" . "true" . "</Active>" . $newLine;

$fileData .= $tab . $tab . "<Students>" . $activityGroup . "</Students>" . $newLine;

$fileData .= $tab . "</Activity>" . $newLine;

}

}

$fileData .= "</Activities\_List>" . $newLine;

//-----------------

//Buildings

//-----------------

$fileData .= "<Buildings\_List>" . $newLine;

$fileData .= "</Buildings\_List>" . $newLine;

//-----------------

//Rooms

//-----------------

$fileData .= "<Rooms\_List>" . $newLine;

foreach ($rooms as $room) {

$roomName = $room->getRoomName();

$roomCapacity = $room->getCapacity();

if (!($roomName == "Any")) {

$fileData .= $tab . "<Room>" . $newLine;

$fileData .= $tab . $tab . "<Name>" . $roomName . "</Name>" .$newLine;

$fileData .= $tab . $tab . "<Building>" . "" . "</Building>" .$newLine;

$fileData .= $tab . $tab . "<Capacity>" . $roomCapacity . "</Capacity>" .$newLine;

$fileData .= $tab . "</Room>" . $newLine;

}

}

$fileData .= "</Rooms\_List>" . $newLine;

//--------------------------------------

//Time Constraints

// Teacher Not available Constraints

// Group Not available Constraints

//--------------------------------------

$fileData .= "<Time\_Constraints\_List>" . $newLine;

$fileData .= $tab . "<ConstraintBasicCompulsoryTime>" . $newLine;

$fileData .= $tab . $tab . "<Weight\_Percentage>" . "100" . "</Weight\_Percentage>" . $newLine;

$fileData .= $tab . "</ConstraintBasicCompulsoryTime>" . $newLine;

$DAOteacherConstraints = new DAO\_teacherconstraints();

$allTeacherConstraints = $DAOteacherConstraints->retrieveConstraints();

//If any teacher constraints exist

if (sizeof($allTeacherConstraints) > 0){

foreach ($teachers as $t){

$teacherId = $t->getUserId();

$teacherName = $t->getUserForename() . ' ' . $t->getUserSurname();

$teacherConstraints = $DAOteacherConstraints->retrieveConstraintsByTeacher($teacherId);

if (sizeof($teacherConstraints) > 0){

$fileData .= $tab . "<ConstraintTeacherNotAvailableTimes>" . $newLine;

$fileData .= $tab . $tab . "<Weight\_Percentage>" . "100" . "</Weight\_Percentage>" . $newLine;

$fileData .= $tab . $tab . "<Teacher>" . $teacherName . "</Teacher>" . $newLine;

$fileData .= $tab . $tab . "<Number\_of\_Not\_Available\_Times>" . sizeof($teacherConstraints) . "</Number\_of\_Not\_Available\_Times>" . $newLine;

foreach ($teacherConstraints as $teacherConstraint){

$fileData .= $tab . $tab . "<Not\_Available\_Time>" . $newLine;

$fileData .= $tab . $tab . $tab . "<Day>" . $teacherConstraint->getDay() . "</Day>" . $newLine;

$fileData .= $tab . $tab . $tab . "<Hour>" . $teacherConstraint->getTime() . "</Hour>" . $newLine;

$fileData .= $tab . $tab . "</Not\_Available\_Time>" . $newLine;

}

$fileData .= $tab . "</ConstraintTeacherNotAvailableTimes>" . $newLine;

}

}

}

$DAOgroupConstraints = new DAO\_groupconstraints();

$allGroupConstraints = $DAOgroupConstraints->retrieveConstraints();

//If any group constraints exist

if (sizeof($allGroupConstraints) > 0){

foreach ($groups as $g){

$groupId = $g->getGroupId();

$groupName = $g->getGroupName();

$groupConstraints = $DAOgroupConstraints->retrieveConstraintsByGroup($groupId);

if (sizeof($groupConstraints) > 0){

$fileData .= $tab . "<ConstraintStudentsSetNotAvailableTimes>" . $newLine;

$fileData .= $tab . $tab . "<Weight\_Percentage>" . "100" . "</Weight\_Percentage>" . $newLine;

$fileData .= $tab . $tab . "<Students>" . $groupName . "</Students>" . $newLine;

$fileData .= $tab . $tab . "<Number\_of\_Not\_Available\_Times>" . sizeof($groupConstraints) . "</Number\_of\_Not\_Available\_Times>" . $newLine;

foreach ($groupConstraints as $groupConstraint){

$fileData .= $tab . $tab . "<Not\_Available\_Time>" . $newLine;

$fileData .= $tab . $tab . $tab . "<Day>" . $groupConstraint->getDay() . "</Day>" . $newLine;

$fileData .= $tab . $tab . $tab . "<Hour>" . $groupConstraint->getTime() . "</Hour>" . $newLine;

$fileData .= $tab . $tab . "</Not\_Available\_Time>" . $newLine;

}

$fileData .= $tab . "</ConstraintStudentsSetNotAvailableTimes>" . $newLine;

}

}

}

$fileData .= "</Time\_Constraints\_List>" . $newLine;

//-----------------

//Space Constraints

//If the room is any then randomly choose a room

//-----------------

$fileData .= "<Space\_Constraints\_List>" . $newLine;

$fileData .= $tab . "<ConstraintBasicCompulsorySpace>" . $newLine;

$fileData .= $tab . $tab . "<Weight\_Percentage>" . "100" . "</Weight\_Percentage>" . $newLine;

$fileData .= $tab . "</ConstraintBasicCompulsorySpace>" . $newLine;

foreach ($lessons as $lesson) {

$roomConstraintName = $lesson->getRoom()->getRoomName();

$roomConstraintId = $lesson->getLessonid();

$roomConstraintLock = "true";

if ($roomConstraintName == "Any") {

foreach ($rooms as $r) {

if (!($r->getRoomName() == "Any")) {

$roomArray[$r->getRoomId()] = $r->getRoomName();

}

}

$randomRoom = array\_rand($roomArray,1);

$roomConstraintName = $roomArray[$randomRoom];

$roomConstraintLock = "false";

}

$fileData .= $tab . "<ConstraintActivityPreferredRoom>" . $newLine;

$fileData .= $tab . $tab . "<Weight\_Percentage>" . "100" . "</Weight\_Percentage>" . $newLine;

$fileData .= $tab . $tab . "<Activity\_Id>" . $roomConstraintId . "</Activity\_Id>" . $newLine;

$fileData .= $tab . $tab . "<Room>" . $roomConstraintName . "</Room>" . $newLine;

$fileData .= $tab . $tab . "<Permanently\_Locked>" . $roomConstraintLock . "</Permanently\_Locked>" . $newLine;

$fileData .= $tab . "</ConstraintActivityPreferredRoom>" . $newLine;

}

$fileData .= "</Space\_Constraints\_List>" . $newLine;

//Write the FET file

try {

$fileData .= "</fet>";

$fp = fopen(APPLICATION\_PATH . '/fet/msm.fet', 'w');

fwrite($fp, $fileData);

fclose($fp);

} catch (Exception $e) {

return false;

}

return true;

}

/\*\*

\*

\* Run the FET timetabling program which

\* will generate the timetables

\* @return bool

\*/

private function runFET ()

{

try {

//@TODO determine OS (WINDOWS or LINUX and call correct app version)

//@TODO Windows only at the moment

$app = '"' . APPLICATION\_PATH . '\fet\fet.exe"';

$input = ' --inputfile="' . APPLICATION\_PATH . '/fet/msm.fet"';

$rootPath = $\_SERVER['DOCUMENT\_ROOT'];

$output = ' --outputdir="' . $rootPath . '/fet/generated"';

$extra = ' --warnifusingnotperfectconstraints=false --warnifusingstudentsminhoursdailywithallowemptydays=false';

$cmd = $app . $input . $output . $extra;

$test = exec($cmd, $execOutput, $execReturnVar);

if ($test == NULL) {

return false;

}

} catch (Exception $e) {

return false;

}

return true;

}

/\*\*

\*

\* This function takes the timetable generated by FET

\* and pulls out the individual Timetables

\* @param string $filename

\* @param string $timetableType

\*/

private function parseFETtimetables ($filename, $timetableType)

{

$rootPath = $\_SERVER['DOCUMENT\_ROOT'];

if (file\_exists($filename)) {

try {

$tableFound = false;

$fp = fopen($filename, 'r');

$table = null;

$teacherName = null;

while (! feof($fp)) {

$buffer = fgets($fp, 4096); // Read a line.

$renderTable = false;

if (strpos($buffer, '<table')) {

$tableFound = true;

}

if ($tableFound == true) {

//Get Teacher Name

if (strpos($buffer, '<tr><td rowspan')) {

$startPos = strpos($buffer, 'colspan') + 12;

$endPos = strpos($buffer, '</th></tr>') - $startPos;

$teacherName = substr($buffer, $startPos, $endPos);

$teacherName = trim($teacherName);

}

//Do not include footer with FET Details or caption

if (!(strpos($buffer, '<tr class="foot">'))) {

if (!(strpos($buffer, '<caption>'))){

$table .= $buffer;

}

}

}

//End of table found

if (strpos($buffer, '</table>')) {

$tableFound = false;

$renderTable = true;

}

//Empty the buffer

$buffer = "";

//Save Individual Timetable

//Save teachers to teachers directory, etc..

//If directory does not exist, create it

if ($renderTable == true && $teacherName != null) {

try {

$teacherName = str\_replace(' ', '\_', $teacherName);

if (!(file\_exists($rootPath . '/fet/saved/' . $timetableType . '/'))) {

mkdir($rootPath . '/fet/saved/' . $timetableType . '/');

}

$ttFile = fopen($rootPath . '/fet/saved/' . $timetableType . '/' . $teacherName . '.html', 'w');

fwrite($ttFile, $table);

fclose($ttFile);

$table = null;

} catch (Exception $e) {

return false;

}

}

}

fclose($fp);

} catch (Exception $e) {

return false;

}

} else {

return false;

}

return true;

}

/\*\*

\*

\* Load Timetable file and pass back timetable html

\* @param string $folderName

\* @param string $fileName

\* @return string $timetable || Null

\*/

private function loadTimetable($folderName, $fileName)

{

$rootPath = $\_SERVER['DOCUMENT\_ROOT'];

$filename = $rootPath . '/fet/saved/' . $folderName . '/' . $fileName . '.html';

if (file\_exists($filename)) {

try {

$fp = fopen($filename, 'r');

$timetable = fread($fp, filesize($filename));

fclose($fp);

} catch (Exception $e) {

return false;

}

} else {

return null;

}

return $timetable;

}

/\*\*

\* Delete all teacher, group and room timetables

\*/

private function deleteTimetables ()

{

$rootPath = $\_SERVER['DOCUMENT\_ROOT'];

$folderArray = array(1 => '/fet/saved/teachers', 2 => '/fet/saved/groups', 3 => '/fet/saved/rooms');

$teachersFolder = '/fet/saved/teachers';

$handle = null;

foreach ($folderArray as $folder){

if (file\_exists($rootPath . $folder)) {

if ($handle = opendir($rootPath . $folder)) {

while (false !== ($file = readdir($handle))) {

unlink($rootPath . $folder . '/' . $file);

}

}

closedir($handle);

}

}

}

}

##### Views

###### Index

Help.phtml

<p>Help</p>

<br>

<p>Welcome to the help section. Just click on a topic to view.</p>

<br>

<ul>

<li>

<a href=*"#Manage"*>Manage</a>

<ul>

<li>

<a href=*"#School"*>School</a>

</li>

<li>

<a href=*"#Rooms"*>Rooms</a>

</li>

<li>

<a href=*"#Subjects"*>Subjects</a>

</li>

<li>

<a href=*"#Teachers"*>Teachers</a>

</li>

<li>

<a href=*"#Students"*>Students</a>

</li>

<li>

<a href=*"#Groups"*>Groups</a>

</li>

<li>

<a href=*"#Lessons"*>Lessons</a>

</li>

</ul>

</li>

<li>

<a href=*"#Timetabling"*>Timetabling</a>

<ul>

<li>

<a href=*"#Info"*>Timetable Information</a>

</li>

<li>

<a href=*"#Timetables"*>Timetables</a>

</li>

</ul>

</li>

<li>

<a href=*"#UserPanel"*>User Panel</a>

<ul>

<li>

<a href=*"#SystemSetup"*>System Setup</a>

<ul>

<li>

<a href=*"#Instruments"*>Instruments</a>

</li>

<li>

<a href=*"#Personalisation"*>Personalisation</a>

</li>

<li>

<a href=*"#News"*>News</a>

</li>

</ul>

</li>

<li>

<a href=*"#MyAccount"*>My Account</a>

<ul>

<li>

<a href=*"#Username"*>Change Username</a>

</li>

<li>

<a href=*"#Password"*>Change Password</a>

</li>

</ul>

</li>

</ul>

</li>

</ul>

<br>

<b><a name=*"Manage"*>Manage</a></b>

<br>

<p>

The manage section is the section which contains all the necessary data required for the successfull creation of a timetable.

