Web-based IDE to create Model and Controller Components for MVC-based Web Applications on CakePHP

A Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Computer Science

by Sugiharto Widjaja
May 2010

ABSTRACT

Web-based IDE to create Model and Controller Components for MVC-based Web Applications on CakePHP

by Sugiharto Widjaja

The purpose of this project is to develop a Web-based IDE that will allow users to easily create Model and Controller components of a Web application on CakePHP. By using this IDE, users do not need to have extensive knowledge of SQL and do not need to deal with complex SQL queries. Users will be able to design any number of database tables, and the IDE should create the database tables automatically. In addition to creating database tables, users will also be able to design complex queries to insert and retrieve data. The IDE will be able to automatically create PHP scripts for the Model and Controller components. It will populate the Model script with all the queries that are designed by the users. The IDE will also include defense mechanisms against popular attacks such as cross-site scripting attacks or SQL injection attacks.

Table of Contents

List of Figures

1. Introduction

The goal of my CS 298 project is to create a Web-based IDE that will allow users to easily create the Model and Controller components for MVC-based Web applications on CakePHP. Users can use this IDE to easily create database tables, insert data to them, and create complex queries to retrieve certain data from them. They will not need to know any SQL nor will they need to worry about the naming conventions, as this IDE will take care of those. The IDE will also create basic PHP scripts for the Controller component.

CakePHP is an open-source Web applications framework written in PHP, and it is modeled after Ruby on Rails. Like Ruby on Rails, CakePHP has three important components: Model, View, and Controller. The model component consists of the data that is used in the Web applications and the functions to retrieve them. The View component is basically the front-end of the Web application and is usually in the format of HTML. The Controller is the component that interacts with both the View (via Web form, mouse clicks, etc) and the Model (by calling certain functions in the Model to get some data).

The creation of the Model component will involve creating tables in database, inserting data to them, and creating functions to retrieve certain data from them. Users will have to go through all the above steps in order to create the Model component. They need to make sure that the naming conventions are correct, or the Web application will not work. They also have to work on the Controller component by writing PHP scripts. By using my IDE, users will have an easy Web-based tool to create model and controller components.

My IDE is comparable to applications such as phpMyAdmin (a Web-based application) or Microsoft Access (a desktop application). However, it will be a Web-based application instead of desktop application. My IDE will be simpler to use than phpMyAdmin since it will use the drag-and-drop approach. The IDE will also be more secure because it will include defense mechanisms against attacks such as cross-site scripting and SQL injection attacks.

In this report, I am going to discuss my deliverables that will prepare me for the implementation of the actual IDE in CS 298. In Deliverable 1, I created a simple Web application that utilized jQuery drag-and-drop feature. In Deliverable 2, I implemented a Web application that allowed users to design

database tables, link them, and import them to actual SQL statements. In Deliverable 3, I implemented a Web application that let users link two or more tables and use QBE (Query by Example) to construct SQL queries. In Deliverable 4, I combined the two Web applications in Deliverable 2 and 3 into one Web application.

2.1 Deliverable 1: Simple drag-and-drop Web application with jQuery

The purpose of this deliverable was to experiment with various features in jQuery and to develop a simple drag-and-drop Web application. I chose a drag-and-drop Web application because my actual IDE in CS 298 will require extensive amounts of drag-and-drop. Therefore, mastering the drag-and-drop feature of jQuery prepared me better for the implementation of the IDE in CS 298.

In this application, users can create new objects (represented by a colored rectangle) by clicking on "Create new object". Upon clicking that button, user will be presented with a simple form, where user can enter the color and the message contained in the object. If the user clicks on the "Create!" button, a small rectangle with user's specified color and message will appear. User will be able to drag the rectangle across the screen.

User will also be able to drop that rectangle to the "Trash Div" section. The "Trash Div" section will behave like the Recycle Bin in Windows O/S or Trash in Max OS. Once the rectangle is dropped on trash div, a confirmation dialog box will occur to confirm the deletion. If the user chooses to delete it, the rectangle will disappear in a slow animation style.

