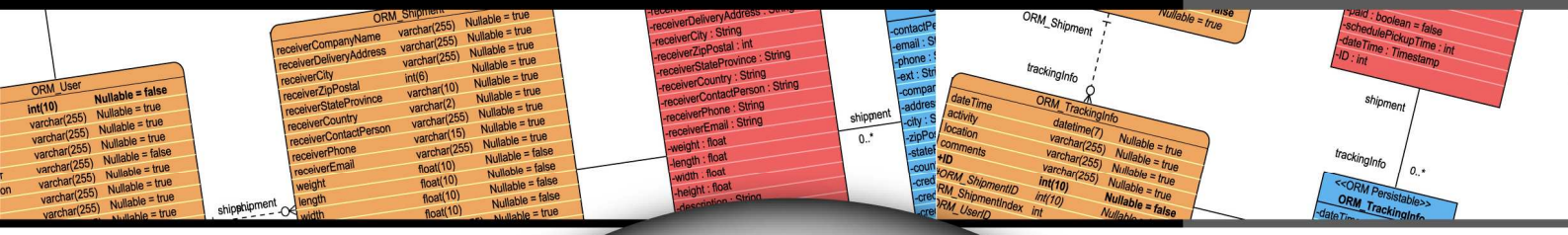
Visual Paradigm

Build Quality Applications Faster, Better and Cheaper

Database Visual ARCHITECT Programmer's Guide for PHP

Access database with Object-Oriented technology



DB Visual ARCHITECT 4.0 Programmer's Guide for PHP

The software and documentation are furnished under the DB Visual ARCHITECT license agreement and may be used only in accordance with the terms of the agreement.

DB Visual ARCHITECT License Agreement

THE USE OF THE SOFTWARE LICENSED TO YOU IS SUBJECT TO THE TERMS AND CONDITIONS OF THIS SOFTWARE LICENSE AGREEMENT. BY INSTALLING, COPYING, OR OTHERWISE USING THE SOFTWARE, YOU ACKNOWLEDGE THAT YOU HAVE READ THIS AGREEMENT, UNDERSTAND IT, AND AGREE TO BE BOUNDED BY ALL OF THE TERMS AND CONDITIONS OF THIS SOFTWARE LICENSE AGREEMENT.

1. **Limited License Grant.** Visual Paradigm grants to you ("the Licensee") a personal, non-exclusive, non-transferable, limited, perpetual, revocable license to install and use Visual Paradigm Products ("the Software" or "the Product"). The Licensee must not re-distribute the Software in whole or in part, either separately or included with a product.
2. **Restrictions.** The Software is confidential copyrighted information of Visual Paradigm, and Visual Paradigm and/or its licensors retain title to all copies. The Licensee shall not modify, adapt, decompile, disassemble, decrypt, extract, or otherwise reverse engineer the Software. Software may not be leased, rented, transferred, distributed, assigned, or sublicensed, in whole or in part. The Software contains valuable trade secrets. The Licensee promises not to extract any information or concepts from it as part of an effort to compete with the licensor, nor to assist anyone else in such an effort. The Licensee agrees not to remove, modify, delete or destroy any proprietary right notices of Visual Paradigm and its licensors, including copyright notices, in the Software.
3. **Disclaimer of Warranty.** The software and documentation are provided "AS IS," WITH NO WARRANTIES WHATSOEVER. ALL EXPRESS OR IMPLIED REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE OR NON-INFRINGEMENT, ARE HEREBY EXCLUDED. THE ENTIRE RISK AS TO SATISFACTORY QUALITY, PERFORMANCE, ACCURACY AND EFFORT IS WITH THE LICENSEE. THERE IS NO WARRANTY THE DOCUMENTATION, Visual Paradigm's EFFORTS OR THE LICENSED SOFTWARE WILL FULFILL ANY OF LICENSEE'S PARTICULAR PURPOSES OR NEEDS. IF THESE WARRANTIES ARE UNENFORCEABLE UNDER APPLICABLE LAW, THEN Visual Paradigm DISCLAIMS SUCH WARRANTIES TO THE MAXIMUM EXTENT PERMITTED BY SUCH APPLICABLE LAW.
4. **Limitation of Liability.** Visual Paradigm AND ITS LICENSORS SHALL NOT BE LIABLE FOR ANY DAMAGES SUFFERED BY THE LICENSEE OR ANY THIRD PARTY AS A RESULT OF USING OR DISTRIBUTING SOFTWARE. IN NO EVENT WILL Visual Paradigm OR ITS LICENSORS BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR DIRECT, INDIRECT, EXEMPLARY, SPECIAL, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF THE USE OF OR INABILITY TO USE THE SOFTWARE, EVEN IF Visual Paradigm HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

5. **Termination.** The Licensee may terminate this License at any time by destroying all copies of Software. Visual Paradigm will not be obligated to refund any License Fees, if any, paid by the Licensee for such termination. This License will terminate immediately without notice from Visual Paradigm if the Licensee fails to comply with any provision of this License. Upon such termination, the Licensee must destroy all copies of the Software. Visual Paradigm reserves all rights to terminate this License.

SPECIFIC DISCLAIMER FOR HIGH-RISK ACTIVITIES. The SOFTWARE is not designed or intended for use in high-risk activities including, without restricting the generality of the foregoing, on-line control of aircraft, air traffic, aircraft navigation or aircraft communications; or in the design, construction, operation or maintenance of any nuclear facility. Visual Paradigm disclaims any express or implied warranty of fitness for such purposes or any other purposes.