</p>

<br>

<b><a name=*"School"*>School</a></b>

<br>

<p>

This information identifies the school, this information is important and should be kept up to date, especially the email address.

</p>

<br>

<b><a name=*"Rooms"*>Rooms</a></b>

<br>

<p>

This section is where you keep a record of all the rooms within the school, and their capacity.

When creating a lesson you may assign a specific room to a specific group or you may choose to have one chosen at random by simply choosing Any room.

</p>

<br>

<b><a name=*"Subjects"*>Subjects</a></b>

<br>

<p>

This section is where you enter all of the subject taught at the school.

When creating a lesson it is these subjects you will be choosing from.

</p>

<br>

<b><a name=*"Teachers"*>Teachers</a></b>

<br>

<p>

The section is where you can add and edit the teachers within the school.

When you add a teacher the system will automatically send a registration email to the teachers email address you have entered.

<br>

Once you have created a teacher you can then add teacher time constraints by clicking the constraints link beside the teachers name. A teacher time constraint is just a time when the teacher is not available.

This will be taken into account when generating the time tables.

</p>

<br>

<b><a name=*"Students"*>Students</a></b>

<br>

<p>

The section is where you can add and edit the students attending the school.

When you add a student the system will automatically send a registration email to the students email address you have entered.

Once you you have created a student you can then edit the instruments that they are associated with by click the instruments link beside the students name.

</p>

<br>

<b><a name=*"Groups"*>Groups</a></b>

<br>

<p>

This section is where you can add or edit the groups.

Groups consist of one or more students.

When creating a lesson it is a group which you are creating the lesson for.

<br>

Once you have created a group you can then add time constraints for that group by clicking on the constraints link beside the group.

A group constraint just specifies a time at which this group is not available.

</p>

<br>

<b><a name=*"Lessons"*>Lessons</a></b>

<br>

<p>

In this section you can add or edit a lesson. Lessons are what makes up the timetable.

The Lesson Length is the length of the lesson in periods.

The Lesson Count is the number of times the lesson is repeated per week.

</p>

<br>

<b><a name=*"Timetabling"*>Timetabling</a></b>

<br>

<p>

In this section you can edit the timetable information, generate amd view timetables.

</p>

<br>

<b><a name=*"Info"*>Timetabling Information</a></b>

<br>

<p>

In this section you specify the timetable information.

Here is a brief description of each of the fields.

</p>

<ul>

<li>

<b>Academic Year</b>: The current academic year.

</li>

<li>

<b>Working Days per week</b>: The number of days per week that the school is open (Minimum 1, Maximum 7).

</li>

<li>

<b>Lessons per day</b>: This is the number of time periods/lessons in a day.

</li>

<li>

<b>Start Time</b>: This is the time you wish to start the first lessons at.

</li>

<li>

<b>Lesson Length</b>: This is the length of each period/Lesson.

</li>

</ul>

<p>

<b>Important</b>: Please be aware that changing either start time or lesson length will cause all time constraints to be removed.

</p>

<br>

<b><a name=*"Timetables"*>Timetables</a></b>

<br>

<p>

This is where you can generate or view the timetables. There are three catagories of timetable:

</p>

<ul>

<li>

Teacher: A timetable will be created for each teacher in the system.

</li>

<li>

Groups: A time table will be created for each group in the system.

</li>

<li>

Rooms: A timetable will be created for each room in the system.

</li>

</ul>

<p>

To generate the timetable just click the generate button.

<br>

To view a specific timetable click on the catagory and then click on the timetable you wish to view.

<br>

To save a timetable click the save button, this will open the timetable in a new window, which you may then print.

</p>

<p>

<b>Important</b>: Please be aware that when you generate new timetables the old timetable are deleted.

</p>

<br>

<b><a name=*"UserPanel"*>User Panel</a></b>

<br>

<p>

From the User Panel you have the choice of selecting either System Setup or My Account.

</p>

<br>

<b><a name=*"SystemSetup"*>System Setup</a></b>

<br>

<p>

In this section you can edit the instruments used by the school,

personalise the site by adding your own banner or create and edit news articles,

which will appear on your frount page as well as the teacher and student home pages.

</p>

<br>

<b><a name=*"Instruments"*>Instruments</a></b>

<br>

<p>

In this section you add, edit or delete the musical instruments associated with the school.

</p>

<br>

<b><a name=*"Personalisation"*>Personalisation</a></b>

<br>

<p>

In this section you can upload a new image to be used as the site banner,

simply click the browser button, select the image you wish to use and click ok.

Then click the upload button, once the image has been uploaded simply select it with the slider and click on it to use it.

<br>

The following types of images files are acceptable: jpg, gif or bmp.

<br>

The recommended dimensions for a new banner are 960x122, while you may use any image you wish one with these dimensions will look the best.

</p>

<br>

<b><a name=*"News"*>News</a></b>

<br>

<p>

The news section allows you to publish, edit and manage news articles.

All news articles published will appear it the following locations: Login Page, teacher home and student home.

</p>

<br>

<b><a name=*"MyAccount"*>My Account</a></b>

<br>

<p>

In this section you can change your personal setting, such as your username and password.

It is recommended that you change your password regularly.

</p>

<br>

<b><a name=*"Username"*>Change Username</a></b>

<br>

<p>

To change your username, simply enter the new username you wish to use, if its not already taken its yours.

</p>

<br>

<b><a name=*"Password"*>Change Password</a></b>

<br>

<p>

To change your password, you must enter your current password,

the new password you wish to use, followed by verification of the new password you wish to use.

</p>

<br>

Index.phtml

<p>Welcome <?php **echo** $this->userName; ?></p>

<br>

<p>Click on a link bellow to navigate to that section</p>

<br>

<a href=*"/admin/index/manage"*>Manage</a>

<br>

<a href=*"/admin/index/timetabling"*>Timetabling</a>

<br>

<a href=*"/admin/index/userpanel"*>User Panel</a>

Manage.phtml

<p>Welcome <?php **echo** $this->userName; ?></p>

<br>

<p>Click on a link bellow to navigate to that section</p>

<br>

<a href=*"/admin/index/manage"*>Manage</a>

<br>

<a href=*"/admin/index/timetabling"*>Timetabling</a>

<br>

<a href=*"/admin/index/userpanel"*>User Panel</a>

Myaccount.phtml

<p>My Account</p>

<br>

<p>Click on a link bellow to navigate to that section</p>

<br>

<a href=*"/admin/myaccount/password/"*>Change Password</a>

<br>

<a href=*"/admin/myaccount/username/"*>Change Username</a>

Systemsetup.phtml

<p>System Setup</p>

<br>

<p>Click on a link bellow to navigate to that section</p>

<br>

<a href=*"/admin/setup/instruments/"*>Instruments</a>

<br>

<a href=*"/admin/setup/personalisation/"*>Personalisation</a>

<br>

<a href=*"/admin/setup/news/"*>News</a>

Timetabling.phtml

<p>Timetabling</p>

<br>

<p>Click on a link bellow to navigate to that section</p>

<br>

<a href=*"/admin/timetabling/information/"*>Timetable Information</a>

<br>

<a href=*"/admin/timetabling/timetables/"*>Timetabling</a>

userpanel.phtml

<p>User Panel</p>

<br>

<p>Click on a link bellow to navigate to that section</p>

<br>

<a href=*"/admin/index/systemsetup"*>System Setup</a>

<br>

<a href=*"/admin/index/myaccount"*>My Account</a>

###### Manage

\_group-constraint-item.phtml

<tr>

<td>

<?php **echo** $this->day; ?>

</td>

<td>

<?php **echo** $this->time;?>

</td>

<td>

<a href="*javascript:Delete(*<?php **echo** $this->groupId; ?>*,* <?php **echo** $this->id; ?>*)*">Delete</a>

</td>

</tr>

\_group-item.phtml

<script type=*"text/javascript"*>

<!--

**function** Delete($id)

{

**if** (confirm('Are you sure you want to delete this group?'))

{

location.href = "/admin/manage/groupdelete/id/" + $id;

}

};

//-->

</script>

<tr>

<td>

<?php **echo** $this->groupName; ?>

</td>

<td>

<?php **echo** $this->groupShortName;?>

</td>

<td>

<?php **echo** $this->groupType;?>

</td>

<td>

<?php **echo** $this->participants;?>

</td>

<td>

<a href="*/admin/manage/groupconstraints/id/*<?php **echo** $this->groupID; ?>">Constraints</a>

</td>

<td>

<a href="*/admin/manage/groupedit/id/*<?php **echo** $this->groupID; ?>">Edit</a>

</td>

<td>

<a href="*javascript:Delete(*<?php **echo** $this->groupID; ?>*)*">Delete</a>

</td>

</tr>

\_group-student-item.phtml

<tr>

<td>

<?php **echo** $this->forename; ?>

</td>

<td>

<?php **echo** $this->surname;?>

</td>

<td>

<?php **echo** $this->email;?>

</td>

<td>

<?php **echo** $this->address;?>

</td>

<td>

<a href="*/admin/manage/studentingroup/id/*<?php **echo** $this->studentID; ?>*/groupid/*<?php **echo** $this->groupId; ?>"><?php **echo** $this->studentInGroup; ?></a>

</td>

</tr>

\_grouptype-item.phtml

<script type=*"text/javascript"*>

<!--

**function** Delete($id, $iid)

{

**if** (confirm('Are you sure you want to delete this group type?'))

{

location.href = "/admin/manage/grouptpyedelete/type/" + $id + "/group/" + $iid;

}

};

//-->

</script>

<tr>

<td>

<?php **echo** $this->groupTypeName; ?>

</td>

<td>

<a href="*/admin/manage/grouptypeedit/type/*<?php **echo** $this->groupTypeId; ?>*/group/*<?php **echo** $this->groupId; ?>">Edit</a>

</td>

<td>

<a href="*javascript:Delete(*<?php **echo** $this->groupTypeId; ?>*,* <?php **echo** $this->groupId; ?>*)*">Delete</a>

</td>

</tr>

\_instrument-item.phtml

<tr>

<td>

<?php **echo** $this->instrumentName;?>

</td>

<td>

<?php **echo** $this->optionSelect; ?>

</td>

<td>

<a href="*/admin/manage/studenteditinstused/id/*

<?php **echo** $this->studentID; ?>*/instrument/*

<?php **echo** $this->instrumentID; ?>"><?php **echo** $this->instrumentUsed;?></a>

</td>

</tr>

\_lesson-item.phtml

<script type=*"text/javascript"*>

<!--

**function** Delete($id)

{

**if** (confirm('Are you sure you want to delete this lessons?'))

{

location.href = "/admin/manage/lessondelete/id/" + $id;

}

};

//-->

</script>

<tr>

<td>

<?php **echo** $this->subject; ?>

</td>

<td>

<?php **echo** $this->group;?>

</td>

<td>

<?php **echo** $this->teacher;?>

</td>

<td>

<?php **echo** $this->room;?>

</td>

<td>

<?php **echo** $this->lessonLength;?>

</td>

<td>

<?php **echo** $this->lessonCount;?>

</td>

<td>

<a href="*/admin/manage/lessonedit/id/*<?php **echo** $this->lessonId; ?>">Edit</a>

</td>

<td>

<a href="*javascript:Delete(*<?php **echo** $this->lessonId; ?>*)*">Delete</a>

</td>

</tr>

\_room-item.phtml

<script type=*"text/javascript"*>

<!--

**function** Delete($id)

{

**if** (confirm('Are you sure you want to delete this room?'))

{

location.href = "/admin/manage/roomdelete/id/" + $id;

}

};

//-->

</script>

<tr>

<td>

<?php **echo** $this->roomName; ?>

</td>

<td>

<?php **echo** $this->roomShortName;?>

</td>

<td>

<?php **echo** $this->capacity;?>

</td>

<td>

<a href="*/admin/manage/roomedit/id/*<?php **echo** $this->roomID; ?>">Edit</a>

</td>

<td>

<a href="*javascript:Delete(*<?php **echo** $this->roomID; ?>*)*">Delete</a>

</td>

</tr>

\_student-item.phtml

<script type=*"text/javascript"*>

<!--

**function** Delete($id)

{

**if** (confirm('Are you sure you want to delete this student?'))

{

location.href = "/admin/manage/studentdelete/id/" + $id;

}

};

//-->

</script>

<tr>

<td>

<?php **echo** $this->forename; ?>

</td>

<td>

<?php **echo** $this->surname;?>

</td>

<td>

<?php **echo** $this->email;?>

</td>

<td>

<?php **echo** $this->address;?>

</td>

<td>

<?php **echo** $this->telephone;?>

</td>

<td>

<?php **echo** $this->mobile;?>

</td>

<td>

<a href="*/admin/manage/studenteditinstruments/id/*<?php **echo** $this->studentID; ?>">Instruments</a>

</td>

<td>

<a href="*/admin/manage/studentedit/id/*<?php **echo** $this->studentID; ?>">Edit</a>

</td>

<td>

<a href="*javascript:Delete(*<?php **echo** $this->studentID; ?>*)*">Delete</a>

</td>

</tr>

\_subject-item.phtml

<script type=*"text/javascript"*>

<!--

**function** Delete($id)

{

**if** (confirm('Are you sure you want to delete this subject?'))

{

location.href = "/admin/manage/subjectdelete/id/" + $id;