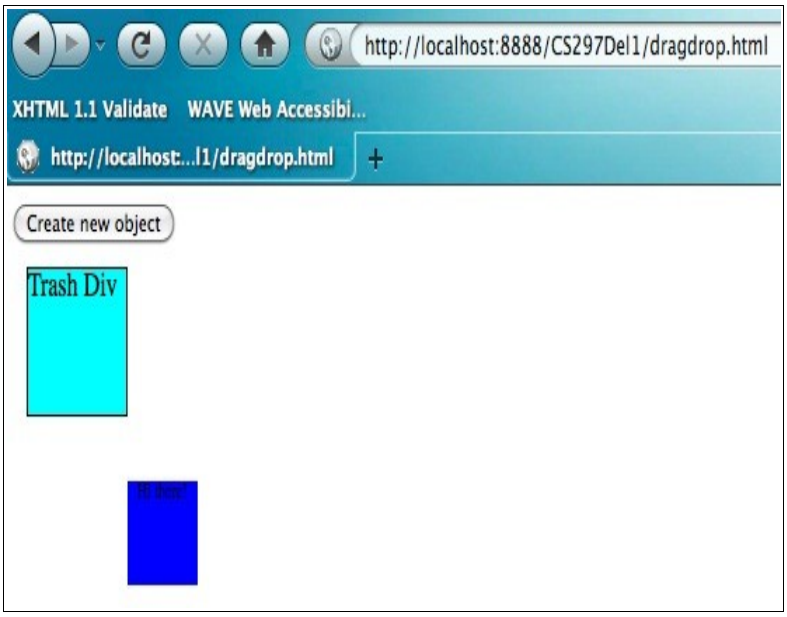

Figure 1: The drag-and-drop Web application

2.2 Deliverable 2: Creating Web application for creating database tables

The main goal of this deliverable was to create a simple Web application that will allow users to design database tables, link them, and import them to actual SQL statements. Users can use this simple application to design simple database tables without knowing any SQL. They can link the two tables without the knowledge of table joining concept in database. Finally, they can also import all the tables and links that they have created into the SQL statements. For this deliverable, I came up with data structures to store the database, tables, attributes, and tables relationships. These data structures will serve as the base model for my actual CS 298 project.

To start creating the tables, the user will click on the “Create new table” button. A simple form will appear, and the user can start adding the attributes to the tables. The user can also delete any attribute from the table. Clicking on the “create the table” button in the form will create the table. The table will be represented by a diagram on the right panel of the screen. Clicking on the [X] link on the diagram will allow user to delete this particular table, while clicking on the [E] link will allow user to edit this table. If the user edits and saves the table, the diagram will be automatically updated to reflect the changes. We can link (join) two or more tables together to retrieve the data from the tables in SQL. This application will allow users to link two or more tables together without having to worry about SQL statements. By dragging an attribute from the first table and drop it on an attribute from the second table, a link will be created between the two tables. In this deliverable, user has to drag a foreign key attribute from first table and drop it on the second table. The foreign key attribute is the attribute that connects the two tables together. This restriction was removed in Deliverable 4. In Deliverable 4, the foreign key was created automatically, when user linked two tables together.

Finally, this application will allow user to import all their tables into a SQL statements. User can run these SQL statements to create the required tables and relationships in the DBMS.

The challenge that I found in this deliverable is to come up with a way to draw the line that connect the two tables together and to update the line when the tables are moved. I decided to use the Walterzorn JavaScript Graphics Library to draw the line between two tables. This library basically creates many div elements to represent the line and stores them inside a div element (canvas).
<div style="position: absolute; left: {x px}; top: {y px}; width: 2px; height: 1px; clip: rect(0pt, 2px,

1px, 0pt); background-color: rgb(0, 0, 0);">

When one of the two tables is moved, the line has to be updated. I achieved this by clearing the current div elements inside the canvas and redraw the line.

