

3

Developing PHP Web Application

Chapter 3 - Developing PHP Web Application

With the DB Visual ARCHITECT (DB-VA) you can develop quality PHP Web Application much faster, better and cheaper. All DB-VA generated code, configuration file and persistent layer library are deployable to Apache Server with PHP. DB-VA generates all PHP code for accessing database. You do not need to write SQL to insert, query, update or delete the record. All code you need to program is plain PHP code (e.g. OrderDAO.save(myOrder);). In this chapter we will use a simple "School System" application to show you how to generate PHP code, configure your web application, creating, querying, updating and deleting objects. Again you do not need to write a single SQL statement for all the above operations.

The architecture of PHP Web Application with DB-VA Persistent Layer

In this chapter:

- Introduction
- Generate code in Apache Http Server
- Creating Object and Saving to Database
- Querying Object from Database
- Updating Object and Saving to Database
- Deleting Object in Database

Introduction

You will develop a School System.

The School System provides the following functions:

- Create course by teacher
- Enroll course for student
- Cancel course by teacher
- Register for user
- Modify the personal information
- View the Course information (number of students enrolled and teacher information of the course)

Required Software:

- DB Visual ARCHITECT 3.0 Java or Professional Edition (<http://www.visual-paradigm.com/download/>)
- PHP 4.4.2 or above (<http://www.php.net/>)
- Apache Http Server 2.20 or above (<http://httpd.apache.org/>)
- MySQL Server 4.1 or above (<http://dev.mysql.com/downloads/>)

Please open the SchoolSystem.vpp project file in the Chapter 3 School System.zip file. The project file contains the following diagrams. For the details about how to draw class diagram and entity relationship diagram, please see the Designer's Guide.

Class Diagram of School System:

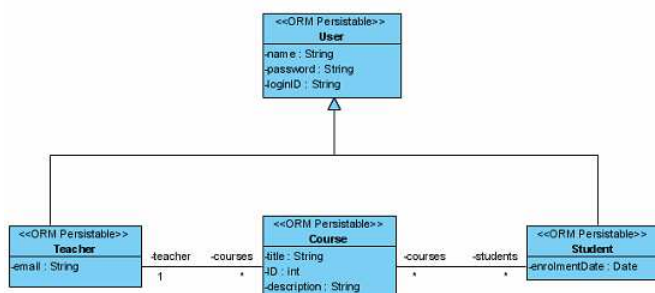


Figure 3.1 The class diagram

Entity Relationship Diagram of School System:

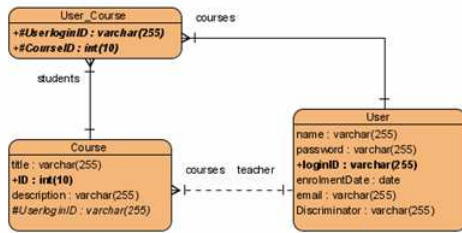


Figure 3.2 - The Entity Relationship Diagram (ERD)

Generate Code in Apache Http Server

You can generate the PHP code in Apache Http Server's htdocs folder or setup a virtual host to execution. The following step will teach you to directly generate PHP code to htdocs folder.

1. Create a new PHP Project in Eclipse. You must select the Directory in Installed Apache Http Server's htdocs.

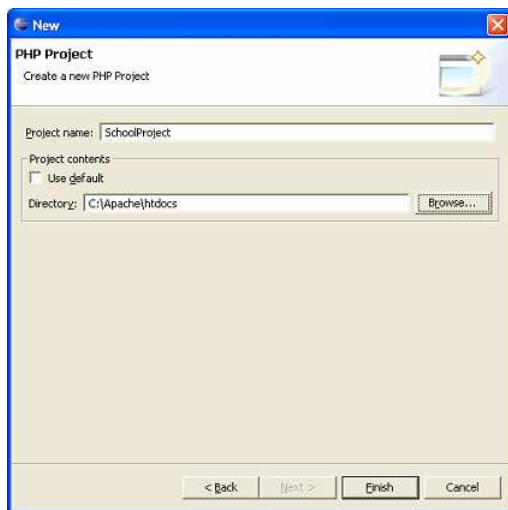


Figure 3.3 - Create a new project

2. Generate the PHP to the SchoolSystem Project in DB-VA. From menu bar, select **Tools > Object-Relational Mapping (ORM) > Generate Code...** to open the **Database Code Generation** dialog box.