NOTICE. The Product is not intended for personal, family or household use; rather, it is intended exclusively for professional use. Its utilization requires skills that differ from those needed to use consumer software products such as word processing or spreadsheet software.

GOVERNMENT RIGHTS. If the Software is licensed by or on behalf of a unit or agency of any government, the Licensee agrees that the Software is "commercial computer software", "commercial computer software documentation" or similar terms and that, in the absence of a written agreement to the contrary, the Licensee's rights with respect to the Software are limited by the terms of this Agreement.

Acknowledgements

Table of Contents

1

Generating PHP, Database and Persistent Library

Chapter 1 - Generating PHP, Database and Persistent Library

DB Visual ARCHITECT (DB-VA) can generate PHP code, export database schema (DDL) to database and create the persistent library based on what your have designed in the class diagram and entity relationship diagram. DB-VA will generate a high performance O/R Mapping (ORM) layer library ready for you to code and build. The ORM library basically intends to take most of the relational to object-oriented mapping burden off your shoulder. With the generated ORM code and library, you can use the plain PHP objects in the application and tell the ORM layer to persist the object for you (e.g. ObjectDAO.save(myObject);) This chapter gives you an introduction to DB-VA, describes how to configure database, generate database and PHP code step by step.

In this chapter:

Introduction

DB Visual ARCHITECT (DB-VA) provides an easy-to-use environment bridging between object model, data model and relational database. You can visually model the system in both logical data design and physical data design, and DB-VA helps automate the mapping between object model and data model.

In this chapter, we assume that you know how to model your system with the class diagram and entity relationship diagram (please refer to the Designer's Guide for more details on modeling with class diagram and entity relationship diagram). Class diagram and entity relationship diagram will be used in this chapter to demonstrate how to make use of the DB-VA to export database schema (DDL) to database and generate PHP persistent code.

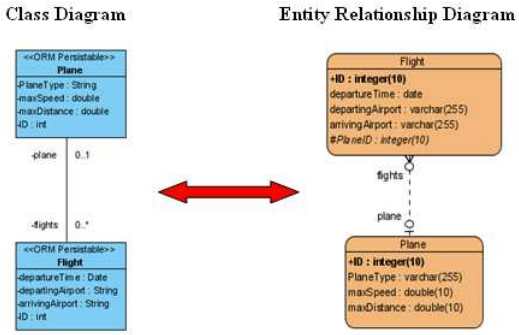


Figure 1.1 - Mapping between Classes and Entities

Configuring Database

DB-VA covers a wide range of databases in the market. You can check the latest supported databases version from http://www.visual-paradigm.com/product/dbva/

1. Please draw the above class diagram and synchronize it to ERD. (Alternatively, you can simply open the sample project, **flight.vpp**)
2. From the menu, select **Tools > Object-Relational Mapping (ORM) > Database Configuration...** to open the **Database Configuration** dialog box.

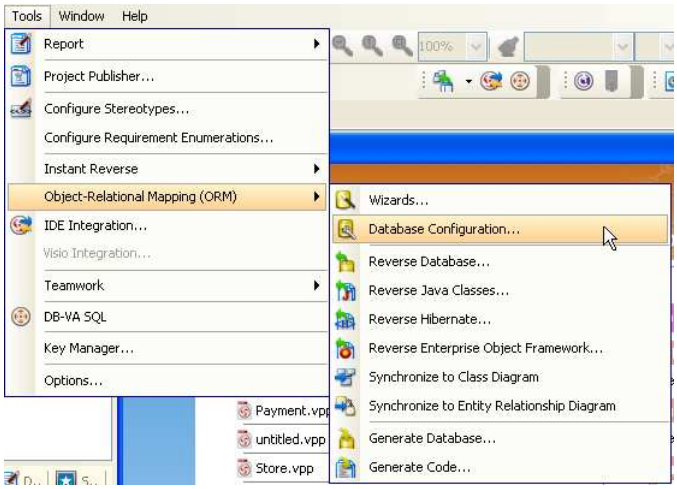


Figure 1.2 - To open Database Configuration

3. Select **PHP** from the drop-down menu of **Language** and select the desired database. MySQL database is used to demonstrate the database configuration.

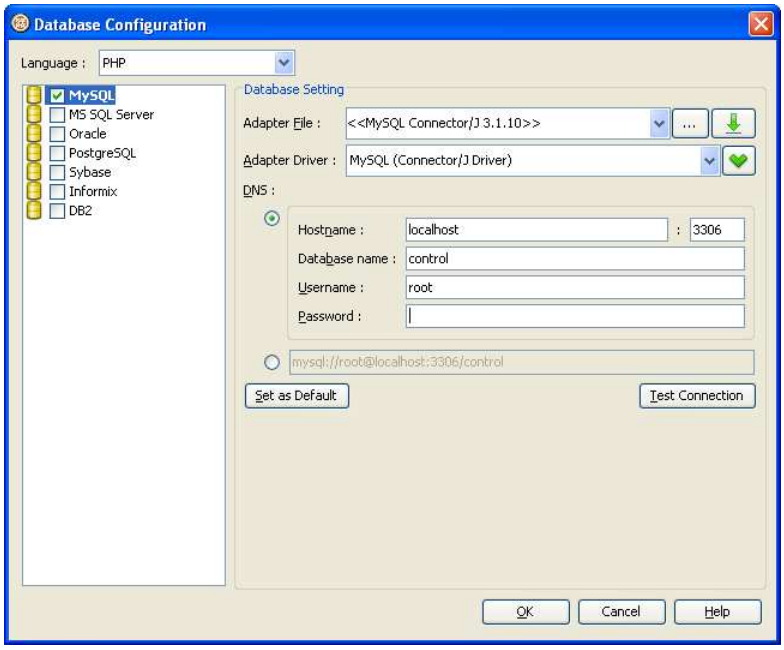


Figure 1.3 - Database Configuration dialog

4. Enter database setting for the selected database.

   For **Adapter File**, you can press the down arrow button to select **Download**, **Update** or **Default Driver**; DB-VA helps you to download the most up-to-date driver according to the **Driver** field information. You can also select **Browse...** to specify the driver file in your computer.

