

This project turns your DVD library program into a web application. The application provides read-only behavior. Users may view, search and sort the database, but they may not add, delete or edit movies. You have already written most of the code for this assignment. The challenge for this project is to understand how to access and validate web-supplied data and to deploy the application on a web server.

Section I: The Framework

You will be given a skeleton application organized in the same way as the previous assignment (i.e., as MVC). The model is complete; you do not need to modify it. The view converts database information to HTML strings; you do not need to modify it. The controller takes input from the user (via the web) and instructs the view to send the client a web page with the appropriate results. The controller is **not** complete; you will need to complete it. Section III contains the instructions for completing the controller. The course website hosts some **(automatically generated) documentation** for the framework.

Section II: Deploying Your Application on the Web

You must get your application to execute on the UTCS Web server. Here are the steps you need to follow to get started:

1. Log in to UTCS. In your home directory create the directory `public_html/cs105` by executing the command `mkdir -p public_html/cs105`.
2. Change to this directory with the command `cd public_html/cs105`.
3. Unpack the project files into this directory with the command `tar -xzf /u/ben/public_html/teaching/cs105/hw/pa4w.tgz`.
4. Now you can use the application by browsing to `http://www.cs.utexas.edu/~username/cs105/pa4.cgi` where `username` is your UTCS login name. The web page will be password-protected, to prevent outsiders from accessing your application while you develop it. The user name and password are posted on the course discussion board.
[Note: This is not a very strong form of access control. You should take down your web application after you are done developing it.]

You can sort the database with a URL like `http://www.cs.utexas.edu/~partner/cs105/pa4.cgi?sort=upc` and search the database with a URL like `http://www.cs.utexas.edu/~partner/cs105/pa4.cgi?stars=Clooney`. The empty application prints errors for both of these operations.

Section III: Completing the Application

You must modify the controller, to validate user-supplied data and to provide search functionality. To do so, you must override methods `validate` and `filter` in the class `MyHTMLMovieController`. The remainder of this section describes the controller in more detail and provides the requirements for implementing `validate` and `filter`.

The controller consists of the files in module `movie.lib.controller`, plus the file you must complete: `MyHTMLMovieController.py`. This file contains a class `MyHTMLMovieController` that inherits from `movie.lib.controller.HTMLMovieController.HTMLMovieController`. This class has three data members:

- `model`: A reference to the database.
- `view`: A reference to the view.
- `form`: A reference to an instance of `cgi.FieldStorage`, which contains the web form data.

Class `HTMLMovieController` provides the following methods:

display. This method prints HTML-encoded movie data. You do not need to modify this method. However, the method relies on your implementations of `validate` and `filter`.

validate. This method validates the user form. You must override its definition in `MyHTMLMovieController`. The `validate` method accesses the controller's `form` data member and must implement the following behavior:

If the form is `None`, raise a `MovieException` with the error message "Missing form".

A valid form is allowed to have a key `"sort"`, or it may have a key whose name matches any name in the model's `schema.ATTRIBUTES` data member (hereafter referred to as "valid attribute names"). If it has any other keys, raise a `MovieException` with the error message "Invalid data".

If the form has a key `"sort"`, then the key's value must be a valid attribute name; otherwise `validate` should raise a `MovieException` with the error message "Unknown sort attribute".

If the form has a key that is a valid attribute name, then the key's value must have the appropriate format for the given attribute. An invalid format should cause the method to raise a `MovieException` with an appropriate message. You can reuse your code from PA2 to validate a schema attribute's value.

The goal of `validate` is to allow URLs like these:

`http://.../pa4.cgi` or `http://.../pa4.cgi?sort=upc&year=1999` or `http://.../pa4.cgi?stars=Clooney`

but to disallow URLs like these:

`http://.../pa4.cgi?unknown=1` or `http://.../pa4.cgi?year=abc`

filter. This method populates the database with a list of movies that satisfies user-supplied search criteria. You must override its definition in `MyHTMLMovieController`. The `validate` method accesses the controller's `form` data, and performs the following operation for each key in the form that is a valid attribute name:

1. It calls the model's `simpleSearch` method, passing the attribute name and the form's value for that name. This will return a list of movies that match the search criteria.
2. It calls the model's `populate` method, passing the results from the previous step.

You can assume that the value for a given attribute name is properly formatted, because the application always calls `validate` before `filter`.

What To Turn In

Turn in `MyHTMLMovieController.py`. If you are working in a team, you must also submit a project report in the file `report.txt`. Each submitted file should have names and EIDs in a comment at the top of the file. Be sure to read the **Programming Assignment Overview** for instructions on how to prepare your report and submit your work. The *homework-name* for this assignment is: `pa4w`.