Lab 3: Model Classes

In this lab you will connect your server-side code to a MySQL relational database. You will fetch and modify records using a Model class that you will build to abstract your Data Access Layer.

Objectives & Outcomes

Successful completion of this activity will show that you can:

- Create Model classes to abstract data access.
- Query a relational database with PHP.
- Query a relational database with ColdFusion.
- Use query parameters to avoid SQL Injection attacks.

Level of Effort

This activity should take approximately 4h to complete. It will require:

- 30m Research
- 5m Prep & Delivery
- 3h Work

If you find that this activity takes you significantly less or more time than this estimate, please contact me for guidance.

Reading & Resources

Books

Object-Oriented Programming in ColdFusion
Chapters:
5-6
Pages:
111-152
Author:
Matt Gifford

Publisher:
Packt
Edition:
1 (2009)
These chapters on Data Access Objects and Gateways cover the most common patterns for server-side data access. You should work to become very comfortable with the patterns presented in these chapters, as you will see them repeated over and over in server-side coding.
PHP Objects, Patterns, and Practice

Chapters:
12-13
Pages:
264-314
Author:
Matt Zandstra

Publisher:
Apress
Edition:
3 (2010)
The section in Chapter 12 on the Business Logic Layer (pp 264-273) is the bare minimum requirement for this lab. The concepts in Chapter 13 are helpful, if overkill for projects this small, and so are not required for this assignment.

Videos

PHP App Architecture 1: A Quick Example (10m)
This video, the first in a series, shows how to build a really basic PHP app. But instead of building it perfectly from scratch, this series starts with old-skool procedural code and refactors it into more modern MVC-style code.
PHP App Architecture 2: Extract a Model Class (7m)
This video takes the procedural app from the previous video and extracts the data access into its own class.
PHP App Architecture 3: Extract a View Class (8m)
This video takes the procedural app from the previous video and extracts the presentation layer into its own View class.

Instructions

This lab is similar to the last one: you will develop data access solutions in both PHP and CFML, using the View classes you've already built. This lab will go in a `03_model` folder.

Development

To prepare for this assignment you should create an `ssl_users` table in your MySQL database. Use the same database that you are using for your DBS assignments. It should include a user ID, first and last names, email address, and password. You may include additional fields if you like, but these are the bare minimum.

The bulk of this lab will be building a complete user management solution. This should include `UserGateway` classes and additions to your `SiteView` classes that facilitate:

- **Create:** A form that allows you to add new users to the database.
- **Read:** A page with either all of the details for one user, listing of all users in the system, or with a form that allows searching for a specific user.

- **Update:** A form that is populated with the user data to be edited and updates the record on submission.
- **Delete:** A form that confirms, and then performs, the removal of a single user from the system.

This task can be approached in a number of different ways, depending on how you would like to build it. You could build a system that is all back-end: there is some super-user that is managing the user accounts. Or you could build a system that is all front-end: users register themselves and manage their own accounts. Or you could build some hybrid of the two. However you approach it, your system must allow for all 4 CRUD operations.

Here's the tricky part: Two of those CRUD operations must be done in PHP and two must be done in CFML. This means you will need two Gateway classes (`UserGateway.php` and `UserGateway.cfc`) and two View classes. If you feel like you could use the practice, you may do all four CRUD operations in each language, but the bare minimum is two in each and all 4 CRUD operations must be implemented between the languages.

Technical Details

You must ensure that the following points are covered:

- All query arguments are provided as query parameters, **not** as inline literals.

Above & Beyond

You have a few options beyond the bare minimum:

- Create a pagination system for your user listing.
- Build out all 4 CRUD operations in both languages.
- Use a DAO/Bean pattern instead of the Gateway pattern.

Git Commits & Video Reflection

The same rules about Git Commits and Reflection Videos apply for this Lab as for previous labs. Review the Lab 1 documentation to ensure you meet all of the required points. Your required questions are mostly the same, with small differences:

1. What didn't go as well as you had hoped? Where did you run into problems with the lab?
2. What went better or was easier than you expected? What went right?
3. **Why do you think we spend all of this time writing data access classes instead of just putting the SQL queries right into the same file as the HTML?**
4. How do you think what you did in this lab is going to help you in your future school work, career, or life?

Where should you be?

By now you should feel pretty comfortable with the basic syntax of CFML and PHP, even if you don't always remember the names of the functions you need. View templates should be second nature. Queries and Model/Gateway/Service classes may be a little fuzzy, but should at least make sense.

Deliverables

Make sure your source code, assets, and reflection video URL are committed to the repository. Push your changes to the server before the start of the next lecture.