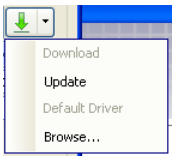


Figure 1.4 - Download button

   After downloaded the driver file, **<<MySQL Connector/J 3.1.10>>** shown on the **Driver file** indicates that the JDBC driver file is downloaded with the specified version number by DB-VA.

   For **Driver**, select the JDBC Driver from the drop-down menu. The driver's description will be shown in the **Database Driver Description** pane.

   For **DNS**, the DNS template for different database is shown. Enter the information for connecting the database.

   The default DNS template for MySQL is:

```
mysql://<username>:<password>@<host_name>:<port_number>/<database_name>
```

   The desired DNS MySQL is:

```
mysql://root@localhost/control
```

   For **User**, enter the valid username who has the access right to connect database

   For **Password**, enter the password for the this user.

5. Press **Test Connection** button after filling in the database information to test whether the database can be connected.

Figure 1.5 - Test Connection button

   If the database can be connected, the **Connection Successful** dialog box will show; otherwise the **Connection Exception** dialog box will be prompted.

Figure 1.6 - Connect successful/failure message

6. Select one database to be the default database connection for generating code and database. To set the default database connection, right click on database and select **Set as default.**

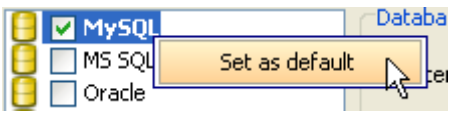


Figure 1.7 - Set the database as default

Generating Database

Now you can export the database schema from the Entity Relationship Diagram to the default database.

1. From the menu, select **Tools > Object-Relational Mapping (ORM) > Generate Database...** to open the **Database Code Generation** dialog box.

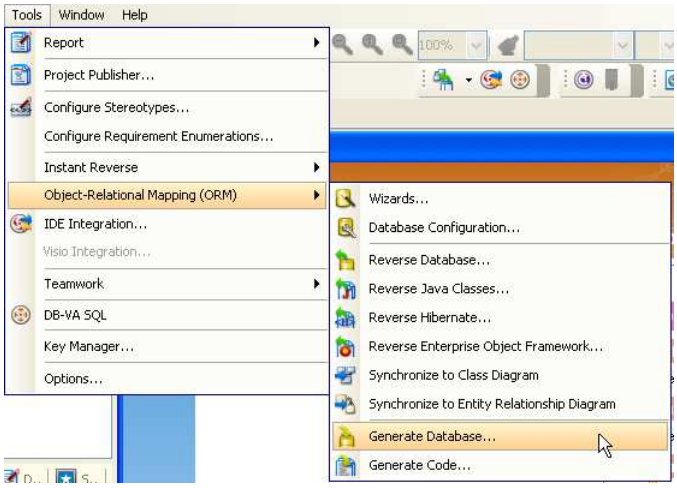


Figure 1.8 - Select generate database

The dialog box shows the previous default database setting.

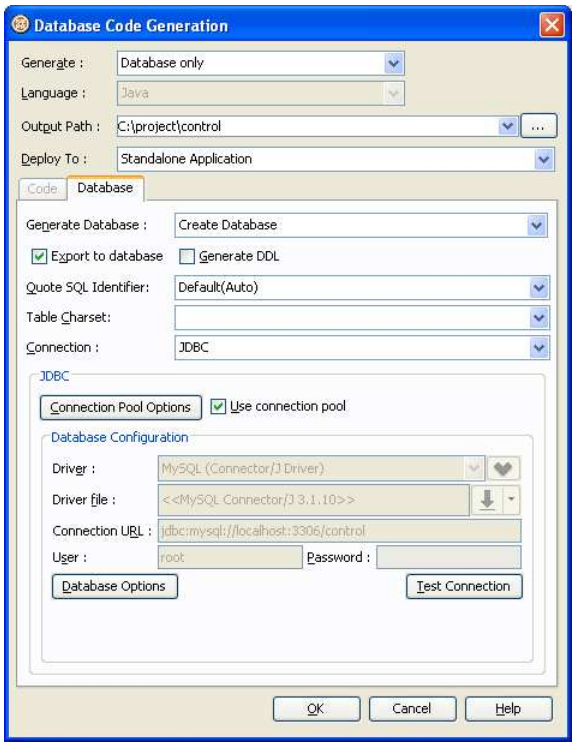


Figure 1.9 - Database Code Generation dialog

2. Select **Generate Database** option which specifies the action for the database. Since it is the first time to export database schema, you can select **Create Database** option. DB-VA allows you to select **Create Database**, **Update Database**, **Drop and Create Database** and **Drop Database**.