}

};

//-->

</script>

<tr>

<td>

<?php **echo** $this->subject; ?>

</td>

<td>

<a href="*/admin/manage/subjectedit/id/*<?php **echo** $this->subjectID; ?>">Edit</a>

</td>

<td>

<a href="*javascript:Delete(*<?php **echo** $this->subjectID; ?>*)*">Delete</a>

</td>

</tr>

\_teacher-constraint-item.phtml

<tr>

<td>

<?php **echo** $this->day; ?>

</td>

<td>

<?php **echo** $this->time;?>

</td>

<td>

<a href="*javascript:Delete(*<?php **echo** $this->teacherId; ?>*,* <?php **echo** $this->id; ?>*)*">Delete</a>

</td>

</tr>

\_teacher-item.phtml

<tr>

<td>

<?php **echo** $this->forename; ?>

</td>

<td>

<?php **echo** $this->surname;?>

</td>

<td>

<?php **echo** $this->email;?>

</td>

<td>

<?php **echo** $this->address;?>

</td>

<td>

<?php **echo** $this->telephone;?>

</td>

<td>

<?php **echo** $this->mobile;?>

</td>

<td>

<a href="*/admin/manage/teacherconstraints/id/*<?php **echo** $this->teacherID; ?>">Constraints</a>

<td>

<td>

<a href="*/admin/manage/teacheredit/id/*<?php **echo** $this->teacherID; ?>">Edit</a>

</td>

<td>

<a href="*javascript:Delete(*<?php **echo** $this->teacherID; ?>*)*">Delete</a>

</td>

</tr>

editschool.phtml

<p>Edit School Information</p>

<?php **echo** $this->editSchoolForm?>

groupadd.phtml

<p>Add a new group</p>

<?php **echo** $this->groupAddForm; ?>

groupconstraintdelete.phtml

This file is empty, its just a placeholder for the Controller Action.

groupconstraints.phtml

<script type=*"text/javascript"*>

<!--

**function** Delete($group, $id)

{

**if** (confirm('Are you sure you want to delete this constraint?'))

{

location.href = "/admin/manage/groupconstraintdelete/group/" + $group + "/id/" + $id;

}

};

//-->

</script>

<p>Group not available constraints for <?php **echo** $this->groupName; ?></p>

<br>

<table id=*"rendered\_table"*>

<tr>

<td>

<b>Day</b>

</td>

<td>

<b>Time</b>

</td>

<td>

</td>

</tr>

<?php **echo** $this->tableContent; ?>

</table>

<br>

<div id=*"the-border"*>

<?php **echo** $this->constraintsForm; ?>

</div>

<?php **echo** $this->errorMessage; ?>

groupdelete.phtml

This file is empty, its just a placeholder for the Controller Action.

groupedit.phtml

<p>Group Details</p>

<?php **echo** $this->groupEdit;?>

<?php **echo** $this->errorMessage; ?>

groupeditstudents.phtml

<p>Edit Students in Group</p>

<br>

<table id=*"rendered\_table"*>

<tr >

<td>

<B>Forename</B>

</td>

<td>

<B>Surname</B>

</td>

<td>

<B>E-mail</B>

</td>

<td>

<B>Address</B>

</td>

<td>

<B>In Group</B>

</td>

</tr>

<?php **echo** $this->studentsTable;?>

</table>

<?php **echo** $this->cancelButton;?>

<?php **echo** $this->errorMessage; ?>

groups.phtml

<p>Groups</p>

<br>

<table id=*"rendered\_table"*>

<tr >

<td>

<B>Group Name</B>

</td>

<td>

<B>Short Group Name</B>

</td>

<td>

<B>Group Type</B>

</td>

<td>

<B>Participants</B>

</td>

</tr>

<?php **echo** $this->tableContent;?>

</table>

<br>

<?php **echo** $this->groupsForm;?>

grouptpyedelete.phtml

This file is empty; it’s just a placeholder for the Controller Action.

grouptype.phtml

<script type=*"text/javascript"*>

<!--

$(**function**(){

$("#Add").click(**function**(e){

**var** newGroup = prompt('Enter New group type.');

**if** (newGroup != **null** && newGroup != "")

{

location.href = "/admin/manage/grouptypeadd/id/" + newGroup + "/groupid/ + <?php **echo** $this->groupID;?>";

}

});

});

//-->

</script>

<p>Group Types</p>

<table id=*"rendered\_table"*>

<tr>

<th>Group Type</th>

</tr>

<?php **echo** $this->tableContent;?>

</table>

<table id=*"rendered\_table"*>

<tr>

<td>

<button id=*"Add"*>Add Group Type</button>

</td>

</tr>

<tr>

<td>

<?php **echo** $this->cancelButton;?>

</td>

</tr>

</table>

grouptypeadd.phtml

This file is empty; it’s just a placeholder for the Controller Action.

grouptypeedit.phtml

<p>Edit Group Type</p>

<?php **echo** $this->groupTypeForm; ?>

lessonadd.phtml

<p>Add Lesson</p>

<?php **echo** $this->lessonForm;?>

lessondelete.phtml

This file is empty; it’s just a placeholder for the Controller Action.

lessonedit.phtml

<p>Edit Lesson</p>

<?php **echo** $this->lessonEditForm;?>

<?php **echo** $this->errorMessage;?>

lessons.phtml

<p>Lessons</p>

<br>

<table id=*"rendered\_table"*>

<tr>

<td>

<B>Subject</B>

</td>

<td>

<B>Group</B>

</td>

<td>

<B>Teacher</B>

</td>

<td>

<B>Room</B>

</td>

<td>

<B>Lesson Length</B>

</td>

<td>

<B>Lesson Count</B>

</td>

</tr>

<?php **echo** $this->tableContent;?>

</table>

<?php **echo** $this->addButton;?>

<br>

roomadd.phtml

<h1>Add Room</h1>

<?php **echo** $this->addRoom;?>

roomdelete.phtml

This file is empty; it’s just a placeholder for the Controller Action.

roomedit.phtml

<h1>Room Details</h1>

<?php **echo** $this->editRoomForm;?>

<?php **echo** $this->errorMessage; ?>

rooms.phtml

<p>Rooms</p>

<br>

<table id=*"rendered\_table"*>

<tr >

<td>

<B>Room Name</B>

</td>

<td>

<B>Room Short Name</B>

</td>

<td>

<B>Capacity</B>

</td>

</tr>

<?php **echo** $this->tableContent;?>

</table>

<br>

<?php **echo** $this->roomForm;?>

school.phtml

<p>School Information</p>

<B>School Name:</B>

<?php **echo** $this->schoolname?>

<br>

<B>School Address:</B>

<?php **echo** $this->schooladdress?>

<br>

<B>School Telephone:</B>

<?php **echo** $this->schoolPhone?>

<br>

<B>School E-mail:</B>

<?php **echo** $this->schoolEmail?>

<?php **echo** $this->editButton?>

studentadd.phtml

<p>Add Student</p>

<?php **echo** $this->addStudent;?>

<?php **echo** $this->errorMessage;?>

studentdelete.phtml

This file is empty; it’s just a placeholder for the Controller Action.

studentedit.phtml

<h1>Student Details</h1>

<?php **echo** $this->editStudentForm;?>

<?php **echo** $this->errorMessage; ?>

studenteditinstruments.phtml

<p>Instruments</p>

<table border=*"0"* width=*"50%"* cellpadding=*"5"*>

<tr>

<td>

<b>Instrument</b>

</td>

<td>

<b>Grade</b>

</td>

<td>

<b>Used</b>

</td>

</tr>

<?php **echo** $this->tableContent;?>

</table>

<?php **echo** $this->cancelButton;?>

<?php **echo** $this->errorMessage;?>

studenteditinstused.phtml

This file is empty; it’s just a placeholder for the Controller Action.

studentingroup.phtml

This file is empty; it’s just a placeholder for the Controller Action.

students.phtml

<p>Students</p>

<br>

<table id=*"rendered\_table"*>

<tr >

<td>

<B>Forename</B>

</td>

<td>

<B>Surname</B>

</td>

<td>

<B>E-mail</B>

</td>

<td>

<B>Address</B>

</td>

<td>

<B>Telephone</B>

</td>

<td>

<B>Mobile</B>

</td>

</tr>

<?php **echo** $this->tableContent;?>

</table>

<br>

<?php **echo** $this->studentsForm;?>

subjectadd.phtml

<h1>Add Subject</h1>

<?php **echo** $this->addSubject;?>

<?php **echo** $this->errorMessage;?>

subjectdelete.phtml

This file is empty; it’s just a placeholder for the Controller Action.

subjectedit.phtml

<p>Subject Details</p>

<br>

<?php **echo** $this->editSubjectForm;?>

<?php **echo** $this->errorMessage; ?>

subjects.phtml

<p>Subjects</p>

<br>

<table id=*"rendered\_table"*>

<tr >

<td>

<B>Subject</B>

</td>

</tr>

<?php **echo** $this->tableContent;?>

</table>

<br>

<?php **echo** $this->subjectsForm;?>

teacheradd.phtml

<h1>Add Teacher</h1>

<?php **echo** $this->addTeacher;?>

<?php **echo** $this->errorMessage;?>

teacherconstraintdelete.phtml

This file is empty; it’s just a placeholder for the Controller Action.

teacherconstraints.phtml

<script type=*"text/javascript"*>

<!--

**function** Delete($teacher, $id)

{

**if** (confirm('Are you sure you want to delete this constraint?'))

{

location.href = "/admin/manage/teacherconstraintdelete/teacher/" + $teacher + "/id/" + $id;

}

};

//-->

</script>

<p>Teacher not available constraints for <?php **echo** $this->teacherName; ?></p>

<br>

<table id=*"rendered\_table"*>

<tr>

<td>

<b>Day</b>

</td>

<td>

<b>Time</b>

</td>

<td>

</td>

</tr>

<?php **echo** $this->tableContent; ?>

</table>

<br>

<div id=*"the-border"*>

<?php **echo** $this->constraintsForm; ?>

</div>

<?php **echo** $this->errorMessage; ?>

teacherdelete.phtml

This file is empty; it’s just a placeholder for the Controller Action.

teacheredit.phtml

<h1>Teacher Details</h1>

<?php **echo** $this->editTeacherForm;?>

<?php **echo** $this->errorMessage; ?>

teachers.phtml

<script type=*"text/javascript"*>

<!--

**function** Delete($id)

{

**if** (confirm('Are you sure you want to delete this teacher?'))

{

location.href = "/admin/manage/teacherdelete/id/" + $id;

}

};

//-->

</script>

<p>Teachers</p>

<br>

<table id=*"rendered\_table"*>

<tr >

<td>

<B>Forename</B>

</td>

<td>

<B>Surname</B>

</td>

<td>

<B>E-mail</B>

</td>

<td>

<B>Address</B>

</td>

<td>

<B>Telephone</B>

</td>

<td>

<B>Mobile</B>

</td>

</tr>

<?php **echo** $this->tableContent;?>

</table>

<br>

<?php **echo** $this->teachersForm;?>

###### Myaccount

Password

<p>Change Password</p>

<?php **echo** $this->passwordForm;?>

<?php **echo** $this->errorMessage;?>

Username

<p>Change Username</p>

<?php **echo** $this->usernameForm; ?>

<?php **echo** $this->errorMessage; ?>

###### Setup

\_image-list-item.phtml

<li><a href="*/admin/setup/useimage/id/*<?php **echo** $this->fileName; ?>"><img src="<?php **echo** $this->imageItem; ?>" alt=*"MSM Image"* /></a></li>

\_instrument-item.phtml

<script type=*"text/javascript"*>

<!--

**function** Delete($id)

{

**if** (confirm('Are you sure you want to delete this instrument?'))

{

location.href = "/admin/setup/instrumentdelete/id/" + $id;

}

};

//-->

</script>

<tr>

<td>

<?php **echo** $this->instrumentName;?>

</td>

<td>

<a href="*/admin/setup/instrumentused/id/*<?php **echo** $this->instrumentID; ?>"><?php **echo** $this->instrumentUsed;?></a>

</td>

<td>

<a href="*/admin/setup/instrumentedit/id/*<?php **echo** $this->instrumentID; ?>">Edit</a>

</td>

<td>

<a href="*javascript:Delete(*<?php **echo** $this->instrumentID; ?>*)*">Delete</a>

</td>

</tr>

\_news-item.phtml

<script type=*"text/javascript"*>

<!--

**function** Delete($id)

{

**if** (confirm('Are you sure you want to delete this news item?'))

{

location.href = "/admin/setup/newsarticledelete/id/" + $id;

}

};

//-->

</script>

<table id=*"News-Item"*>

<tr>

<td>

<b><?php **echo** $this->title;?></b>

|

<?php **echo** $this->publishedDate ;?>

</td>

<td align=*"right"*>

<a href="*/admin/setup/newsitemedit/id/*<?php **echo** $this->newsId; ?>">Edit</a>

|

<a href="*javascript:Delete(*<?php **echo** $this->newsId; ?>*)*">Delete</a>

</td>

</tr>

<tr>

<td>

<?php **echo** nl2br($this->article);?>

</td>

</tr>

</table>

<br>

instrumentadd.phtml

<p>Add New Instrument</p>

<?php **echo** $this->instrumentAddForm; ?>

<?php **echo** $this->errorMessage; ?>

instrumentdelete.phtml

This file is empty; it’s just a placeholder for the Controller Action.

instrumentedit.phtml

<p>Edit Instrument</p>

<?php **echo** $this->instrumentEditForm; ?>

<?php **echo** $this->errorMessage; ?>

instruments.phtml

<p>Instruments</p>

<table border=*"0"* width=*"60%"* cellpadding=*"5"*>

<tr>

<th>Name</th>

<th>Used</th>

</tr>

<?php **echo** $this->tableContent?>

</table>

<?php **echo** $this->addButton?>

instrumentused.phtml

This file is empty; it’s just a placeholder for the Controller Action.

news.phtml

<p>News</p>

<br>

<?php **echo** $this->newsForm?>

<?php **echo** $this->newsItems?>

newsarticledelete.phtml

This file is empty; it’s just a placeholder for the Controller Action.

newsitemedit.phtml

<p>Edit News Item</p>

<?php **echo** $this->newsItemEditForm?>

personalisation.phtml

<p>Personalisation</p>

<br>

<div id=*"slider"*>

<ul>

<?php **echo** $this->listContent; ?>

</ul>

</div>

<br>

<br>

<div id=*"the-border"*>

<?php **echo** $this->uploadForm; ?>

</div>

useimage.phtml

This file is empty; it’s just a placeholder for the Controller Action.

