Hamden Politics - Influence Intelligence

csc545 Summer 2016

Hamden Politics - *Influence Intelligence* uses a framework built for abstracting the data access layer in order to quickly switch between database back ends. A factory pattern is used to fetch a generic database object which interacts with the database layer via Data Access Objects. Methods that are used throughout the applications are declared in the database interface and implemented in the corresponding database abstraction class.

- People
    - DAO - dbo\Person.php
    - Interface - dbo\PeopleInterface.php
    - DB - dbo\MySQLPeopleDatabase.php
    - DB - dbo\MongoPeopleDatabase.php
- Organizations
    - DAO - dbo\Organization.php
    - Inerface - dbo\OrganizationInterface.php
    - DB - dbo\MySQLOrganizationDatabase.php
    - DB - dbo\MongoOrganizationDatabase.php
- Events
    - DAO - dbo\Event.php
    - Interface - dbo\EventInterface.php

    - DB - dbo\MySQLEventDatabase.php
    - DB - dbo\MongoEventDatabase.php

> *For the sake of developement speed we chose to use an MVC pattern. Using an MVC pattern required writing some MVC framework classes that enable us to quickly re use code for the different pages that are involved in the application. More info on the other pieces of the application framework can be found in comments throughout the code.*

MVC Details

- Model Layer
    - The *M* in the MVC pattern is implemented in the above database abstraction layer.
- View Layer
    - The *V* in the MVC pattern is implemented using PHP templates in the *templates* folder. These templates are rendered using the “render” method of the “Controller” class. The “render” method requires the template name and a set of variables that should be available to the template. This method sets the scope of each template to the scope of the method and makes arguments available in the template as PHP variables. This type of view layer can be seen in popular PHP frameworks such as Codeigniter and CakePHP
- Controller Layer

    - The *C* in the MVC pattern is implemented using a “Controller” class. Each controller defines methods that can be called through web requests. The web requests are mapped to a Controller and a Controllers Method via the “Router” class.
- Routing
    - Routing is done using a *.htaccess* file that rewrites all web requests to the *index.php* file. The *index.php* file loads the configuration by including the “bootstrap.php” file and instantiating an instance of the router class as well as calling the “router” method. Routes are done automaticly by looking at the path of the web request. A typical web request would look like **/:controller/:method/:arg1/:arg2**. Public methods that are created in an instance of a Controller are available as web requests that can be used through a browser. If the router can not find the correct method the router will include the “notfound.html” file to show a404 message as well as send back a 404 response header to indicate that the page was not found.

Render Method in the Controller class

```
    protected function render($template, $args = array())
     {
         extract($args);
         unset($args);
         include_once $template;
     }
```

.htaccess file

```
RewriteEngine on
RewriteCond %{REQUEST_FILENAME} !-d
RewriteCond %{REQUEST_FILENAME} !-f
RewriteRule . index.php [L]
```

Folder Structure

- cssjs - Contains all CSS and JavaScript files
    - These files are added to the **head.php** template by adding the file name to the **$css_files** or **$js_files** class properties for a controller. Each method can add a page specific stylesheet or script file by directly adding the filename to the class property.
- images - Contains any images that we added for the application not including images used by “bootstrap css framework”
- src - PHP Source files
    - controllers - Contains all of the applications Controller classes. Controllers must be in this folder and use the naming convention “:entity:Controller.php”
    - dbo - Contains Database Objects. DAO’s, Interfaces, and Database Abstraction classes.
    - lib - Contains classes used to enable the MVC pattern and any other helper classes.
- templates - Contains all PHP templates

bootstrap.php

The **bootstrap.php** file is the first file included in the index.php file.

This file is used both as a configuration file and a autoloader file.