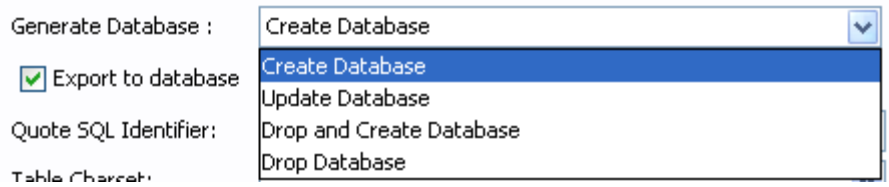


Figure 1.10 - Generate Database options

3. Select **Export to database** option to allow altering the database immediately after you click the **OK** button.

Figure 1.11 - Export to database option

4. Select **Generate DDL** option to allow the generation of DDL file.

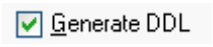


Figure 1.12 - Generate DDL option

5. You may use some reserved words (e.g. Order) in your database design, you can select the **Quote SQL Identifier** to avoid the naming problem in your design with the target database.
   - **Auto** - Only the detected reserved word will be quoted.
   - **Yes** - All table names and column names will be quoted.
   - **No** - No words will be quoted; i.e. reserved words cannot be used in generating the database.

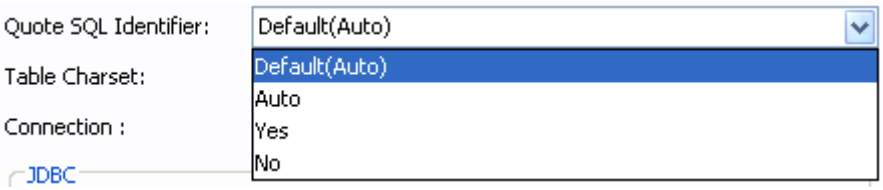


Figure 1.13 - Quote SQL Identifier options

6. Select **Table Charset** option to defined table character set standard.

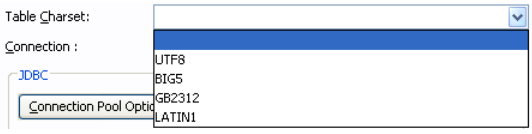


Figure 1.14 - Table Charset options

7. Click **OK** on the dialog box, DB-VA automatically exports the database schema to the default database and generate the DDL file to the specified output path.

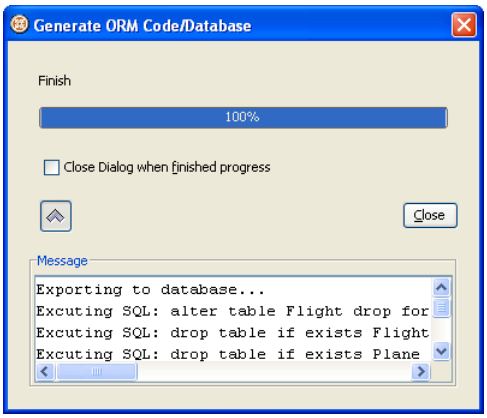


Figure 1.15 - Generate ORM Code/Database dialog

The generated DDL file is shown as below.

```
create table Flight (ID int not null auto_increment, departureTime date, departingAirport
varchar(255), arrivingAirport varchar(255), PlaneID int, primary key (ID)) type=InnoDB;

create table Plane (ID int not null auto_increment, PlaneType varchar(255), maxSpeed double
not null, maxDistance double not null, primary key (ID)) type=InnoDB;

alter table Flight add index FK_Flight_1115 (PlaneID), add constraint FK_Flight_1115 foreign
key (PlaneID) references Plane (ID);
```

If you choose to generate DDL only without exporting to database, you can later use the generated DDL to generate database manually.

```
C:\Documents and Settings\mingkit>mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2 to server version: 5.0.15

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> create database control;
Query OK, 1 row affected (0.03 sec)

mysql> mysql> Aborted

C:\Documents and Settings\mingkit>mysql -u root control
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3 to server version: 5.0.15

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> show tables;
+-------------------+
| Tables_in_control |
+-------------------+
| flight            |
| plane             |
+-------------------+
2 rows in set (0.02 sec)

mysql>
```

Figure 1.16 - The tables generated in database

Generating PHP Code

After generating and exporting the database, you can generate the persistent Java code from the class diagram for developing your application. The following are the steps to generate Java code:

1. From the menu, select **Tools > Object-Relational Mapping (ORM) > Generate Code...**to open **Database Code Generation** dialog box.