###### Timetabling

\_timetable-subtype-item.phtml

<li><a

href="*/admin/timetabling/timetables/type/*<?php

**echo** $this->type;

?>*/id/*<?php

**echo** $this->Id;

?>"><?php

**echo** $this->name;

?></a></li>

Information.phtml

<p>Timetabling Information</p>

<br>

<div id=*"the-border"* style="width: *85%*">

<p>

<b>

Note:

</b>

Please be aware that changing either start time or lesson length will cause all time constraints to be removed.

</p>

</div>

<br>

<?php **echo** $this->timetableInfoForm;?>

Timetables.phtml

<p>Timetables</p>

<script>

$(**function**(){

$("#teachers-link").click(**function**(e){

e.preventDefault();

$('#data ul').hide();

$('#teachers-data').show();

});

$("#groups-link").click(**function**(e){

e.preventDefault();

$('#data ul').hide();

$('#groups-data').show();

});

$("#rooms-link").click(**function**(e){

e.preventDefault();

$('#data ul').hide();

$('#rooms-data').show();

});

});

</script>

<script type=*"text/javascript"*>

**function** printTT(data){

myWindow = window.open('','');

myWindow.document.write(data);

}

</script>

<ul id=*"tt-selector"*>

<li><a id=*"teachers-link"* href=*"#"*>Teachers</a></li>

<li><a id=*"groups-link"* href=*"#"*>Groups</a></li>

<li><a id=*"rooms-link"* href=*"#"*>Rooms</a></li>

</ul>

<div id=*"data"*>

<ul id=*"teachers-data"*>

<?php **echo** $this->teacherList; ?>

</ul>

<ul id=*"groups-data"*>

<?php **echo** $this->groupList; ?>

</ul>

<ul id=*"rooms-data"*>

<?php **echo** $this->roomList; ?>

</ul>

</div>

<br>

<br>

<br>

<div id=*"center\_timetable"*>

<?php **echo** $this->timetable; ?>

</div>

<div id=*"center\_b"*>

<?php **echo** $this->printTimetable; ?>

</div>

<?php **echo** $this->generateButton; ?>

<?php **echo** $this->errorMessage; ?>

#### Default

##### Controllers

###### LoginController

<?php

/\*\*

\*

\* LoginController

\* @author Aaron Redmond

\* @version 1.0

\* @created 07-Feb-2011

\*/

require\_once APPLICATION\_PATH . '/modules/AbstractController.php';

use models\entities\users;

use models\entities\Session;

use models\entities\news;

class LoginController extends AbstractController

{

/\*\*

\* Show the Login page

\* Handle the login procedure

\*/

public function indexAction ()

{

if (! $this->DAOSession->isLoggedIn()) {

$loginForm = new Form\_Login();

$this->view->loginForm = $loginForm;

//Render news items

$DAOnews = new DAO\_news();

$newsItems = $DAOnews->retrieveNewsItems();

$tableItem = "";

foreach ($newsItems as $newsItem){

$tableItem .= $this->renderNewsTable($newsItem);

}

//Validate Data

if ($this->\_request->isPost() && $loginForm->isValid($\_POST)) {

$DAOObject = new DAO\_users();

if ($DAOObject->checkLogin($loginForm->getValue('username'),

$loginForm->getValue('password'))) {

//Create Session

$users = $DAOObject->authenticate($loginForm->getValue('username'),

$loginForm->getValue('password'));

if ($users) {

$this->DAOSession->create($users);

}

//Redirect User based on their usertype

//if unknown then destroy session

switch ($this->DAOSession->getUser()->getUserType()) {

case 'student':

$this->\_redirect('/student');

case 'teacher':

$this->\_redirect('/teacher');

case 'admin':

$this->\_redirect('/admin');

default:

$this->DAOSession->destroySession();

$this->view->errorMsg = "<br>Login Error<br>";

}

}

else {

$this->view->errorMsg = "<br>Incorrect Login<br>";

}

}

}

if ($tableItem != ""){

$this->view->newsTitle = "News";

}

$this->view->news = $tableItem;

$this->\_helper->layout->setLayout('login\_layout');

}

/\*\*

\*

\* Render a news item into a \_news-item table

\* @param models/entities/news $newsArticle

\* @return /index/\_news-item.phtml

\*/

private function renderNewsTable($newsArticle)

{

$tableItem = new Zend\_View();

$tableItem->setScriptPath($this->view->getScriptPaths());

$tableItem->newsId = $newsArticle->getNewsid();

$tableItem->title = $newsArticle->getTitle();

$tableItem->article = $newsArticle->getArticle();

$tableItem->publishedDate = $newsArticle->getPublishedDate();

return $tableItem->render('/index/\_news-item.phtml');

}

}

###### RequestController

<?php

/\*\*

\* RequestController

\*

\* @author Aaron Redmond

\* @version 1.0

\* @created 06-Apr-2011

\*/

require\_once APPLICATION\_PATH . '/modules/AbstractController.php';

use models\entities\users;

class RequestController extends AbstractController

{

/\*\*

\*

\* Request Username/Passsword

\*

\* If the user emters a valid email address

\* generate a new password and email the

\* details to the supplied email address.

\*/

public function indexAction ()

{

$requestForm = new Form\_Request();

if ($this->\_request->isPost() && $requestForm->isValid($\_POST)) {

$email = $requestForm->getValue('email');

$DAOusers = new DAO\_users();

$userDetails = $DAOusers->retrieveUserByUserEmail($email);

if ($userDetails != null){

$userName = $userDetails->getUserName();

//Generate New Password

$password = $this->generatePassword();

//Set New Password

$userDetails->setUserPassword(md5($password));

//Email Details

$bodyText = 'Your login details are:' . PHP\_EOL;

$bodyText .= 'Username: ' . $userName . PHP\_EOL;

$bodyText .= 'Password: ' . $password . PHP\_EOL;

$bodyText .= PHP\_EOL;

$bodyText .= 'http://' . $\_SERVER['SERVER\_NAME'] . '/Login' . PHP\_EOL;

$recipient = $userDetails->getUserForename() . ' ' . $userDetails->getUserSurname();

$subject = 'Requested Details';

$mail = new Zend\_Mail();

$mail->setBodyText($bodyText);

$mail->setFrom();

$mail->addTo($email, $recipient);

$mail->setSubject($subject);

try {

$mail->send();

} catch (Exception $e) {

$this->view->errorMessage = 'Unable to send email. Please contact the system administrator.';

$messageFailure = true;

}

} else {

$this->view->errorMessage = "The system cannot find a user with that email address.";

}

}

$this->view->requestForm = $requestForm;

$this->\_helper->layout->setLayout('request\_layout');

}

/\*\*

\*

\* Generate a Random Password

\* @return string $password

\*/

private function generatePassword ()

{

//Start with a blank Password

$password = "";

//Generate Random Password Length

$passwordLength = mt\_rand(6, 32);

//Define Possible characters

$possible = "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";

$counter = 0;

//Generate Password

while ($counter < $passwordLength) {

$char = substr($possible, mt\_rand(0, strlen($possible) - 1), 1);

if (!strstr($password, $char)){

$password .= $char;

$counter++;

}

}

return $password;

}

}

##### Views

###### \_news-item.phtml

<table id=*"News-Item"*>

<tr>

<td>

<b><?php **echo** $this->title;?></b>

|

<?php **echo** $this->publishedDate ;?>

</td>

</tr>

<tr>

<td>

<?php **echo** nl2br($this->article);?>

</td>

</tr>

</table>

<br>

###### index.phtml

<div id=*"loginForm"*>

<?php **echo** $this->loginForm; ?>

</div>

#### Student

##### Controllers

###### IndexController

<?php

/\*\*

\* Student IndexController

\*

\* @author Aaron Redmond

\* @version 1.0

\*/

require\_once APPLICATION\_PATH . '/modules/AbstractController.php';

use models\entities\news;

class Student\_IndexController extends AbstractController

{

/\*\*

\*Check to see if the user is logged in and of the correct user type

\*This function is applied to all pages

\*/

public function init ()

{

if (! ($this->DAOSession->isLoggedIn())) {

$this->\_redirect('/Login');

}

if (!($this->DAOSession->getUser()->getUserType() == 'student')){

$this->\_redirect('/Login');

}

}

/\*\*

\* The default action - show the Admin home page

\*/

public function indexAction ()

{

$userEntity = $this->DAOSession->getUser();

$userName = $userEntity->getUserForename();

$this->view->userName = $userName;

//Render news items

$DAOnews = new DAO\_news();

$newsItems = $DAOnews->retrieveNewsItems();

$tableItem = "";

foreach ($newsItems as $newsItem){

$tableItem .= $this->renderNewsTable($newsItem);

}

$this->view->news = $tableItem;

$this->\_helper->layout->setLayout('/student/student\_layout');

}

/\*\*

\*

\* Render a news item into a \_news-item table

\* @param models/entities/news $newsArticle

\* @return /index/\_news-item.phtml

\*/

private function renderNewsTable($newsArticle)

{

$tableItem = new Zend\_View();

$tableItem->setScriptPath($this->view->getScriptPaths());

$tableItem->newsId = $newsArticle->getNewsid();

$tableItem->title = $newsArticle->getTitle();

$tableItem->article = $newsArticle->getArticle();

return $tableItem->render('/index/\_news-item.phtml');

}

/\*\*

\* Destroy Session and redirect user to login page

\*/

public function logoutAction()

{

$this->DAOSession->destroySession();

$this->\_redirect('/Login');

}

}

###### myaccountController

<?php

/\*\*

\* Student myaccountController

\*

\* @author Aaron Redmond

\* @version 1.0

\*/

require\_once APPLICATION\_PATH . '/modules/AbstractController.php';

use models\entities\users;

class Student\_myaccountController extends AbstractController

{

/\*\*

\*Check to see if the user is logged in and of the correct user type

\*This function is applied to all pages

\*/

public function init ()

{

if (! ($this->DAOSession->isLoggedIn())) {

$this->\_redirect('/Login');

}

if (!($this->DAOSession->getUser()->getUserType() == 'student')){

$this->\_redirect('/Login');

}

}

/\*\*

\* The default action - show the My Account page

\*/

public function indexAction ()

{

$this->\_helper->layout->setLayout('/student/student\_layout');

}

/\*\*

\*

\* Change username page

\* Allow the student to change their login username

\*/

public function usernameAction()

{

$usernameForm = new Form\_Admin\_myaccount\_username();

$user = $this->DAOSession->getUser();

if ($this->\_request->isPost() && $usernameForm->isValid($\_POST)) {

$values = $usernameForm->getValues();

$newUsername = $values['newUsername'];

$DAOusers = new DAO\_users();

$usernameexists = $DAOusers->retrieveUserByUsername($newUsername);

if ($usernameexists == null) {

$DAOusers->updateUserName($user, $newUsername);

$this->view->errorMessage = "Username successfully changed";

}else {

$this->view->errorMessage = "Username already exists";

}

}

$this->view->usernameForm = $usernameForm;

$this->\_helper->layout->setLayout('/student/student\_layout');

}

/\*\*

\*

\* Change password page

\* Allow student to change their login password

\*/

public function passwordAction()

{

$passwordForm = new Form\_Admin\_myaccount\_password();

if ($this->\_request->isPost() && $passwordForm->isValid($\_POST)) {

$values = $passwordForm->getValues();

$oldPassword = $values['oldPassword'];

$newPassword = $values['newPassword'];

$verifyNewPassword = $values['verifyNewPassword'];