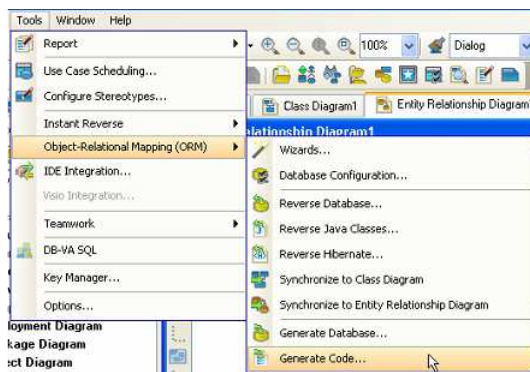


Figure 3.4 - Select generate code

- Fill in code generation information.

Code tab:

For **Generate**, select **Code and Database** to generate code and create database option.

For **Language**, select **PHP** language.

For **Output Path**, select the `C:\Apache\htdocs\SchoolSystem`. The project created in Eclipse.

For **Persistent API**, select **Factory Class**.

For **Sample**, check **Sample**.

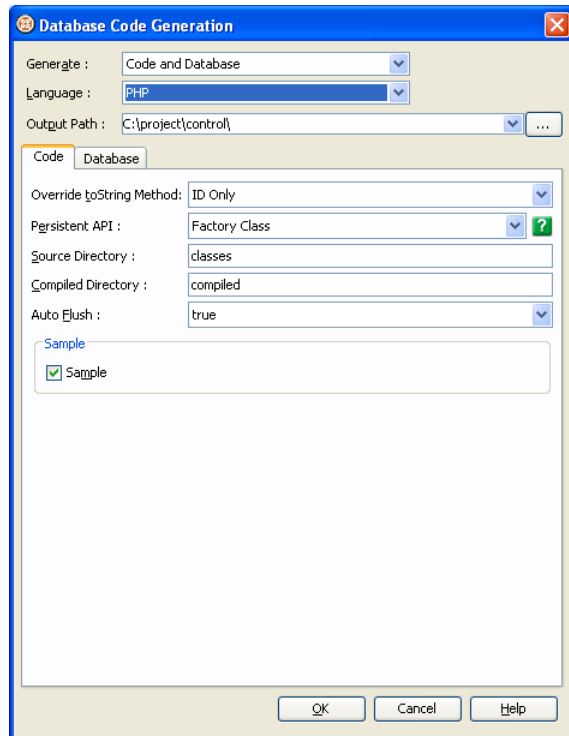


Figure 3.5 - Database Code Generation dialog

Database tab:

You can reference Chapter 1 to select export the database schema and configure the default database of JDBC connection.

- Click OK to the PHP code is generated to `C:\Apache\htdocs\SchoolSystem`.
- Refresh the SchoolSystem Project in Eclipse.

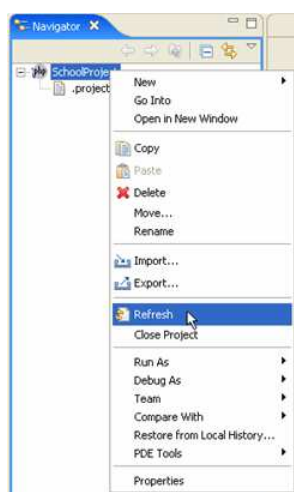


Figure 3.6 - To refresh the project

- The PHP Code is generated. You can start Apache Http Server and try to access the SchoolSystem project.

For example:

<http://localhost/SchoolSystem>

You can develop your PHP Web Application in Eclipse now.

Creating Object and Saving to Database

The school system provides a separate register page for teacher and student to enter their information. The register method of teacher and student method is the same, so we will demonstrate how to create the teacher and save to database.

- Create "teacherreg.html" for teacher to input their personal information and register to the system. The submit information will be processed by userlogin.php to add new user.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
  <title>Teacher Register</title>
</head>
<body>
  <h2>Teacher Register</h2>
  <hr>
  <form action="createUser.php" method="post">
    <table border="0" width="50%">
      <tr><td><strong>User ID : </strong></td><td><input name="loginID"
        type="text" id="loginID"/></td></tr>
      <tr><td><strong>Teacher Full Name : </strong></td><td><input
        name="name" type="text" id="name"/></td></tr>
      <tr><td><strong>Password : </strong></td><td><input name="password"
        type="text" id="password"/></td></tr>
      <tr><td><strong>Email : </strong></td><td><input name="email"
        type="text" id="email"/></td></tr>
      <tr><td></td><td><input type="submit" name="Submit"
        value="Submit"></td></tr>
    </table>
    <input name="userType" type="hidden" value="teacher"/>
  </form>
  <a href="index.html">Index page</a>
</body>
</html>
```

Source File : SchoolSystem\teacherreg.html

Teacher Register

User ID :	<input type="text" value="Tony"/>
Teacher Full Name :	<input type="text" value="Smith Tony"/>
Password :	<input type="text" value="1234"/>
Email :	<input type="text" value="simithony@abc.college"/>
	<input type="submit" value="Submit"/>

[Index page](#)

Figure 3.7 - Teacher Register page

2. Create the user to create the teacher record in system and provide the unique user ID.
 - The createUser.php can identify to create teacher or student by hidden field in "teacherreg.html"

```
<input name="userType" type="hidden" value="teacher"/>
```

- The createuser.php method to create teacher and student. It gets the information from the "teacherreg.html" by \$_POST.