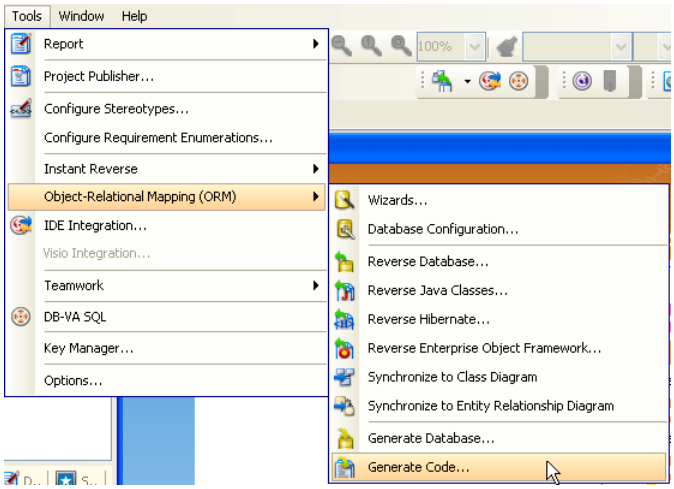


Figure 1.17 - Select generate code

2. Specify the setting for PHP code generation on the **Database Code Generation** dialog box.

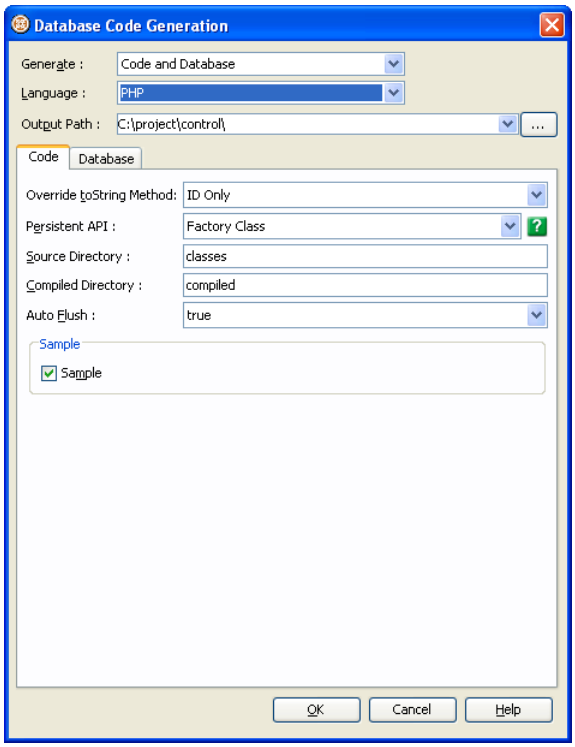


Figure 1.18 - Database Code Generation dialog

For **Output Path**, specify the location for storing the generated Java persistent code.

For **Persistent API**, select the type of persistent code to be generated, either Static Methods, Factory Class, DAO or POJO.

- **Static Method** -Client can create, retrieve and persist with the PersistentObject directly.
- **Factory Class** -FactoryObject class will be generated for client to create and retrieve the PersistentObject. Client can directly persist with the PersistentObject.
- **DAO** -The PersistentObjectDAO class helps client to create, retrieve and persists to PersistentObject.
- **POJO** -The PersistentManager helps client to retrieve and persist with PersistentObject.

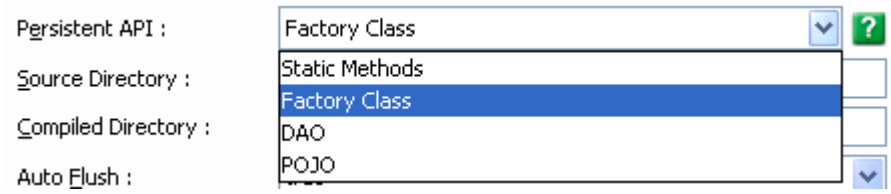


Figure 1.19 - Persistent API options

For **Source Directory,** the directory to store the generate classes from DB-VA.

For **Compile Directory,** the directory to store the compiled information.

For **Auto Flush**, saves all objects it creates before quitting the script.

For **Samples**, sample files, including PHP application sample is available for generation. The generated sample files guide you through the usage of the PHP persistence class. You can check the options to generate the sample files for reference.

You have to select samples for generation in this example so that you can modify the sample file to test and execute the generated PHP code.

3. The generated **Source Directory** folder contains the PHP persistent source code and the **lib** folder contains the persistent library.

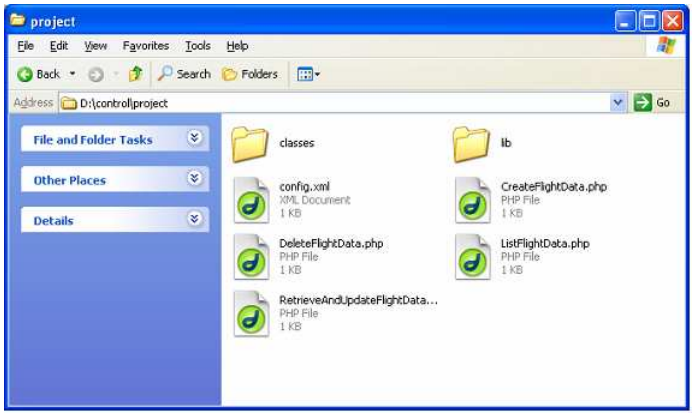


Figure 1.20 - Teh generated PHP source code

2 Configuring Source and Library in Eclipse

Chapter 2 - Configuring Source and Library in Eclipse

In Chapter 1, you have generated PHP code, exported database schema (DDL) and persistent library. In this chapter, we will show you how to import the generated code from Chapter 1 to Eclipse. You will create a new project and copy the generated persistent code and library to Eclipse, and you can further your development with the generate model efficiently in Eclipse.

In this chapter:

- Install PHP plugin for Eclipse
- Copying Generated Source and Library to Eclipse Project
- Modifying the Sample Program to Test the Generated PHP Model

Installing PHP plugin for Eclipse

The PHPEclipse is a PHP plugin can support for Eclipse IDE Framework. It provides a flexible environment to develop PHP project just like develop Java project in Eclipse. Some features are PHP parser, debugger, code formatter, outline view, templates...etc.

1. Download PHPEclipse from http://sourceforge.net/project/showfiles.php?group_id=57621
2. Unzip the download PHPEclipse and copy to the installed Eclipse folder
3. Start Eclipse and open the perspective of PHP.

