Programmer Guide

Train Database Management System™

Table of Contents

Assumptions about the Programmer

In order to make meaningful changes to this product, you will need an intermediate-level understanding of HTML, Javascript, PHP,and mySQL. Therefore, the writer of this guide assumes that you have these requisite skills. If you do not, the "Helpful Links" section at the end of this document will refer you to useful tutorials.

Recommended Tools

We recommend that you use a proper web development IDE for any modifications to this product. Aptana and IntelliJ both provide full support for the languages used, and both have free versions available online. Find links to their homepages in the "Helpful Links" section at the end of this document.

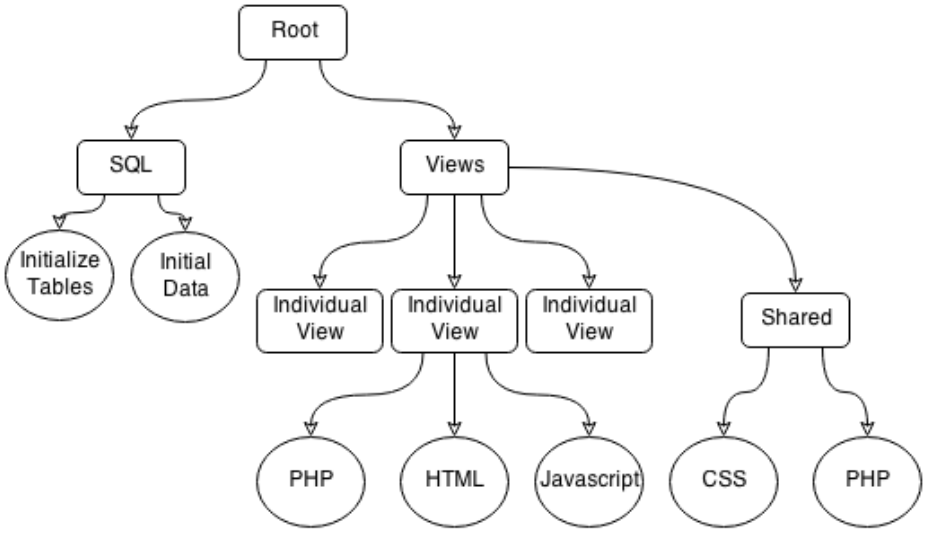
We also recommend using a modern browser with a powerful developer's console. Our product was tested and built with Mozilla

Firefox and Google Chrome. They are both superb browsers. Again, find links to their respective homepages at the end of this document.

Code Directory Structure

At the root of our code base, there are two folders. The SQL folder contains the SQL scripts used to initialize the database and fill it with sample values. The Views folder has the code to display and support each of our product's view.

Each view is in its own separate folder. There is also a folder titled "Shared" for resources that are common across multiple views. The stylesheet and often-used PHP documents are stored here. Each view folder will have relevant Javascript, HTML, and PHP files. Our directory structure is modeled by the figure below.



Conventions

Database:

Our tables are named such that the first letter of each word is capitalized. Our attributes follow suit, except the first word is not capitalized.

Our database is compliant with Boyce-Codd Normal Form, and any modifications to the database schema should attempt to preserve this. Any "isa" relationships should be decomposed using the object-oriented approach. This means that each table must contain a full set of its parents' attributes. The parents are thus unnecessary and are not added to our database.

Views

HTML, PHP, CSS, and Javascript code are kept in separate files. This is done to separate our concerns. Doing so isolates functional areas of our code for easier location and comprehension. All HTML files also

reference a common stylesheet and jQuery library located in the shared folder.

Files and Folders

Our folders are given capital names that accurately describe their contents. In our code filenames, words are left un-capitalized and are separated by underscores.

Description of Front/Middle/Back End

The product's back end is a mySQL database containing a number of relations. The front is made up of the styled HTML pages from each view. Each page shows a different selection of information from the database. The middle end is the Javascript and PHP files used in each view. These are responsible for transporting data from the back end to the front end and vice versa. Some pages make heavy use of Javascript, some pages use PHP exclusively.

Understand that the front end views do not accurately reflect the structure of the back end. That is, each view does not correspond to one table in the database. Rather, views display combinations of multiple tables with some columns removed. This is an intentional, well-reasoned separation. The databases' tables and attributes are organized for data integrity while the front end is directed towards user convenience and comprehension.

Our middle end is mostly concerned with SCRUD operations. SCRUD is an abbreviation for "search, create, read, update, delete". These operations may be triggered by the page loading or by user action. If the user modifies data that is currently being displayed, that display should be updated to reflect their modification.

How the Code Works

The following is a step-by-step breakdown of code execution in a generic view. We will assume this view has a full range of functionality, allowing a user to view, filter, create, and delete rows of data. For a more detailed explanation of our code, please consult the commenting and documentation within our code files.

Loading and Displaying a View (Javascript)

1. The user navigates to a view page. The page renders itself, linking in a stylesheet and Javascript file.
2. The page finishes rendering, which triggers an onload event listener in the Javascript file.
3. The event listener sends an AJAX request to whichever PHP file is responsible for fetching the data featured in the view.
4. The PHP file receives the request and attempts to connect to the database.

5. Once connected, the PHP code issues a mySQL query to the database to obtain whatever data we would like to initially display.
6. The query returns a result object which the PHP code then casts to JSON.
7. The PHP code then echoes the JSON object, and the AJAX call in the Javascript file returns this object.
8. The Javascript file iterates through the rows in the JSON object and appends each as a HTML table row to the view.
9. The Javascript code terminates, the user now sees a selection of the database.