I introduced four classes in this deliverable:

- DB : represents the current active database.
- Table: represents the database table.
- Attribute: represents the attributes of a database table.
- Relationship: represents existing relationship between two tables.

A DB object can have one or more tables and relationships (Note that a relationship requires two tables). A database table can have one or more attributes. These four classes were used in the rest of the deliverables.

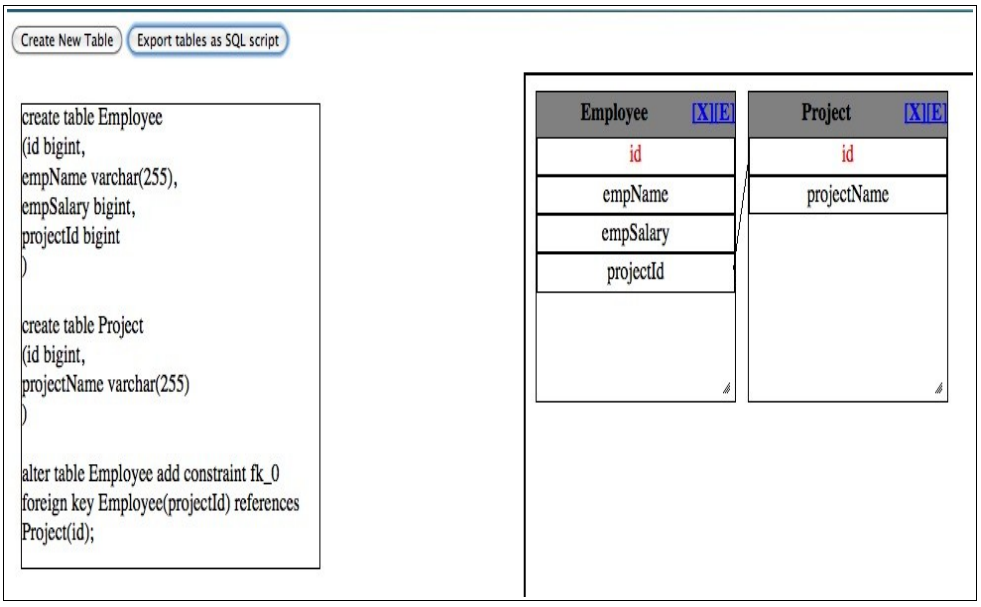

Figure 2: Importing Employee and Project table to SQL statements.

2.3 Deliverable 3: Using QBE to construct SQL queries

The goal of this deliverable was to create a simple Web application that allowed users to link two or more tables together and use QBE (Query by Example) to construct a complex SQL queries. QBE is a feature that is provided in various database applications (for example: Microsoft Access, phpMyAdmin). Without QBE, users have to write SQL queries (they can be really long and complex) to retrieve data from the database. QBE will provide users with a simple graphical user interface (GUI) to design simple or complex queries. The inclusion of a QBE interface will definitely be very helpful for users, and it will be included in the actual IDE in CS 298.

In this deliverable, I used three static tables:

- Employee(id, empName, empDeptId)
- Department(id, deptName, deptBuildingId)
- Building (id, buildingName)

The GUI has three panels. The topmost panel contains the diagrams of the three tables. The middle panel has a QBE interface that will allow users to construct their SQL queries. The bottom panel will contains important messages and the SQL queries that users have constructed.
If the user links table Employee and Department, the QBE interface will display the attributes of Employee and Department tables. User can select the attributes that they want to include in the SQL queries and set the aliases for them. For example: to include the attribute Employee.empName and to set alias 'ename' to it, the user will enter 'ename on the alias row under the empDeptId column and click on the checkbox in the Show row. The SORT option will give user the ability to sort the return data by the particular attribute. The Criteria option will allow user to specify certain conditions that has to be true for the particular data to be displayed.