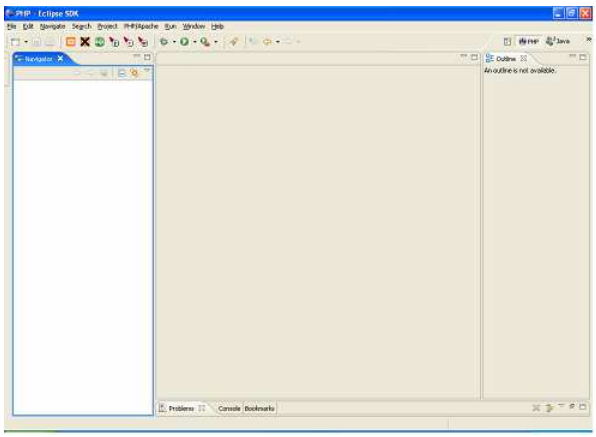


Figure 2.1 - Eclipse with PHPEclipse

Copying Generated Source and Library to Eclipse Project

You can copy the generated persistent code to Eclipse project and develop an application.

1. Open Eclipse and select **File** > **New** > **Project...** from menu

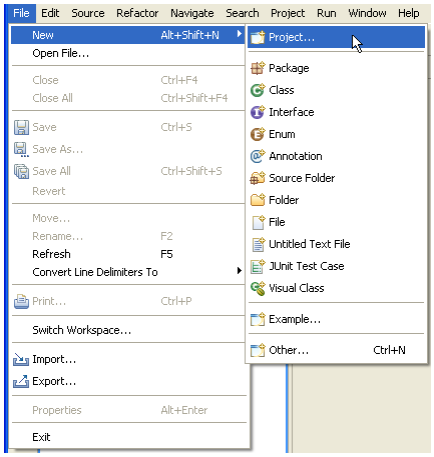


Figure 2.2 - Create a new project

2. Select **Java Project**, click **Next >.**

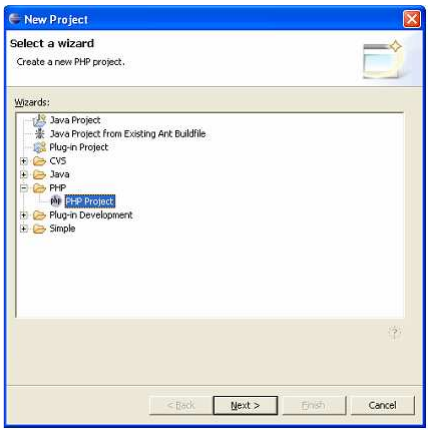


Figure 2.3 - Select PHP project

3. Enter "**AirportProject**" for the project name.

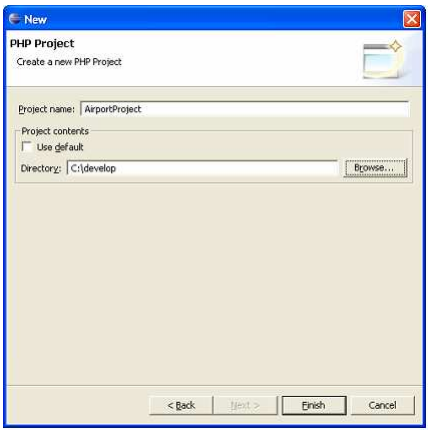


Figure 2.4 - Enter the Project name

4. The AirportProject is created and shown on the **Navigator**.

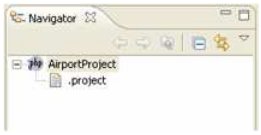


Figure 2.5 - The PHP project is created

5. Copy the PHP persistent code from the specified output folder.

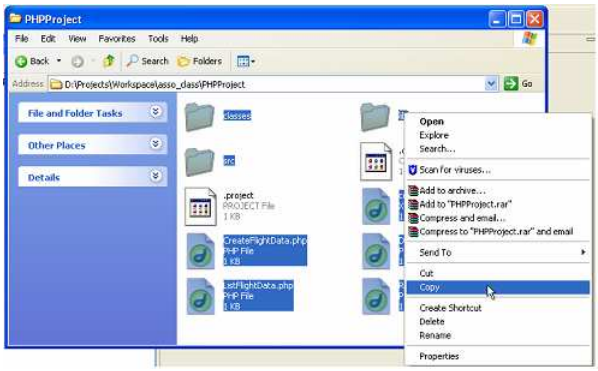


Figure 2.6 - Copy the generated code

6. Paste the Java persistent code on the AirportProject in **Package Explorer**.

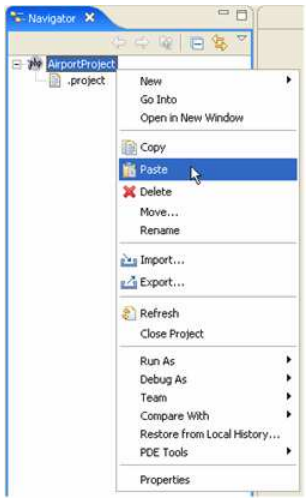


Figure 2.7 - Paste in Eclipse

7. All the generated PHP persistent codes are copied to the AirportProject.

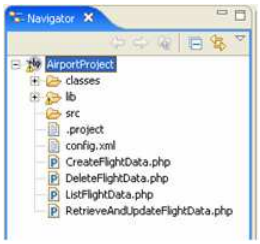


Figure 2.8 - The generated PHP files will copy to the Eclipse project directory

Modifying the Sample Program to Test the Generated PHP Model

Now the AirportProject is able to compile, you can modify the code in generated sample to test the persistent class interact with the created database.