```
$userType = $_POST["userType"];
$id = $_POST["loginID"];
$name = $_POST["name"];
$password = $_POST["password"];
```

Source File : SchoolSystem\createuser.php

- Create Teacher instance from TeacherFactory.

```
if ($userType == "teacher"){
    $user = TeacherFactory::createTeacher();
    $user->email = $_POST["email"];
} else {
    $curtime = time();
    $user = StudentFactory::createStudent();
    $user->enrolmentDate = date("Ymd");
}
```

Source File : SchoolSystem\createuser.php

- Call save() method of Teacher instance to create the record in database. Add Teacher id to session to represent the user has login to the system.

```
$user->save();
session_start();
session_register('userID');
$_SESSION['userID'] = $id;
```

Source File : SchoolSystem\createuser.php

3. When you submit the form the Teacher object will be added to the database. The new user id is added to the Session. If user attribute exists in session, the user is login to the system.

```
mysql> select * from user;
+-----+-----+-----+-----+-----+-----+
| name          | password | loginID | Discriminator | enrolmentDate | email          |
+-----+-----+-----+-----+-----+-----+
| Smith Tony   | 1234    | Tony   | Teacher      | NULL          | smithtony@abc.college |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

Figure 3.8 - The record is added

Querying Object from Database

You can retrieve the record in database as object. For example, you need to create the login function for the School System. You will require the user to input the User ID and password to login, the system retrieve the User object from the user id and compare the password to validate the user.

1. Create the Login page (login.php). It submits the form to the userlogin.php. If user has login, it will redirect the user to student page or teacher page depends on the user type.

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
  <title>Login Page</title>
</head>
<body>
  <?php
    require_once(
      realpath(dirname(__FILE__)).'/lib/phporm/ezpdo_runtime.php');
    session_start();
    $userID = $_SESSION['userID'];
    if (isset($userID))
      $user = UserFactory::loadUserByORMID($userID);
    if ($user == null){
  ?>
  <h2>Login</h2>
  <hr>
  <form action="userlogin.php" method="post">
    <table border="0" width="50%">
      <tr><td><strong>Login ID : </strong></td><td><input name="id"
        type="text" id="id"/></td></tr>
      <tr><td><strong>Password :</strong></td><td><input name="password"
        type="password" id="password"/></td></tr>
      <tr><td></td><td><input type="submit" name="Submit"
        value="Submit"></td></tr>
    </table>
  </form>
  <?php
    } elseif (is_a($user->getForeignObject(), 'Student')){
      header("Location: studentpage.php");
    } elseif (is_a($user->getForeignObject(), 'Teacher')){
      header("Location: teacherpage.php");
    }
  %>
  <a href="index.html">Index page</a>
</body>
</html>
```

Source File : SchoolSystem\login.php

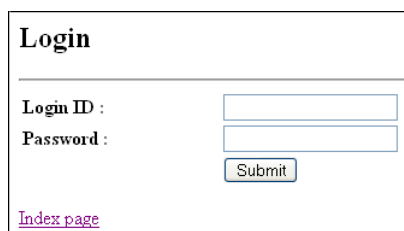


Figure 3.9 - Login Page

- Use User ID to call loadUserByORMID() method to retrieve the User object in userlogin.php and compare the password

```

$user = UserFactory::loadUserByORMID($id);
if (!is_null($user) && $user->password == $pwd){
    session_start();
    session_register('userID');
    $_SESSION['userID'] = $id;
if (is_a($user->getForeignObject(), 'Student')){
    header("Location: studentpage.php");
} elseif (is_a($user->getForeignObject(), 'Teacher')){
    header("Location: teacherpage.php");
}
}
}

```

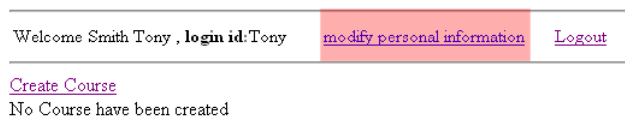
Source File : SchoolSystem\userlogin.php

Updating Object and Saving to Database

You can modify the teacher information and update the record in database. You get the User id from the session and get the user instance from database to set the new values for the User object, finally call save() method to update the record in database.

- Click modify personal information link on the Teacher Page

Teacher Page