Users can also construct more complex SQL queries. For example, if user wants to display all employees that works in department 'Accounting' and their names are not 'Joe Black' sorted by employee ID in ascending order, he will do the following:

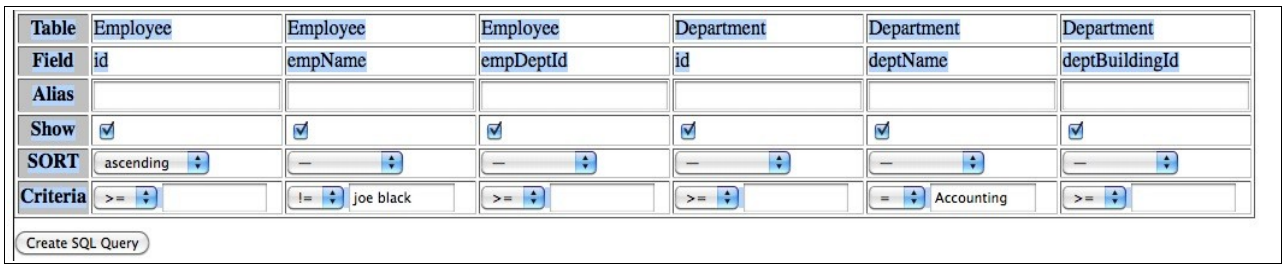

Figure 3: Using the QBE

This will result in SQL query:

select Employee.id, Employee.empName, Employee.empDeptId, Department.id, Department.deptName, Department.deptBuildingId from Employee join Department on Employee.empDeptId = Department.id where Employee.empName != 'joe black' and Department.deptName = 'Accounting' order by Employee.id ASC

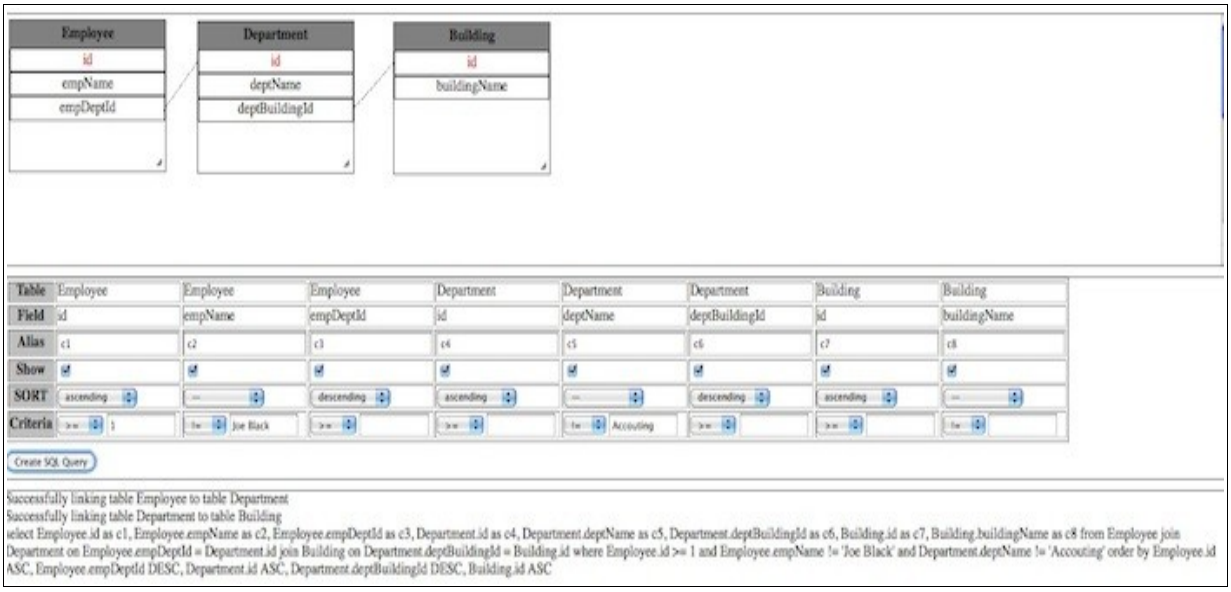

Figure 4: The Complete GUI of Deliverable 3

2.4 Deliverable 4: Combining table design and queries design (QBE) into one Web application

In this deliverable, I combined all the features that are found in Deliverable 2 and 3. In this application, users will be able to do the following:

- Create new tables
- Link two or more tables
- Use QBE to create complex SQL queries
- Import the SQL queries

The GUI that I developed in this deliverable will serve as the base model for the actual IDE in CS 298.