$user = $this->DAOSession->getUser();

$existingPassword = $user->getUserPassword();

if (md5($oldPassword) == $existingPassword) {

if ($newPassword == $verifyNewPassword) {

$DAOusers = new DAO\_users();

$DAOusers->updateUserPassword($user, md5($newPassword));

$this->view->errorMessage = "Password successfully changed";

}else {

$this->view->errorMessage = "New password and verify new password must match";

}

} else {

$this->view->errorMessage = "Incorrect password entered.";

}

}

$this->view->passwordForm = $passwordForm;

$this->\_helper->layout->setLayout('/student/student\_layout');

}

}

###### timetableController

<?php

/\*\*

\* Student timetableController

\*

\* @author Aaron Redmond

\* @version 1.0

\*/

require\_once APPLICATION\_PATH . '/modules/AbstractController.php';

use models\entities\users;

use models\entities\groups;

class Student\_timetableController extends AbstractController

{

/\*\*

\*Check to see if the user is logged in and of the correct user type

\*This function is applied to all pages

\*/

public function init ()

{

if (! ($this->DAOSession->isLoggedIn())) {

$this->\_redirect('/Login');

}

if (!($this->DAOSession->getUser()->getUserType() == 'student')){

$this->\_redirect('/Login');

}

}

/\*\*

\*

\* Show Group Timetable for student if available

\*/

public function timetableAction ()

{

//Get groups associated with student

$student = $this->DAOSession->getUser();

$groups = $student->getGroups();

$grouplist = "";

foreach ($groups as $group){

$grouplist .= $this->renderGroupList($group);

}

$id = $this->\_getParam('id');

if ($id != null){

foreach ($groups as $group){

if ($group->getGroupId() == $id){

$groupName = $group->getGroupName();

$groupName = str\_replace(' ', '\_', $groupName);

$timetable = $this->loadTimetable('groups', $groupName);

}

}

} else {

if (sizeof($groups) > 0){

$firstGroup = $groups[0];

$groupName = $firstGroup->getGroupName();

$groupName = str\_replace(' ', '\_', $groupName);

$timetable = $this->loadTimetable('groups', $groupName);

} else {

$timetable = null;

}

}

//If there is a timetable available

if ($timetable != null){

$this->view->timetable = $timetable;

$this->view->printTimetable = new Form\_Admin\_timetabling\_printbutton($timetable);

}else{

$this->view->errorMessage = 'There is no timetable available at this time';

$this->view->printTimetable = null;

}

$this->view->groupList = $grouplist;

$this->\_helper->layout->setLayout('/student/student\_layout');

}

/\*\*

\* Renders a group into a \_timetable-subtype-item list template

\* @param models\entities\groups $group

\*/

private function renderGroupList (groups $group)

{

$tableItem = new Zend\_View();

$tableItem->setScriptPath($this->view->getScriptPaths());

$tableItem->Id = $group->getGroupId();

$tableItem->name = $group->getGroupName();

$tableItem->type = "group";

return $tableItem->render('/timetable/\_timetable-subtype-item.phtml');

}

/\*\*

\*

\* Load Timetable file and pass back timetable html

\* @param string $folderName

\* @param string $fileName

\*/

private function loadTimetable($folderName, $fileName)

{

$rootPath = $\_SERVER['DOCUMENT\_ROOT'];

$filename = $rootPath . '/fet/saved/' . $folderName . '/' . $fileName . '.html';

if (file\_exists($filename)) {

try {

$fp = fopen($filename, 'r');

$timetable = fread($fp, filesize($filename));

fclose($fp);

} catch (Exception $e) {

return false;

}

} else {

return null;

}

return $timetable;

}

}

##### Views

###### Index

\_news-item.phtml

<table id=*"News-Item"*>

<tr>

<td>

<b><?php **echo** $this->title;?></b>

</td>

</tr>

<tr>

<td>

<?php **echo** nl2br($this->article);?>

</td>

</tr>

</table>

<br>

index.phtml

<p>Welcome <?php **echo** $this->userName; ?></p>

<br>

<p><b>News</b></p>

<br>

<?php **echo** $this->news; ?>

###### Myaccount

Index.phtml

<p>My Account</p>

<br>

<p>Click on a link bellow to navigate to that section</p>

<br>

<a href=*"/student/myaccount/username/"*>Change Username</a>

<br>

<a href=*"/student/myaccount/password/"*>Change Password</a>

Password.phtml

<p>Change Password</p>

<?php **echo** $this->passwordForm;?>

<?php **echo** $this->errorMessage;?>

username.phtml

<p>Change Username</p>

<?php **echo** $this->usernameForm; ?>

<?php **echo** $this->errorMessage; ?>

###### timetable

\_timetable-subtype-item.phtml

<li>

<a href="*/student/timetable/timetable/id/*<?php **echo** $this->Id; ?>"><?php **echo** $this->name; ?></a>

</li>

timetable.phtml

<script type=*"text/javascript"*>

**function** printTT(data){

myWindow = window.open('','');

myWindow.document.write(data);

}

</script>

<p>Timetable</p>

<br>

<div id=*"tt-selector"*>

<?php **echo** $this->groupList; ?>

</div>

<br>

<br>

<div id=*"center\_timetable"*>

<?php **echo** $this->timetable; ?>

</div>

<div id=*"center\_b"*>

<?php **echo** $this->printTimetable; ?>

</div>

<br>

<?php **echo** $this->errorMessage; ?>

#### Teacher

##### Controllers

###### IndexController.php

<?php

/\*\*

\* Teacher IndexController

\*

\* @author Aaron Redmond

\* @version 1.0

\*/

require\_once APPLICATION\_PATH . '/modules/AbstractController.php';

use models\entities\news;

class Teacher\_IndexController extends AbstractController

{

/\*\*

\*Check to see if the user is logged in and of correct user type

\*This function is applied to all pages

\*/

public function init ()

{

if (! ($this->DAOSession->isLoggedIn())) {

$this->\_redirect('/Login');

}

if (!($this->DAOSession->getUser()->getUserType() == 'teacher')){

$this->\_redirect('/Login');

}

}

/\*\*

\* The default action - show the Teacher home page

\*/

public function indexAction ()

{

$userEntity = $this->DAOSession->getUser();

$userName = $userEntity->getUserForename();

$this->view->userName = $userName;

//Render news items

$DAOnews = new DAO\_news();

$newsItems = $DAOnews->retrieveNewsItems();

$tableItem = "";

foreach ($newsItems as $newsItem){

$tableItem .= $this->renderNewsTable($newsItem);

}

$this->view->news = $tableItem;

$this->\_helper->layout->setLayout('/Teacher/teacher\_layout');

}

/\*\*

\*

\* Render a news item into a \_news-item table

\* @param models/entities/news $newsArticle

\* @return /index/\_news-item.phtml

\*/

private function renderNewsTable($newsArticle)

{

$tableItem = new Zend\_View();

$tableItem->setScriptPath($this->view->getScriptPaths());

$tableItem->newsId = $newsArticle->getNewsid();

$tableItem->title = $newsArticle->getTitle();

$tableItem->article = $newsArticle->getArticle();

return $tableItem->render('/index/\_news-item.phtml');

}

/\*\*

\* Destroy Session and redirect user to login page

\*/

**public function** helpAction()

{

$this->\_helper->layout->setLayout('/student/student\_layout');

}

/\*\*

\*

\* Destroy Session and redirect user to login page

\*/

public function logoutAction()

{

$this->DAOSession->destroySession();

$this->\_redirect('/Login');

}

}

###### myaccountController.php

<?php

/\*\*

\* Teacher myaccountController

\*

\* @author Aaron Redmond

\* @version 1.0

\*/

require\_once APPLICATION\_PATH . '/modules/AbstractController.php';

use models\entities\users;

class Teacher\_myaccountController extends AbstractController

{

/\*\*

\*Check to see if the user is logged in and of correct user type

\*This function is applied to all pages

\*/

public function init ()

{

if (! ($this->DAOSession->isLoggedIn())) {

$this->\_redirect('/Login');

}

if (!($this->DAOSession->getUser()->getUserType() == 'teacher')){

$this->\_redirect('/Login');

}

}

/\*\*

\* The default action - show the My Account page

\*/

public function indexAction ()

{

$this->\_helper->layout->setLayout('/Teacher/teacher\_layout');

}

/\*\*

\*

\* Change username page

\* Allow the teacher to change their login username

\*/

public function usernameAction()

{

$usernameForm = new Form\_Admin\_myaccount\_username();

$user = $this->DAOSession->getUser();

if ($this->\_request->isPost() && $usernameForm->isValid($\_POST)) {

$values = $usernameForm->getValues();

$newUsername = $values['newUsername'];

$DAOusers = new DAO\_users();

$usernameexists = $DAOusers->retrieveUserByUsername($newUsername);

if ($usernameexists == null) {

$DAOusers->updateUserName($user, $newUsername);

$this->view->errorMessage = "Username successfully changed";

}else {

$this->view->errorMessage = "Username already exists";

}

}

$this->view->usernameForm = $usernameForm;

$this->\_helper->layout->setLayout('/Teacher/teacher\_layout');

}

/\*\*

\*

\* Change password page

\* Allow teacher to change their login password

\*/

public function passwordAction()

{

$passwordForm = new Form\_Admin\_myaccount\_password();

if ($this->\_request->isPost() && $passwordForm->isValid($\_POST)) {

$values = $passwordForm->getValues();

$oldPassword = $values['oldPassword'];

$newPassword = $values['newPassword'];

$verifyNewPassword = $values['verifyNewPassword'];

$user = $this->DAOSession->getUser();

$existingPassword = $user->getUserPassword();

if (md5($oldPassword) == $existingPassword) {

if ($newPassword == $verifyNewPassword) {

$DAOusers = new DAO\_users();

$DAOusers->updateUserPassword($user, md5($newPassword));

$this->view->errorMessage = "Password successfully changed";

}else {

$this->view->errorMessage = "New password and verify new password must match";

}

} else {

$this->view->errorMessage = "Incorrect password entered.";

}

}

$this->view->passwordForm = $passwordForm;

$this->\_helper->layout->setLayout('/Teacher/teacher\_layout');

}

}

###### timetableController.php

<?php

/\*\*

\* Teacher timetableController

\*

\* @author Aaron Redmond

\* @version 1.0

\*/

require\_once APPLICATION\_PATH . '/modules/AbstractController.php';

class Teacher\_timetableController extends AbstractController

{

/\*\*

\*Check to see if the user is logged in and of correct user type

\*This function is applied to all pages

\*/

public function init ()

{

if (! ($this->DAOSession->isLoggedIn())) {

$this->\_redirect('/Login');

}

if (!($this->DAOSession->getUser()->getUserType() == 'teacher')){

$this->\_redirect('/Login');

}

}

/\*\*

\*

\* Show Teacher Timetable if available

\*/

public function timetableAction ()

{

$teacher = $this->DAOSession->getUser();

$timetableName = $teacher->getUserForename() . '\_' . $teacher->getUserSurname();

$timetable = $this->loadTimetable('teachers', $timetableName);

if ($timetable != null){

$this->view->timetable = $timetable;

$this->view->printTimetable = new Form\_Admin\_timetabling\_printbutton($timetable);

}else{

$this->view->errorMessage = 'There is no timetable available at this time';

$this->view->printTimetable = null;

}

$this->\_helper->layout->setLayout('/Teacher/teacher\_layout');

}

/\*\*

\*

\* Load Timetable file and pass back timetable html

\* @param string $folderName

\* @param string $fileName

\* @return string $timetable || Null

\*/

private function loadTimetable($folderName, $fileName)

{

$rootPath = $\_SERVER['DOCUMENT\_ROOT'];

$filename = $rootPath . '/fet/saved/' . $folderName . '/' . $fileName . '.html';

if (file\_exists($filename)) {

try {

$fp = fopen($filename, 'r');

$timetable = fread($fp, filesize($filename));

fclose($fp);

} catch (Exception $e) {

return false;

}

} else {

return null;

}

return $timetable;

}

}

##### Views

###### Index

Help.phtml

<p>Help</p>

<br>

<p>Welcome to the help section. Just click on a topic to view.</p>

<br>

<ul>

<li>

<a href=*"#Timetable"*>Timetable</a>

</li>

<li>

<a href=*"#UserPanel"*>User Panel - My Account</a>

<ul>

<li>

<a href=*"#MyAccount"*>My Account</a>

<ul>

<li>

<a href=*"#Username"*>Change Username</a>

</li>

<li>

<a href=*"#Password"*>Change Password</a>

</li>

</ul>

</li>

</ul>

</li>

</ul>

<br>

<b><a name=*"Timetable"*>Timetable</a></b>

<br>

<p>

This is where you can view your timetables. Simply click on a timetable to display it.

Click on the save button to open the timetable in a new window, which you may them print or save.

</p>

<br>

<b><a name=*"UserPanel"*>User Panel - My Account</a></b>

<br>

<p>

From here you can may select the My Account option where you can change your username or password.

</p>

<br>

<b><a name=*"Username"*>Change Username</a></b>

<br>

<p>

To change your username, simply enter the new username you wish to use, if its not already taken its yours.