```

Welcome Smith Tony , login id: Tony  modify personal information  Logout


---


Create Course
No Course have been created

```

Figure 3.10 - Teacher Page

- Modify the information in modifyUser.php. Get the User id from session and get the user instance from database to set the information to field and allow the user to modify the information, finally submit the form to the processModifyUser.php.

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
    <title>Modify Information</title>
</head>
<body>
    <h2>Modify User Information</h2>
    <hr>
    <?php
        require_once
            (realpath(dirname(__FILE__)).'/lib/phporm/ezpdo_runtime.php');
        session_start();
        $userID = $_SESSION['userID'];
        if (isset($userID))
            $user = UserFactory::loadUserByORMID($userID);
        if (is_null($user)) {
            header("Location: login.php");
        }
    ?>
    <form action="processModifyUser.php" method="post">
        <table border="0">
            <tr><td>Name : </td><td><input name="name" type="text"
                id="studentName" value="<?php echo $user->name; ?>"></td></tr>
            <tr><td>Password : </td><td><input name="password" type="text"
                id="password" value="<?php echo $user->password; ?>"></td></tr>
        <?php
            if (is_a($user->getForeignObject(), 'Teacher')){

```

```

?>
<tr><td>Email :</td><td><input name="email" type="text" id="email"
value="<?php echo $user->email; ?>"></td></tr>
<?php } ?>
<tr><td></td><td><input type="submit" name="Submit"
value="Submit"></td></tr>
</table>
</form>
</body>
</html>

```

Source File : SchoolSystem\modifyUser.php

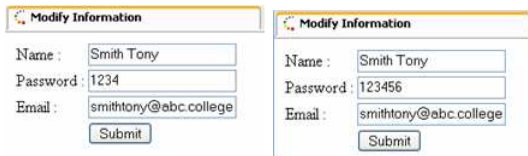


Figure 3.11 - Modify Information

3. The processModifyUser.php get the modified information from the submitted form and set to the User object and call save() method to update the record in database. It is similar to create user.

```

<?php
require_once (realpath(dirname(__FILE__)).'/lib/phporm/ezpdo_runtime.php');
session_start();
$userID = $_SESSION['userID'];
if (isset ($userID))
    $user = UserFactory :: loadUserByORMID($userID);
if (is_null($user)) {
    header("Location: login.php");
}

$name = $_POST['name'];
$password = $_POST['password'];
if (is_a($user->getForeignObject(), 'Teacher')){
    $email = $_POST['email'];
    $user->email = $email;
}
$user->name = $name;
$user->password = $password;
$user->save();
header("Location: login.php");
?>

```

Source File : SchoolSystem\processModifyUser.php

Deleting Object in Database

Teacher can create course for students for registration and they can cancel the course in the system. They only need to click cancel hyperlink of the course then the course information will be deleted in the database and all its relationship with registered students will be removed.

1. Teacher can create the course by selecting Create Course hyperlink in teacher page. Fill in Course name and Description to create Course.

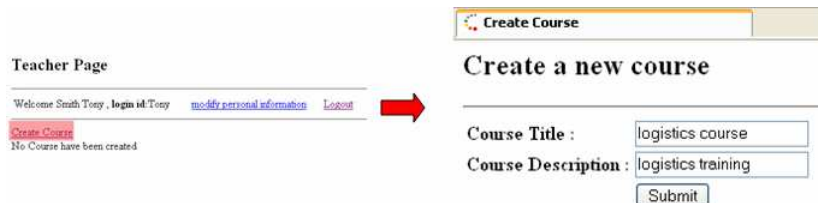


Figure 3.12 - Create Course

2. Student can register the course in the student page.



Figure 3.13 - Student Page

3. The teacher can view how many students have registered his course and he can cancel the course.

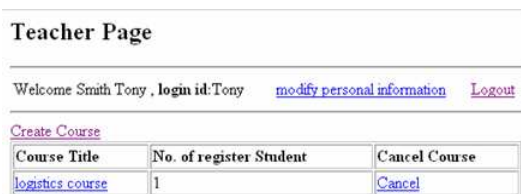


Figure 3.14 - Teacher Page

4. Click Cancel of a Course then it will pass the course id to the processDeleteCourse.php. The processDeleteCourse.php contains which uses the Course ID to retrieve to Course Object.

```
$courseID = $_GET[ 'courseID' ];
$course = CourseFactory::loadCourseByORMID( $courseID );
...
```

Source File : SchoolSystem\processDeleteCourse.php

5. Call deleteAndDissociate() method to delete the course object and remove the relationships of student and teacher with the course.

```
$course->deleteAndDissociate();
```

Source File : SchoolSystem\processDeleteCourse.php