As I have mentioned in Deliverable 2, this deliverable included a new feature: auto creation of foreign key. In Deliverable 2, user needed to manually add the foreign key attribute to the first table before linking it to the second table. This can be inconvenient since users might not have knowledge of the foreign key concept. With this new feature, users could just link the two tables without having to worry about the foreign keys. Figure 5 below shows the automatic creation of foreign key Employee_Department_fk when we link tables Employee and Department, and foreign key Department_Building_fk when we link tables Department and Building.

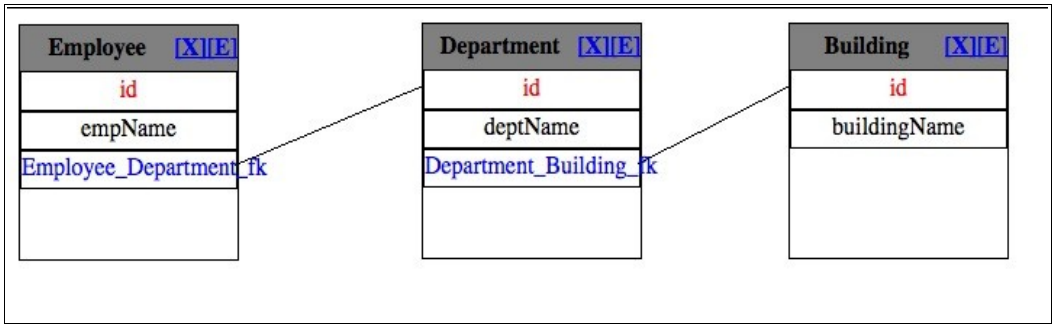

Figure 5: Automatic creation of foreign keys

The GUI has four panels. The first panel has the "Create new table", "Create QBE", and "Export tables to SQL" buttons. The second panel has the diagrams of all the tables created by users. The third panel has either the table creation form or the QBE form depending on the current mode of the application. The last panel serves as "note" that has all the SQL queries (the tables, foreign keys, and SQL queries creation). The "Create new table" and "Create QBE" operations are exactly the same as in Deliverable 2 and 3. The "Export tables to SQL" will dump the SQL queries (that are used to create the tables and

foreign keys) to the last panel. The complete GUI can be found in figure 6. In figure 6, we see that the user has used the QBE interface to select all employees that do not work in engineering department and do not work in MQ Hall building. The attributes that will be displayed are employee's name (alias: ename), department name (alias: dname), and building name (alias: bname). The results will be sorted by employee id in descending order, department id in ascending order, and building id in descending order. We also know that the user has used the 'Export tables to SQL' function because we can see the SQL queries in the bottom panel.