</p>

<br>

<b><a name=*"Password"*>Change Password</a></b>

<br>

<p>

To change your password, you must enter your current password,

the new password you wish to use, followed by verification of the new password you wish to use.

</p>

\_news-item.phtml

<table id=*"News-Item"*>

<tr>

<td>

<b><?php **echo** $this->title;?></b>

</td>

</tr>

<tr>

<td>

<?php **echo** nl2br($this->article);?>

</td>

</tr>

</table>

<br>

index.phtml

<p>Welcome <?php **echo** $this->userName; ?></p>

<br>

<p><b>News</b></p>

<br>

<?php **echo** $this->news; ?>

###### Myaccount

Index.phtml

<p>My Account</p>

<br>

<p>Click on a link bellow to navigate to that section</p>

<br>

<a href=*"/teacher/myaccount/username/"*>Change Username</a>

<br>

<a href=*"/teacher/myaccount/password/"*>Change Password</a>

Password.phtml

<p>Change Password</p>

<?php **echo** $this->passwordForm;?>

<?php **echo** $this->errorMessage;?>

username.phtml

<p>Change Username</p>

<?php **echo** $this->usernameForm; ?>

<?php **echo** $this->errorMessage; ?>

###### Timetable

Timetable.phtml

<script type=*"text/javascript"*>

**function** printTT(data){

myWindow = window.open('','');

myWindow.document.write(data);

}

</script>

<p>Timetable</p>

<br>

<div id=*"center\_timetable"*>

<?php **echo** $this->timetable; ?>

</div>

<div id=*"center\_b"*>

<?php **echo** $this->printTimetable; ?>

</div>

<?php **echo** $this->errorMessage; ?>

#### AbstractController

<?php

use models\entities\users;

use models\entities\schoolinfo;

abstract class AbstractController extends Zend\_Controller\_Action

{

protected $DAOSession;

protected $config;

protected $image;

public function \_\_construct (Zend\_Controller\_Request\_Abstract $request,

Zend\_Controller\_Response\_Abstract $response, array $invokeArgs = array())

{

$this->DAOSession = new DAO\_Session();

$this->config = Zend\_Registry::get('config');

parent::\_\_construct($request, $response, $invokeArgs);

$DAOschoolInfo = new DAO\_schoolinfo();

$schoolInfo = $DAOschoolInfo->retriveSchoolInfo();

if ($schoolInfo != null){

$image = $schoolInfo->getSchoolimage();

if ($image == null){

$image = 'msm\_banner.jpg';

}

$this->\_helper->layout()->image = $image;

} else {

$image = 'msm\_banner.jpg';

$this->\_helper->layout()->image = $image;

}

}

}

?>

### Validators

The following classes are used for form validation.

#### Abstract.php

<?php

abstract class Validate\_Abstract extends Zend\_Validate\_Abstract

{

protected function \_setError (array $errors)

{

if (sizeof($errors)) {

$this->\_error($errors[0]);

} else {

$this->\_errors("INVALID");

}

}

/\*\*

\*

\* Get the first error message

\* @return string | null

\*/

public function getFirstMessage ()

{

if (sizeof($this->\_messages)) {

$messages = array\_values($this->\_messages);

return $messages[0];

}

return null;

}

}

?>

#### ConfirmPassword.php

<?php

/\*\*

\*

\* Validate Confirmed Password against password

\* @author Aaron Redmond

\* @version 1.0

\* @created 14-Feb-2011

\*/

class Validate\_ConfirmPassword extends Validate\_Abstract

{

const NOT\_MATCH = 'notMatch';

protected $\_messageTemplates = array(self::NOT\_MATCH => 'Passwords do not match');

public function isValid ($value, $context = null)

{

if (is\_array($context))

{

if (!(isset($context['confirmpassword']) && ($value == $context['password'])))

{

$this->\_error(self::NOT\_MATCH);

return false;

}

}

return true;

}

}

?>

#### Email.php

<?php

/\*\*

\*

\* Validate email string

\* @author Aaron Redmond

\* @version 1.0

\* @created 14-Feb-2011 00:37:07

\*/

class Validate\_Email extends Validate\_Abstract

{

const INVALID = 'stringLengthInvalid';

const IS\_EMPTY = 'isEmpty';

/\*\*

\* @var array

\*/

protected $\_messageTemplates = array(self::INVALID => "Please enter a valid email address",

self::IS\_EMPTY => "Please enter your email");

public function isValid ($value)

{

$ValidateNonEmpty = new Zend\_Validate\_NotEmpty(

Zend\_Validate\_NotEmpty::STRING);

if (! $ValidateNonEmpty->isValid($value))

{

$this->\_setError($ValidateNonEmpty->getErrors());

return false;

}

$ValidateEmail = new Zend\_Validate\_EmailAddress();

if(!$ValidateEmail->isValid($value))

{

$this->\_setError(array(self::INVALID));

return false;

}

return true;

}

}

?>

#### EmailUnique.php

<?php

/\*\*

\*

\* Checks database to see if Email already in use

\* @author Aaron Redmond

\* @version 1.0

\* @created 14-Feb-2011 00:37:07

\*/

class Validate\_EmailUnique extends Validate\_Abstract

{

const INVALID = 'emailAlreadyExists';

protected $\_messageTemplates = array(self::INVALID => "A user with this email alreay exists.");

public function isValid ($value)

{

$DAOusers = new DAO\_users();

$emailExists = $DAOusers->retrieveUserByUserEmail($value);

if ($emailExists != null){

$this->\_setError(array('emailAlreadyExists'));

return false;

}

return true;

}

}

?>

#### Password.php

<?php

/\*\*

\*

\* Check For Empty String/Invalid Characters/Length

\* @author Aaron Redmond

\* @version 1.0

\* @created 09-Feb-2011

\*/

class Validate\_Password extends Validate\_Abstract

{

const INVALID = 'stringLengthInvalid';

const TOO\_SHORT = 'stringLengthTooShort';

const TOO\_LONG = 'stringLengthTooLong';

const IS\_EMPTY = 'isEmpty';

private $minLength = 6;

private $maxLength = 20;

/\*\*

\* @var array

\*/

protected $\_messageTemplates = array(self::INVALID => "Invalid password",

self::TOO\_SHORT => "Password is too short",

self::TOO\_LONG => "Password is too long",

self::IS\_EMPTY => "Please enter your password");

/\* (non-PHPdoc)

\* @see Zend\_Validate\_Interface::isValid()

\*/

public function isValid ($value)

{

$ValidateNonEmpty = new Zend\_Validate\_NotEmpty(

Zend\_Validate\_NotEmpty::STRING);

if (! $ValidateNonEmpty->isValid($value)) {

$this->\_setError($ValidateNonEmpty->getErrors());

return false;

}

$StringLengthValidator = new Zend\_Validate\_StringLength($this->minLength,

$this->maxLength);

if (! $StringLengthValidator->isValid($value)) {

$this->\_setError($StringLengthValidator->getErrors());

return false;

}

return true;

}

}

?>

#### Time.php

<?php

/\*\*

\* Validate Time for Timetable Info

\*

\* @author Aaron Redmond

\* @version 1.0

\* @created 01-Mar-2011

\*

\*/

class Validate\_Time extends Validate\_Abstract

{

const NOT\_MATCH = 'notMatch';

const HOUR\_ERROR = 'hourError';

const MIN\_ERROR = 'minError';

protected $\_messageTemplates = array(self::NOT\_MATCH => 'Incorrect time format, correct format is HH:MM',

self::HOUR\_ERROR => 'Incorrect value, the hour value should be between 00 and 12',

self::MIN\_ERROR => 'Incorrect value, the minutes value should be between 00 and 59');

public function isValid ($value, $context=null)

{

$valueArray = preg\_split("/:/", $value);

//If 8 was entered make it 8:00

while (sizeof($valueArray) < 2) {

$valueArray[] = '00';

}

//if the value is not now in the format XX:XX

if (sizeof($valueArray) > 2){

$this->\_error(self::NOT\_MATCH);

return false;

}

//If the hour value is greater than 12

if ($valueArray[0] > 12){

$this->\_error(self::HOUR\_ERROR);

return false;

}

//If the minute value is greater than 59

if ($valueArray[1] > 59){

$this->\_error(self::MIN\_ERROR);

return false;

}

//Ensure Hours and minutes are actually numbers

foreach ($valueArray as $v){

if (!(is\_numeric($v))){

$this->\_error(self::NOT\_MATCH);

return false;

}

}

return true;

}

}

?>

#### User.php

<?php

/\*\*

\*

\* Check For Empty String/Invalid Characters/Length

\* @author Aaron Redmond

\* @version 1.0

\* @created 09-Feb-2011

\*/

class Validate\_User extends Validate\_Abstract

{

const INVALID = 'stringLengthInvalid';

const TOO\_SHORT = 'stringLengthTooShort';

const TOO\_LONG = 'stringLengthTooLong';

const IS\_EMPTY = 'isEmpty';

const NOT\_MATCH = 'regexNotMatch';

private $minLength = 6;

private $maxLength = 20;

/\*\*

\* @var array

\*/

protected $\_messageTemplates = array(self::INVALID => "Invalid user name",

self::TOO\_SHORT => "User name is too short. It should be between 6 and 20 characters.",

self::TOO\_LONG => "User name is too long. It should be between 6 and 20 characters.",

self::IS\_EMPTY => "Please enter your User name",

self::NOT\_MATCH => "Invalid symbol in user name");

/\* (non-PHPdoc)

\* @see Zend\_Validate\_Interface::isValid()

\*/

public function isValid ($value)

{

$ValidateNonEmpty = new Zend\_Validate\_NotEmpty(

Zend\_Validate\_NotEmpty::STRING);

if (! $ValidateNonEmpty->isValid($value)) {

$this->\_setError($ValidateNonEmpty->getErrors());

return false;

}

$CharValidator = new Zend\_Validate\_Regex( "/^[a-zA-Z0-9@.\_]/");

if (! $CharValidator->isValid($value)) {

$this->\_setError($CharValidator->getErrors());

return false;

}

$StringLengthValidator = new Zend\_Validate\_StringLength($this->minLength,

$this->maxLength);

if (! $StringLengthValidator->isValid($value)) {

$this->\_setError($StringLengthValidator->getErrors());

return false;

}

return true;

}

}

?>

#### Username.php

<?php

/\*\*

\* @author Aaron Redmond

\* @version 1.0

\* @created 30-Mar-2011

\* @example

\* Check For Empty String </br>

\* Check For Invalid Characters </br>

\* Check Length between 6 - 20 Characters

\* Validate Unique username

\*/

use models\entities\users;

class Validate\_Username extends Validate\_Abstract

{

const INVALID = 'stringLengthInvalid';

const TOO\_SHORT = 'stringLengthTooShort';

const TOO\_LONG = 'stringLengthTooLong';

const IS\_EMPTY = 'isEmpty';

const NOT\_MATCH = 'regexNotMatch';

const EXISTS = 'usernameExists';

private $minLength = 6;

private $maxLength = 20;

/\*\*

\* @var array

\*/

protected $\_messageTemplates = array(self::INVALID => "Invalid user name",

self::TOO\_SHORT => "User name is too short. It should be between 6 and 20 characters.",

self::TOO\_LONG => "User name is too long. It should be between 6 and 20 characters.",

self::IS\_EMPTY => "Please enter your User name",

self::NOT\_MATCH => "Invalid symbol in user name",

self::EXISTS => "That username is already taken.");

public function isValid ($value)

{

$ValidateNonEmpty = new Zend\_Validate\_NotEmpty(

Zend\_Validate\_NotEmpty::STRING);

if (! $ValidateNonEmpty->isValid($value)) {

$this->\_setError($ValidateNonEmpty->getErrors());

return false;

}

$CharValidator = new Zend\_Validate\_Regex( "/^[a-zA-Z0-9@.\_]/");

if (! $CharValidator->isValid($value)) {

$this->\_setError($CharValidator->getErrors());

return false;

}

$StringLengthValidator = new Zend\_Validate\_StringLength($this->minLength,

$this->maxLength);

if (! $StringLengthValidator->isValid($value)) {

$this->\_setError($StringLengthValidator->getErrors());

return false;

}

$DAOusers = new DAO\_users();

$usernameExists = $DAOusers->retrieveUserByUsername($value);

if ($usernameExists != null){

$this->\_setError(array('usernameExists'));

return false;

}

return true;

}

}

?>

## Public

### Js

The next piece of code is an opensource piece of jquery which allow me to implement the graphic slider used on the personalisation page.

#### easySlider1.7.js

/\*

\* Easy Slider 1.7 - jQuery plugin

\* written by Alen Grakalic

\* http://cssglobe.com/post/4004/easy-slider-15-the-easiest-jquery-plugin-for-sliding

\*

\* Copyright (c) 2009 Alen Grakalic (http://cssglobe.com)

\* Dual licensed under the MIT (MIT-LICENSE.txt)

\* and GPL (GPL-LICENSE.txt) licenses.