1. Open the **src/ormsamples/CreateAirportData.php** file. The following is the original code

```
<?php
require_once( realpath(dirname(__FILE__)).'/lib/phporm/ezpdo_runtime.php');
/**
* "Visual Paradigm: DO NOT MODIFY THIS FILE!"
*
* This is an automatic generated file. It will be regenerated every time
* you generate persistence class.
*
* Modifying its content may cause the program not work, or your work may lost.
*/

/**
* Licensee: Demo
* License Type: Purchased
*/
class CreateFlightData {
	public function createTestData() {
		$flight = FlightFactory::createFlight();
		// Initialize the properties of the persistent object
		$flight->save();

		$plane = PlaneFactory::createPlane();
		// Initialize the properties of the persistent object
		$plane->save();
	}
}

$createFlightData = new CreateFlightData();
$createFlightData->createTestData();
?>
```

2. Modify to create Flight and Plane instance and create the relationship between them

```
public function createTestData() {
echo "Create Flight\n";
$flight = FlightFactory::createFlight();
$flight->departureTime = "20060203";
$flight->departingAirport = "Kansai International Airport";
$flight->arrivingAirport = "Hong Kong International Airport";

echo "Create Plane\n";
$plane = PlaneFactory::createPlane();
$plane->planeType = "747 plane";
$plane->maxSpeed = (float)967;
$plane->maxDistance = (float)8232;

echo "Create the relationship between Flight and Plane\n";
$plane->flights[] = $flight;

echo "Save the Plane and the Flight object\n";
$plane->save();
}
```

There is no record for the Flight and Plane tables before the execution of the sample application.

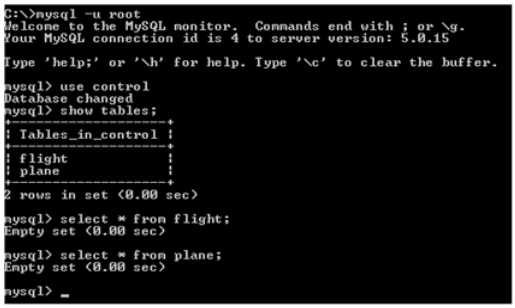


Figure 2.9 - There is no record in the tables

3. Open command prompt and go to the project folder. Type "php CreateFlightData" to execute the CreateFlightData code.

   The execution result:

Figure 2.10 - The execute result

4. After executing the CreateAirportData.java, new records are created in tables.

Figure 2.11 - The records are created on database

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:

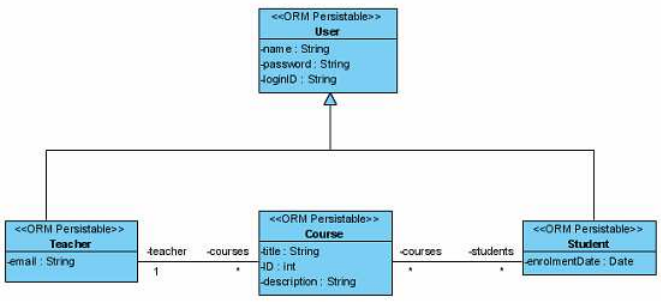


Figure 3.1 The class diagram

Entity Relationship Diagram of School System:

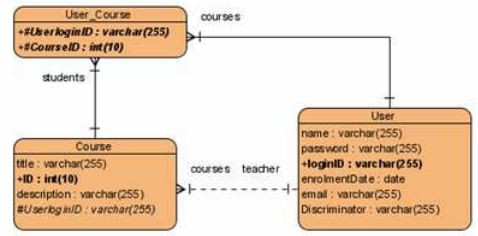


Figure 3.2 - The Entity Relationship Diagram (ERD)

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.

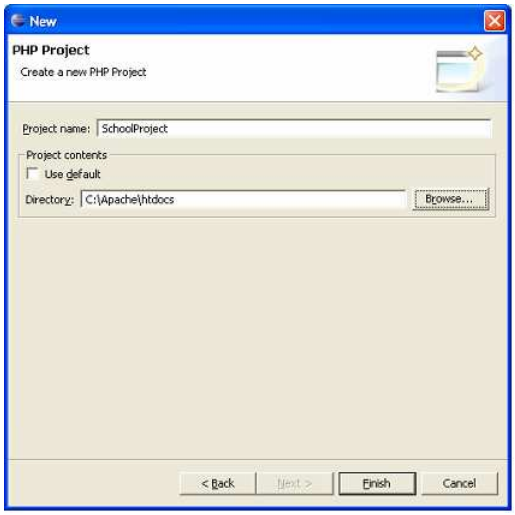


Figure 3.3 - Create a new project

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.

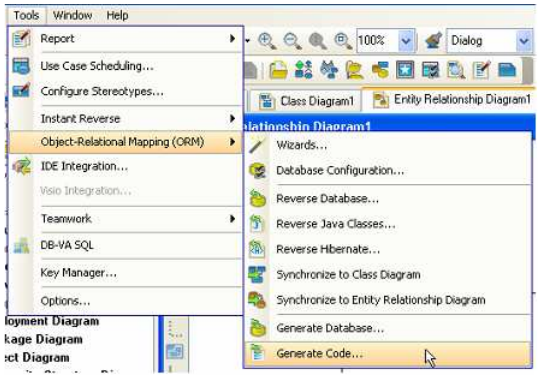


Figure 3.4 - Select generate code

3. 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**.