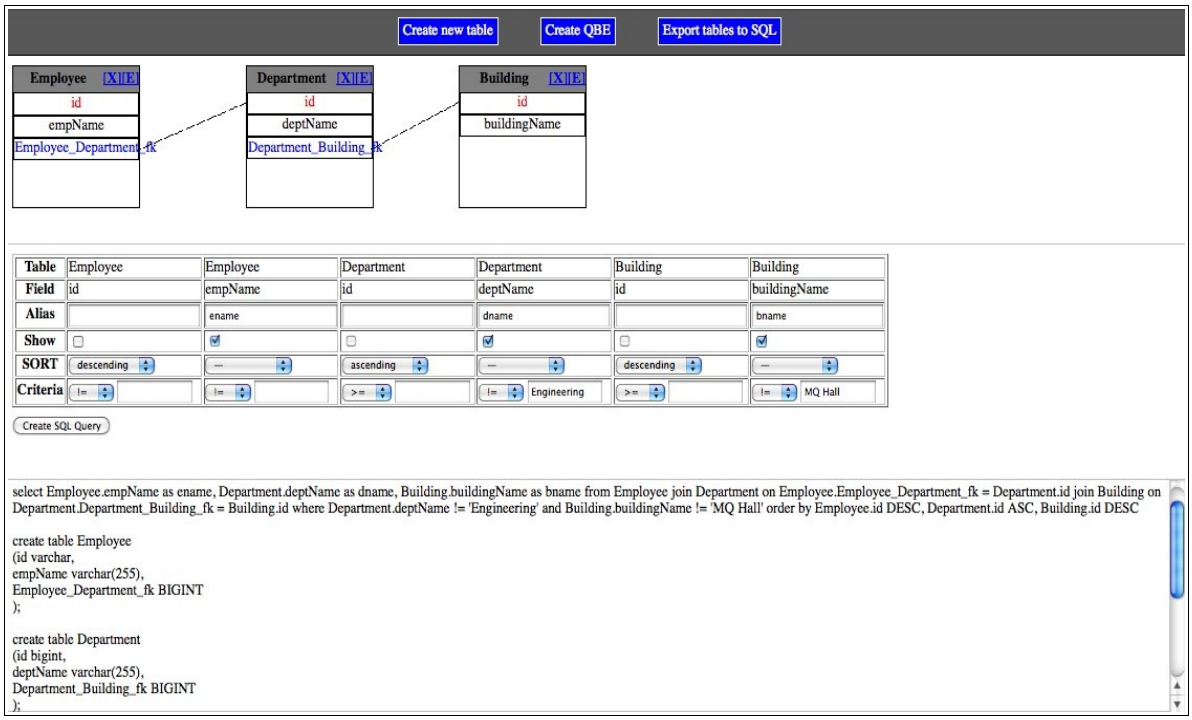

Figure 6: The complete GUI of deliverable 4

3. Conclusion

I have chosen (with the guidances of Prof. Pollett) the deliverables in such a way that they will support my actual project in CS 298. The first deliverable gave me chance to learn more about jQuery and its features. I have found that jQuery is simpler to use than Yahoo UI and it is also smaller in size. The first deliverable is very important because it used drag-and-drop feature, which will be used extensively in actual project. In the second deliverable, I had chance to work with Walterzorn Javascript Library to draw lines. The application could be used to create simple database tables. In the third deliverable, I could use the QBE to create complex SQL queries without having knowledge of SQL queries. In the last deliverable, I added a feature that will automatically add foreign key whenever I join two tables. I think this feature is really important because there are users who are not familiar with foreign key concepts. The application, that I develop in deliverable 4, will serve as base model for the actual application in CS 298.

There are many features that can be added to this application when I extend this project in Fall 2010 for my CS 298. Below are the tentative list of features that can/will be added to this application:
Automatically create the .php scripts for the model and controller components.
Import the SQL queries to various format of files.
Export from current project. With this feature, user will be able to save the current project and work on it later. This will probably require the creation of meta files to save the current states of the project.
Rework the GUI to make the application more user-friendly.
Include help & tutorial for the application.
Etc.

4. References

CakePHP Manual. Retrieved December 7th 2009 from CakePHP Website: http://book.cakephp.org/

jQuery documentation. Retrieved May 4th 2010, from http://docs.jquery.com/Main_Page

jQuery UI documentation. Retrieved May 4th 2010, from http://jqueryui.com/demos/

[2002] Database Systems: The Complete Book. Hector Garcia-Molina, Jeffrey D. Ullman, and Jennifer Widom. Prentice Hall. 2002

[2005] Database System Concepts. Avi Silberschatz, Henry F. Korth, and S. Sudarshan. McGraw-Hill. 2005