\*

\* Built for jQuery library

\* http://jquery.com

\*

\*/

/\*

\* markup example for $("#slider").easySlider();

\*

\* <div id="slider">

\* <ul>

\* <li><img src="images/01.jpg" alt="" /></li>

\* <li><img src="images/02.jpg" alt="" /></li>

\* <li><img src="images/03.jpg" alt="" /></li>

\* <li><img src="images/04.jpg" alt="" /></li>

\* <li><img src="images/05.jpg" alt="" /></li>

\* </ul>

\* </div>

\*

\*/

(function($) {

$.fn.easySlider = function(options){

// default configuration properties

var defaults = {

prevId: 'prevBtn',

prevText: 'Previous',

nextId: 'nextBtn',

nextText: 'Next',

controlsShow: true,

controlsBefore: '',

controlsAfter: '',

controlsFade: true,

firstId: 'firstBtn',

firstText: 'First',

firstShow: false,

lastId: 'lastBtn',

lastText: 'Last',

lastShow: false,

vertical: false,

speed: 800,

auto: false,

pause: 2000,

continuous: false,

numeric: false,

numericId: 'controls'

};

var options = $.extend(defaults, options);

this.each(function() {

var obj = $(this);

var s = $("li", obj).length;

var w = $("li", obj).width();

var h = $("li", obj).height();

var clickable = true;

obj.width(w);

obj.height(h);

obj.css("overflow","hidden");

var ts = s-1;

var t = 0;

$("ul", obj).css('width',s\*w);

if(options.continuous){

$("ul", obj).prepend($("ul li:last-child", obj).clone().css("margin-left","-"+ w +"px"));

$("ul", obj).append($("ul li:nth-child(2)", obj).clone());

$("ul", obj).css('width',(s+1)\*w);

};

if(!options.vertical) $("li", obj).css('float','left');

if(options.controlsShow){

var html = options.controlsBefore;

if(options.numeric){

html += '<ol id="'+ options.numericId +'"></ol>';

} else {

if(options.firstShow) html += '<span id="'+ options.firstId +'"><a href=\"javascript:void(0);\">'+ options.firstText +'</a></span>';

html += ' <span id="'+ options.prevId +'"><a href=\"javascript:void(0);\">'+ options.prevText +'</a></span>';

html += ' <span id="'+ options.nextId +'"><a href=\"javascript:void(0);\">'+ options.nextText +'</a></span>';

if(options.lastShow) html += ' <span id="'+ options.lastId +'"><a href=\"javascript:void(0);\">'+ options.lastText +'</a></span>';

};

html += options.controlsAfter;

$(obj).after(html);

};

if(options.numeric){

for(var i=0;i<s;i++){

$(document.createElement("li"))

.attr('id',options.numericId + (i+1))

.html('<a rel='+ i +' href=\"javascript:void(0);\">'+ (i+1) +'</a>')

.appendTo($("#"+ options.numericId))

.click(function(){

animate($("a",$(this)).attr('rel'),true);

});

};

} else {

$("a","#"+options.nextId).click(function(){

animate("next",true);

});

$("a","#"+options.prevId).click(function(){

animate("prev",true);

});

$("a","#"+options.firstId).click(function(){

animate("first",true);

});

$("a","#"+options.lastId).click(function(){

animate("last",true);

});

};

function setCurrent(i){

i = parseInt(i)+1;

$("li", "#" + options.numericId).removeClass("current");

$("li#" + options.numericId + i).addClass("current");

};

function adjust(){

if(t>ts) t=0;

if(t<0) t=ts;

if(!options.vertical) {

$("ul",obj).css("margin-left",(t\*w\*-1));

} else {

$("ul",obj).css("margin-left",(t\*h\*-1));

}

clickable = true;

if(options.numeric) setCurrent(t);

};

function animate(dir,clicked){

if (clickable){

clickable = false;

var ot = t;

switch(dir){

case "next":

t = (ot>=ts) ? (options.continuous ? t+1 : ts) : t+1;

break;

case "prev":

t = (t<=0) ? (options.continuous ? t-1 : 0) : t-1;

break;

case "first":

t = 0;

break;

case "last":

t = ts;

break;

default:

t = dir;

break;

};

var diff = Math.abs(ot-t);

var speed = diff\*options.speed;

if(!options.vertical) {

p = (t\*w\*-1);

$("ul",obj).animate(

{ marginLeft: p },

{ queue:false, duration:speed, complete:adjust }

);

} else {

p = (t\*h\*-1);

$("ul",obj).animate(

{ marginTop: p },

{ queue:false, duration:speed, complete:adjust }

);

};

if(!options.continuous && options.controlsFade){

if(t==ts){

$("a","#"+options.nextId).hide();

$("a","#"+options.lastId).hide();

} else {

$("a","#"+options.nextId).show();

$("a","#"+options.lastId).show();

};

if(t==0){

$("a","#"+options.prevId).hide();

$("a","#"+options.firstId).hide();

} else {

$("a","#"+options.prevId).show();

$("a","#"+options.firstId).show();

};

};

if(clicked) clearTimeout(timeout);

if(options.auto && dir=="next" && !clicked){;

timeout = setTimeout(function(){

animate("next",false);

},diff\*options.speed+options.pause);

};

};

};

// init

var timeout;

if(options.auto){;

timeout = setTimeout(function(){

animate("next",false);

},options.pause);

};

if(options.numeric) setCurrent(0);

if(!options.continuous && options.controlsFade){

$("a","#"+options.prevId).hide();

$("a","#"+options.firstId).hide();

};

});

};

})(jQuery);

### styles

The following CSS files control the lookof the site.

#### menu.css

@CHARSET "ISO-8859-1";

**ul**.topnav {

list-style: none;

padding: 0 20px;

margin: 0;

float: left;

width: 920px;

background: #222;

font-size: 1.2em;

background: <img alt="" src="images/topnav\_bg.gif"> repeat-x;

}

**ul**.topnav **li** {

float: left;

margin: 0;

padding: 0 15px 0 0;

position: relative; /\*--Declare X and Y axis base--\*/

}

**ul**.topnav **li** **a** {

padding: 10px 5px;

color: #fff;

display: block;

text-decoration: none;

float: left;

}

**ul**.topnav **li** **a**:hover {

background: <img alt="" src="images/topnav\_hover.gif"> no-repeat center top;

}

**ul**.topnav **li** **span** {

/\*--Drop down trigger styles--\*/

width: 17px;

height: 35px;

float: left;

background: <img alt="" src="images/subnav\_btn.gif"> no-repeat center top;

}

**ul**.topnav **li** **span**.subhover {

background-position: center bottom;

cursor: pointer;

}

/\*--Hover effect for trigger--\*/

**ul**.topnav **li** **ul**.subnav {

list-style: none;

/\*--Important - Keeps subnav from affecting main navigation flow--\*/

position: absolute;

left: 0;

top: 35px;

background: #333;

margin: 0;

padding: 0;

display: none;

float: left;

width: 170px;

-moz-border-radius-bottomleft: 5px;

-moz-border-radius-bottomright: 5px;

-webkit-border-bottom-left-radius: 5px;

-webkit-border-bottom-right-radius: 5px;

border: 1px solid #111;

}

**ul**.topnav **li** **ul**.subnav **li** {

margin: 0;

padding: 0;

border-top: 1px solid #252525;

/\*--Create bevel effect--\*/

border-bottom: 1px solid #444;

/\*--Create bevel effect--\*/

clear: both;

width: 170px;

}

**html** **ul**.topnav **li** **ul**.subnav **li** **a** {

float: left;

width: 145px;

background: #333 <img alt="" src="images/dropdown\_linkbg.gif"> no-repeat 10px center;

padding-left: 20px;

}

**html** **ul**.topnav **li** **ul**.subnav **li** **a**:hover {

/\*--Hover effect for subnav links--\*/

background: #222 <img alt="" src="images/dropdown\_linkbg.gif"> no-repeat 10px center;

}

#header **img** {

margin: 20px 0 10px;

}

**ul**.topnav **li** **ul**.subnav **li** **ul**.subnav2 {

list-style: none;

/\*--Important - Keeps subnav from affecting main navigation flow--\*/

position: absolute;

left: 170px;

top: 0px;

background: #333;

margin: 0;

padding: 0;

display: none;

float: left;

width: 170px;

border: 1px solid #111;

}

#### msmstyles.css

@CHARSET *"UTF-8"*;

/\*

Main Content

\*/

*#wrap* {

margin: *0* *auto*;

position: *relative*;

text-align: *left*;

width: *960px*;

border: *solid* *1px* *black*;

behavior: *url('/styles/PIE.htc')*;

border-radius: *10px*;

-moz-border-radius: *10px*;

-webkit-border-radius: *10px*;

color: *black*;

background-color: *#96B9FF*;

}

*#insidewrap* {

padding: *10px*;

font: *16px* *Helvetica,* *Sans-Serif*;

}

*#header* {

margin: *10px* *0* *10px* *0*;

width:*100%*;

position:*relative*;

float:*left*;

clear:*both*;

padding-top: *10px*;

}

*#footer* {

margin: *10px* *0* *10px* *0*;

border-top: *1px* *solid* *#ccc*;

width:*100%*;

position: *relative*;

clear:*both*;

text-align: *center*;

font-size: *x-small*;

}

/\*

Login Styles

\*/

**div***#outer* {

width: *50%*;

height: *350px*;

background-color:*#FFFFFF*;

margin-top: *50px*;

margin-bottom: *50px*;

margin-left: *auto*;

margin-right: *auto*;

padding: *0px*;

border: *thin* *solid* *#000000*;

}

**div***#leftside* {

width: *40%*;

padding: *10px*;

margin-top: *50px*;

float: *left*;

height:*80%*;

}

**div***#righside* {

margin-left: *50%*;

margin-top: *70px*;

padding: *10px*;

height:*80%*;

}

/\*

Rendered Tables styles

\*/

*#rendered\_table*{

border: *0*;

width: *100%*;

}

*#rendered\_table* **td**{

padding: *5*;

}

/\*

Timetable

\*/

*#center\_timetable*{

margin-left: *auto*;

margin-right: *auto*;

width: *60%*;

background-color: *white*;

}

*#center\_b*{

margin-left: *auto*;

margin-right: *auto*;

width: *20%*;

}

/\*

News Items

\*/

*#News-Item*{

border: *thin* *solid* *#000000*;

frame: *box*;

width: *100%*;

rules: *none*;

background-color: *white*;

}

/\*

Containers

\*/

*#the-border* {

background-color: *gray*;

padding: *10px*;

width: *30%*;

border: *solid* *1px* *black*;

}

*#login* {

margin-left: *auto*;

margin-right: *auto*;

margin-top: *150px*;

background-color: *gray*;

padding: *10px*;

width: *30%*;

border: *solid* *1px* *black*;

}

/\*

Timetable List

\*/

*#tt-selector*{

display: *inline*;

list-style-type: *none*;

}

*#tt-selector* **li** {

float:*left*;

margin-right:*10px*;

}

*#data* **ul** {

display: *inline*;

list-style-type: *none*;

}

*#data* **li** {

float:*left*;

margin-right:*10px*;

}

*#data* **ul** {

display:*none*;

}

/\*

Formatting for errors

\*/

**ul***.errors* {

list-style: *none*;

font: *13px* *Arial*;

color:*red*;

padding-left: *5px*;

}

#### PIE.htc

File used to render styles in IE.