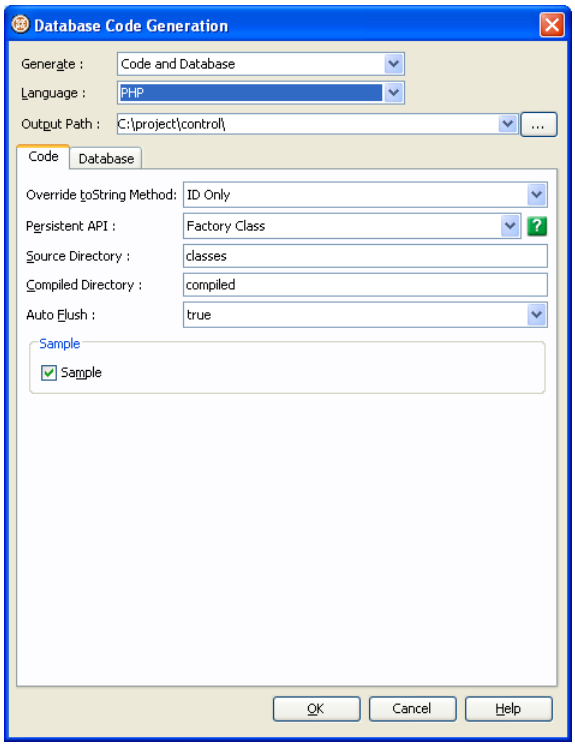


Figure 3.5 - Database Code Generation dialog

Database tab:

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

4. Click OK to the PHP code is generated to **C:\Apache\htdocs\SchoolSystem**.
5. Refresh the SchoolSystem Project in Eclipse.

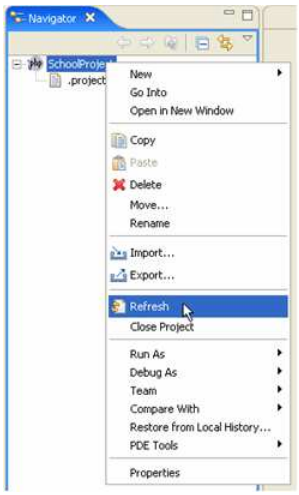


Figure 3.6 - To refresh the project

6. 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.

1. 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

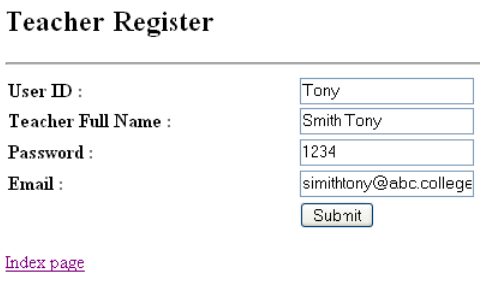


Figure 3.7 - Teacher Register page

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"];
     $pwd = $_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.

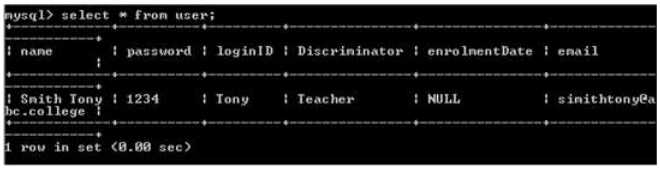


Figure 3.8 - The record is added

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

2. 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.

1. Click modify personal information link on the Teacher Page

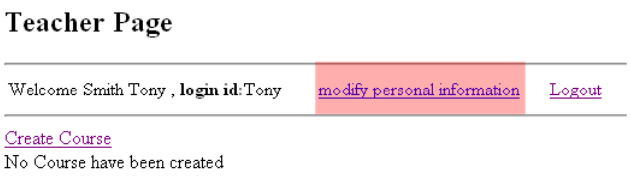


Figure 3.10 - Teacher Page

2. 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

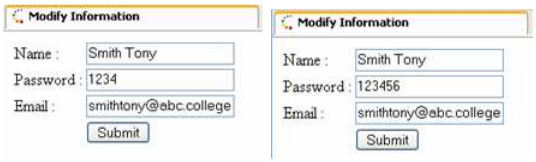


Figure 3.11 - Modify Information

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'];
        $pwd = $_POST['password'];
        if (is_a($user->getForeignObject(), 'Teacher')){
                $email = $_POST['email'];
                $user->email = $email;
        }
        $user->name = $name;
        $user->password = $pwd;
        $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.

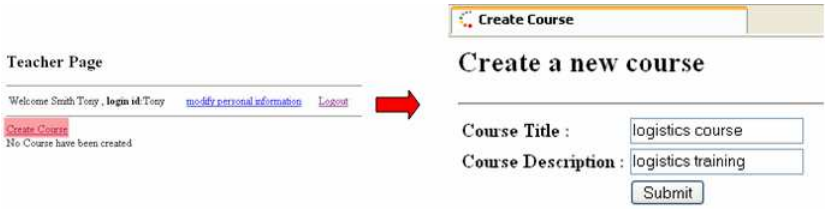


Figure 3.12 - Create Course

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

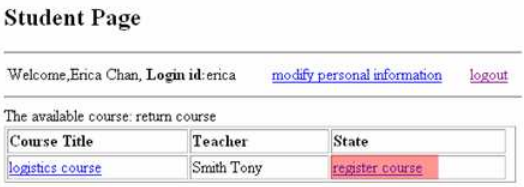


Figure 3.13 - Student Page

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

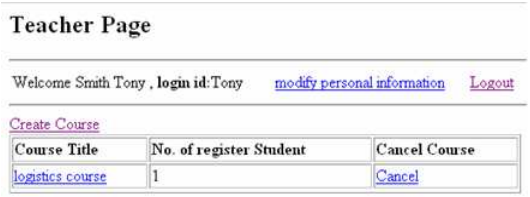


Figure 3.14 - Teacher Page

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