Loading and Displaying a View (PHP only)

1. The user navigates to a view page containing an imbedded PHP request.
2. On the server side, the PHP code executes, fetching information from the database and writing it as HTML.
3. The page is sent to the client, who can then see the information fetched from the database.

Inserting from a View (Javascript)

1. The user enters all the data required to create a new tuple into the view. Their input may take the form of drop-down lists selections, text field entries, etc.
2. The user presses a submit button, which calls a Javascript function.
3. The Javascript function retrieves the user's input and packs it as a JSON object.
4. The function then makes an AJAX call containing the JSON object to whichever PHP file is responsible for inserting new data.
5. The PHP code receives the call and connects to the database.
6. If the input data must be massaged to match mySQL's accepted data formats, the PHP code does this here.
7. The PHP code construct an insert query from the passed data and (possibly) massaged data.
8. After the AJAX call completes, the data table displayed in the view is reloaded. See steps 3-9 of **Loading and Displaying a View** (Javascript).

Inserting in a View (PHP only)

1. The user enters all the data required to create a new tuple into the view. Their input may take the form of drop-down lists selections, text field entries, etc.

2. The user presses a submit button, which posts the user's input to the current page.
3. The server executes the PHP code on this page that handles parsing posted data. The code assembles an insert query.
4. The query is executed and server-side code execution finishes.
5. The page loads on the client's side.

Deleting from a View (Javascript)

1. The user clicks on an "X" beside the tuple they would like to delete, which calls a Javascript function.
2. The Javascript function retrieves the primary key of the tuple and packs it as a JSON object.
3. The function then makes an AJAX call containing the JSON object to whichever PHP file is responsible for performing deletions.
4. The PHP code receives the call and connects to the database.
5. The PHP code construct a delete query from the passed key.
6. After the AJAX call completes, the data table displayed in the view is reloaded (see steps 3-9 of **Loading and Displaying a View**).

Deleting from a View (PHP only)

1. The user clicks on an "X" beside the tuple they would like to delete, which posts the tuple's ID to the current page.
2. The server executes the PHP code on this page. The posted value is used to assemble a deletion query.
3. The query is executed and the server-side code execution finishes.
4. The page is loaded on the client's side.

Anticipated Changes

Adding/Dropping Tables

If the product is not yet deployed, add/drop operations are easily performed by modifying the initialization mySQL script found in the SQL directory off root directory. Otherwise, an add/drop operation should be done via mySQL transaction.

Take care when deleting a table. A deletion will probably have a cascading effect across the entire product. Other tables may have foreign keys to the target table and will require alteration. Please refer to the database's relational diagram to better understand how our tables are interconnected. Each table you alter may be represented in multiple views. Each view's PHP, HTML, and Javascript files may require alteration to reflect the dropped attributes.

There are several code areas in the view that are most vulnerable to change. These include the input elements in the HTML page, input value retrieval in the Javascript file, the query string in the PHP file, and the Javascript loop that appends query results to the HTML page.

Adding/Removing Attributes

If the product is not yet deployed, alterations are easily performed by modifying the initialization mySQL script found in the SQL directory off root directory. Otherwise, an add/drop operation should be done via mySQL transaction.

Take care when deleting an attribute. If you wish to delete a primary key attribute, you must insert or assign a new primary key. Also be wary of deleting foreign keys. Each table you alter may be represented in multiple views. Please refer to the database's relational diagram to better understand how our tables are interconnected. Each view's PHP, HTML, and Javascript files may require alteration to reflect dropped attributes.

There are several code areas in the view that are most vulnerable to change. These include the input elements in the HTML page, input value retrieval in the Javascript file, the query string in the PHP file, and the Javascript loop that appends query results to the HTML page.

Adding/Removing Views

Removing a view is as simple as deleting or hiding its HTML page. Adding a view is a more involved process, as it requires creating new HTML, Javascript, and PHP pages. To create a new view, you should begin by carefully examining the code of a view similar to what you desire. Then either write your new view from scratch, or modify a copy of a similar view.

When creating a new view based on another, there are a few areas of code that will probably require the most significant changes. These include the input elements in the HTML page, input value retrieval in the Javascript file, the query string in the PHP file, and the Javascript loop that appends query results to the HTML page.

Modifying Views

- To change the styling of the view, alter the CSS file located in ~root/Views/Shared.
- To change the structure of the page, (but not its functionality) alter its corresponding HTML file.
- To remove functionality from a page, delete from the HTML file any buttons or fields related to that functionality. Find the Javascript function triggered by the related button and determine whether that code is used by any other part of the view. If not, delete or comment

out that code. If the Javascript code makes a request to a PHP file that is used nowhere else, delete or hide that file.

- To add functionality to a page, you must add a trigger button to the HTML, as well as input fields if the action requires user input. Add a Javascript function to listen for clicks on your new button. Your new functionality goes here. If your must fetch additional data from the database, do so with a new PHP file.

Adding/Removing Constraints

Constraint modifications should be done via a mySQL transaction. Adding a constraint is straightforward. Be careful when dropping constraints. Dropping a foreign key constraint and then inserting or deleting from affected relations may have unintended consequences for the JOIN operations in views' PHP files.

Helpful Links

Google Chrome is a standards-compliant web browser with a full-featured developer console.

http://www.google.ca/chrome/

Mozilla Firefox is another excellent browser that is also standards-compliant and has a good developer console.

https://www.mozilla.org/en-US/firefox/desktop/

W3Schools is a learning center for web development. It contains extensive tutorials on HTML, CSS, Javascript, PHP, and SQL.

http://www.w3schools.com/

PHP's official site contains downloads and documentation.

http://php.net/

MySQL's official site contains downloads and documentation.

http://www.mysql.com/