#### Screen.css

/\* Easy Slider \*/

*#slider* **ul,** *#slider* **li**{

margin:*0*;

padding:*0*;

list-style:*none*;

}

*#slider* **li**{

/\*

define width and height of list item (slide)

entire slider area will adjust according to the parameters provided here

\*/

width:*796px*;

height:*241px*;

overflow:*hidden*;

}

*#prevBtn* **a,** *#nextBtn* **a**{

/\*\*

display:block;

position:relative;

width:30px;

height:77px;

background:url(../images/btn\_prev.gif) no-repeat 0 0;

\*\*/

}

*#nextBtn* **a**{

/\*\*background:url(../images/btn\_next.gif) no-repeat 0 0; \*\*/

}

/\* // Easy Slider \*/

### Index.php

<?php

// Define path to application directory

defined('APPLICATION\_PATH')

|| define('APPLICATION\_PATH', realpath(dirname(\_\_FILE\_\_) . '/../application'));

//Check to see if the install.php file exists

**if** (!(file\_exists(APPLICATION\_PATH . '/configs/' . 'installed.php'))) {

header('Location: /install.php');

**exit**();

}

ini\_set('display\_errors', 1);

// Define application environment

defined('APPLICATION\_ENV')

|| define('APPLICATION\_ENV', (getenv('APPLICATION\_ENV') ? getenv('APPLICATION\_ENV') : 'development'));

// Ensure library/ is on include\_path

set\_include\_path(implode(PATH\_SEPARATOR, **array**(

realpath(APPLICATION\_PATH . '/../library'), get\_include\_path()

)));

/\*\* Zend\_Application \*/

**require\_once** 'Zend/Application.php';

// Create application, bootstrap, and run

$application = **new** Zend\_Application(

APPLICATION\_ENV,

APPLICATION\_PATH . '/configs/application.ini'

);

$application->bootstrap()

->run();

### Install\_2.php

<?php

/\*\*

\* Set Required Configurations

\* as app has yet to bootstrap

\*/

ini\_set('display\_errors', 0);

defined('APPLICATION\_PATH')

|| define('APPLICATION\_PATH', realpath(dirname(\_\_FILE\_\_) . '/../application'));

defined('APPLICATION\_ENV')

|| define('APPLICATION\_ENV', (getenv('APPLICATION\_ENV') ? getenv('APPLICATION\_ENV') : 'development'));

**require\_once** 'Zend/Config/Ini.php';

$config = **new** Zend\_Config\_Ini(APPLICATION\_PATH . "/configs/application.ini", APPLICATION\_ENV,**true**);

**if**($\_POST) {

/\*\*

\* Get Post Data

\*/

$schoolName = $\_POST['schoolname'];

$schoolAddress = $\_POST['schooladdress'];

$schoolPhone = $\_POST['schoolphone'];

$schoolEmail = $\_POST['schoolemail'];

/\*\*

\* Get Database values from Ini

\*/

$server = $config->doctrine->host;

$port = $config->doctrine->port;

$username = $config->doctrine->user;

$password = $config->doctrine->password;

$databaseName = $config->doctrine->db;

/\*\*

\* Create Database Connection

\*/

$dbConnection = mysql\_pconnect($server . ':' . $port, $username, $password);

/\*\*

\* Use Database

\*/

mysql\_query("USE " . $databaseName);

/\*\*

\* Create and Execute Query

\*/

$sql = "INSERT INTO `schoolinfo` (`school\_name`, `school\_address`, `school\_phone`, `school\_email`) VALUES\n"

. "('" . $schoolName . "', '" . $schoolAddress . "', '" . $schoolPhone . "', '" . $schoolEmail . "');";

mysql\_query($sql);

/\*\*

\* Create installed.php

\*/

$ttFile = fopen(APPLICATION\_PATH . '/configs/' . 'installed.php', 'w');

fwrite($ttFile, 'MSM successfully installed');

fclose($ttFile);

/\*\*

\* Navigate to home page

\*/

header('Location: /Login');

}

?>

<head>

<script type=*"text/javascript"*>

**function** schoolsetup(form){

valid = **true**;

**if** ( document.schoolsetup\_form.schoolname.value == "" ){

document.getElementById('schoolname\_help').innerHTML = "Please enter a Name.";

valid = **false**;

} **else**{

document.getElementById('schoolname\_help').innerHTML = "";

}

**if** ( document.schoolsetup\_form.schooladdress.value == "" ){

document.getElementById('schooladdress\_help').innerHTML = "Please enter an address.";

valid = **false**;

} **else**{

document.getElementById('schooladdress\_help').innerHTML = "";

}

**if** ( document.schoolsetup\_form.schoolphone.value == "" ){

document.getElementById('schoolphone\_help').innerHTML = "Please enter a Phone number.";

valid = **false**;

} **else**{

document.getElementById('schoolphone\_help').innerHTML = "";

}

**if** ( document.schoolsetup\_form.schoolemail.value == "" ){

document.getElementById('schoolemail\_help').innerHTML = "Please enter an email.";

valid = **false**;

} **else**{

document.getElementById('schoolemail\_help').innerHTML = "";

}

**if** (valid == **true**){

form.submit();

}

}

</script>

<style>

<!--

**body** {

font-family: *sans-serif*;

margin: *20px*;

}

**table** {

border: *thin dotted #7e7e7e*;

padding: *10px*;

}

**th** {

text-align: *right*;

vertical-align: *top*;

padding-right: *10px*;

padding-top: *2px*;

}

**td** {

vertical-align: *top*;

padding-bottom: *15px*;

}

-->

</style>

<title>Music School Manager Install</title>

</head>

<body>

<h1>School Setup</h1>

<form name=*"schoolsetup\_form"* method=*"post"*>

<table>

<tr>

<th>

<label>Name:</label>

</th>

<td>

<input type=*"text"* name=*"schoolname"* />

<span id=*"schoolname\_help"*></span>

</td>

</tr>

<tr>

<th>

<label>Address:</label>

</th>

<td>

<input type=*"text"* name=*"schooladdress"* />

<span id=*"schooladdress\_help"*></span>

</td>

</tr>

<tr>

<th>

<label>Phone:</label>

</th>

<td>

<input type=*"text"* name=*"schoolphone"* />

<span id=*"schoolphone\_help"*></span>

</td>

</tr>

<tr>

<th>

<label>Email:</label>

</th>

<td>

<input type=*"text"* name=*"schoolemail"* />

<span id=*"schoolemail\_help"*></span>

</td>

</tr>

<tr>

<td>

<input type=*"button"* value=*"Finish"* onclick="schoolsetup(this.form);" />

</td>

</tr>

</table>

</form>

</body>

### Install.php

<?php

/\*\*

\* Set Required Configurations

\* as app has yet to bootstrap

\*/

ini\_set('display\_errors', 0);

defined('APPLICATION\_PATH')

|| define('APPLICATION\_PATH', realpath(dirname(\_\_FILE\_\_) . '/../application'));

defined('APPLICATION\_ENV')

|| define('APPLICATION\_ENV', (getenv('APPLICATION\_ENV') ? getenv('APPLICATION\_ENV') : 'development'));

**require\_once** 'Zend/Config/Ini.php';

**require\_once** 'Zend/Config/Writer/Ini.php';

$config = **new** Zend\_Config\_Ini(APPLICATION\_PATH . "/configs/application.ini", APPLICATION\_ENV, **true**);

$configWriter = **new** Zend\_Config\_Writer\_Ini();

**if**($\_POST) {

/\*\*

\* Get Post Data

\*/

$server = $\_POST['host'];

$port = $\_POST['port'];

$username = $\_POST['username'];

$password = $\_POST['password'];

$databaseName = $\_POST['dbname'];

/\*\*

\* Create Database Connection

\*/

$dbConnection = mysql\_pconnect($server . ':' . $port, $username, $password);

/\*\*

\* Path to SQ dump file

\*/

$dbFile = APPLICATION\_PATH . '/configs/db.sql';

/\*\*

\* Providing we can connect to MySQL

\*/

**if**($dbConnection) {

/\*\*

\* Create Database

\*/

mysql\_query("CREATE DATABASE " . $databaseName, $dbConnection);

**if** (mysql\_query("USE " . $databaseName)){

/\*\*

\* Create Database tables from dump file

\*/

$line ="";

$fp = fopen($dbFile, 'r');

**while** (! feof($fp)) {

$buffer = fgets($fp, 4096);

$line .= $buffer . PHP\_EOL;

**if** (strpos($buffer, ';')){

mysql\_query($line);

$line = "";

}

$buffer = "";

}

fclose($fp);

/\*\*

\* Set Ini values

\*/

$config->doctrine->user = $username;

$config->doctrine->password = $password;

$config->doctrine->port = $port;

$config->doctrine->host = $server;

$config->doctrine->db = $databaseName;

$configWriter->setConfig($config);

$configWriter->write(APPLICATION\_PATH . "/configs/application.ini");

/\*\*

\* Navigate to next setup page

\*/

header('Location: /install\_2.php');

} **else** {

**die**('The database ' . $databaseName . 'must be created manually before you can procede with this step. ' . mysql\_error());

}

mysql\_close($dbConnection);

} **else** {

**die**("Error connecting to database. " . mysql\_error());

}

}

?>

<head>

<script type=*"text/javascript"*>

**function** dbsetup(form){

valid = **true**;

**if** ( document.dbsetup\_form.username.value == "" ){

document.getElementById('username\_help').innerHTML = "Please enter a Username.";

valid = **false**;

} **else**{

document.getElementById('username\_help').innerHTML = "";

}

**if** ( document.dbsetup\_form.password.value == "" ){

document.getElementById('password\_help').innerHTML = "Please enter a Password.";

valid = **false**;

} **else**{

document.getElementById('password\_help').innerHTML = "";

}

**if** ( document.dbsetup\_form.port.value == "" ){

document.getElementById('port\_help').innerHTML = "Please enter a Port value (Default is 3306).";

valid = **false**;

} **else**{

document.getElementById('port\_help').innerHTML = "";

}

**if** ( document.dbsetup\_form.host.value == "" ){

document.getElementById('host\_help').innerHTML = "Please enter a Host.";

valid = **false**;

} **else**{

document.getElementById('host\_help').innerHTML = "";

}

**if** ( document.dbsetup\_form.dbname.value == "" ){

document.getElementById('dbname\_help').innerHTML = "Please enter a Database Name.";

valid = **false**;

} **else**{

document.getElementById('dbname\_help').innerHTML = "";

}

**if** (valid == **true**){

form.submit();

}

}

</script>

<style>

<!--

**body** {

font-family: *sans-serif*;

margin: *20px*;

}

**table** {

border: *thin dotted #7e7e7e*;

padding: *10px*;

}

**th** {

text-align: *right*;

vertical-align: *top*;

padding-right: *10px*;

padding-top: *2px*;

}

**td** {

vertical-align: *top*;

padding-bottom: *15px*;

}

-->

</style>

<title>Music School Manage Install</title>

</head>

<body>

<h1>Database Setup</h1>

<form name=*"dbsetup\_form"* method=*"post"*>

<table>

<tr>

<th>

<label>Username:</label>

</th>

<td>

<input type=*"text"* name=*"username"* />

<span id=*"username\_help"*></span>

</td>

</tr>

<tr>

<th>

<label>Password:</label>

</th>

<td>

<input type=*"text"* name=*"password"* />

<span id=*"password\_help"*></span>

</td>

</tr>

<tr>

<th>

<label>Port:</label>

</th>

<td>

<input type=*"text"* name=*"port"* value=*"3306"* />

<span id=*"port\_help"*></span>

</td>

</tr>

<tr>

<th>

<label>Host:</label>

</th>

<td>

<input type=*"text"* name=*"host"* />

<span id=*"host\_help"*></span>

</td>

</tr>

<tr>

<th>

<label>Database name:</label>

</th>

<td>

<input type=*"text"* name=*"dbname"* />

<span id=*"dbname\_help"*></span>

</td>

</tr>

<tr>

<td>

<input type=*"button"* value=*"Next"* onclick="dbsetup(this.form);" />

</td>

</tr>

</table>

</form>

</body>

# Technical Manual

## Software Requirements

* Windows Operating System
* MySQL 5.1
* Php 5.3

## Install Instructions

Unzip the archive.

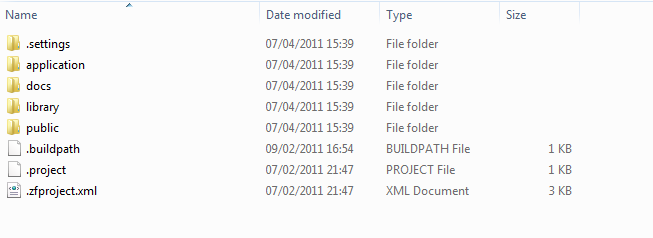


Figure : Archive Contents

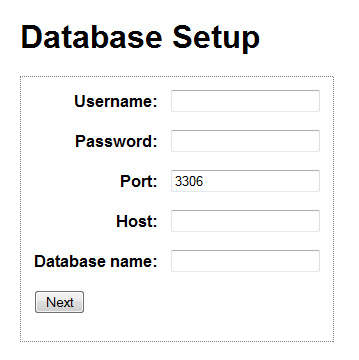


Figure :Install Step 1

Enter the correct configuration for your MySQL database, and click next. If the process cannot create the database for you, you will have to manually create the database, and supply its name in the Database name field.

Then you are ready to move onto step 2.

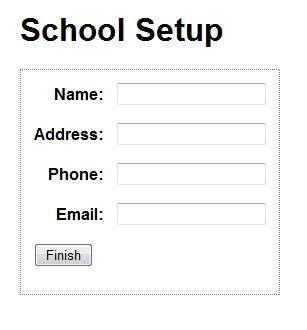


Figure : Install Ste

The next step is to enter some basic school information, once this data has been entered, click the Finish button to start your Music School Manager.

Default Login Details

Default admin Username: administrator

Default admin Password: administrator

**Important**: Please change your password ASAP.

If after this step, you are not greeted with the Login page, you should:

* Delete the file “../application/configs/installed.php”.
* Delete the MySQL database.
* Try again.

Alternatively:

1. Manually create a MySQL database.
2. Import the following dump file “../application/configs/db.sql” into the database created in step 1.
3. Execute the following SQL statement, replacing the values in red with the appropriate values:

$sql = "INSERT INTO `schoolinfo` (`school\_name`, `school\_address`, `school\_phone`, `school\_email`) VALUES ('schoolName', 'schoolAddress', 'schoolPhone', 'schoolEmail');";

1. Open applications/configs/application.ini and modify the following values in red:

* doctrine.user = "username" *MySQL Username*
* doctrine.password = "password" *MySQL password*
* doctrine.port = "3306" *3306 is the default value*
* doctrine.host = "server address” *127.0.0.1 is the default value*
* doctrine.db = "DatabaseName" *Your MySQL database name*