Assignment 6 Kevin Bacon

What you need to do

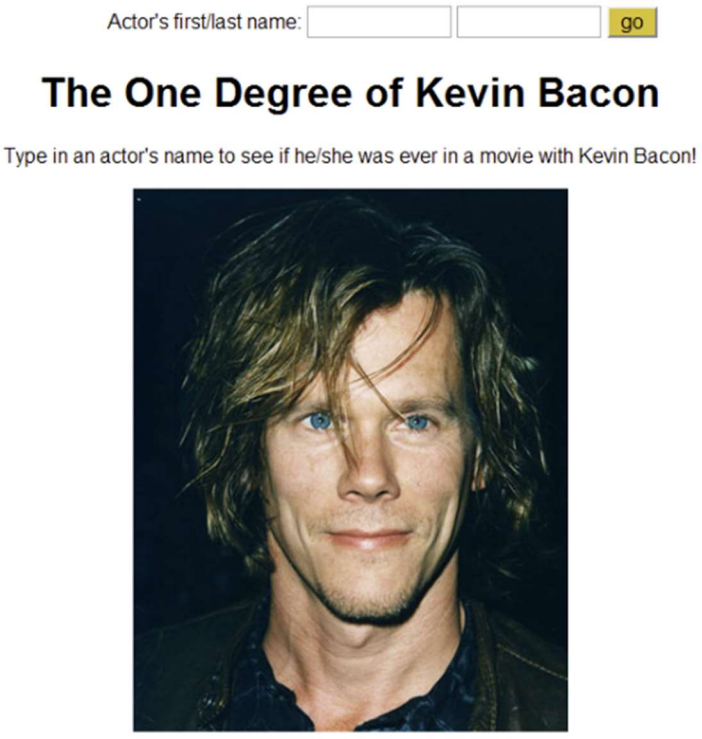
- Connect to a database using PHP
- Develop a SQL query to present desired data based on user-provided input
- Display the data from the SQL query within an HTML document
- Handle user input securely

Requirements

The Six Degrees of Kevin Bacon is a game based upon the theory that every actor can be connected to actor Kevin Bacon by a chain of movies no more than 6 in length. (Most, but not all, can reach him in 6 steps. 12% of all actors cannot reach him at all.)

Your task for this assignment is to write the HTML and PHP code for a web site called *MyMDb* that mimics part of the popular IMDb movie database site. Your site will show the movies in which another actor has appeared with *Kevin Bacon*. The site will also show a list of all movies in which the other actor has appeared.

The front search page *mymdb.php* has a form where the user can type an actor's name.



When the form is submitted, the results are shown on search.php.

The One Degree of Kevin Bacon

Betsy Palmer and Kevin Bacon were together in:

No.	Title	Year
1	Friday the 13th	1980

All of Betsy Palmer performances:

No.	Title	Year
1	Super Secret Movie Rules: Slashers	2004
2	Return to Crystal Lake: Making 'Friday the 13th'	2003
3	Joan Crawford: The Ultimate Movie Star	2002
4	Fear: Resurrection, The	1999
5	Unveiled	1994
6	Still Not Quite Human	1992
7	Columbo: Death Hits the Jackpot	1991
8	Goddess of Love	1988
9	Windmills of the Gods	1988
10	Friday the 13th: The Final Chapter	1984
11	Friday the 13th Part 3: 3D	1982
12	Isabel's Choice	1981
13	Friday the 13th Part 2	1981
14	Friday the 13th	1980
15	"Number 96"	1980
16	"Knots Landing"	1979
17	"Girl Talk"	1963
18	Last Angry Man, The	1959
19	It Happened to Jane	1959
20	True Story of Lynn Stuart, The	1958
21	Tin Star, The	1957
22	"What's It For"	1957
23	Queen Bee	1955
24	Long Gray Line, The	1955
25	Mister Roberts	1955
26	Marty	1953
27	"Today"	1952
28	"I've Got a Secret"	1952
29	"Masquerade Party"	1952

Files to turn in

mymdb.php, the front signup page (partial skeleton provided)
bacon.css, the CSS styles for both pages
Bacon.js the JavaScript code for both pages [In case that you use JavaScript in your implementation, otherwise, this file is not needed.]
search.php, the search results page
common.php, any common code that is shared between pages *("optional")*

[Note that the above file list provides some guideline, which will be explained below, on what code you need to implement. But I would allow the flexibility for you to structure code as you wish as long as your implementation satisfies the assignment requirements.]

Appearance Constraints (both pages):

Your mymdb.php and search.php must **both** match certain appearance criteria listed below. Beyond these, any other aspects of the page are up to you, so long as it does not conflict with what is required.

- The same title, and links to the same CSS and JavaScript resources.
- A common stylistic theme, and an overall non-trivial number of styles, such as fonts, colors, borders, and layout. As much as possible, set up your styles to look correct on a variety of systems.
- A form to type an actor's first/last name and search for matching actors in the database.
- A central area of results/info. In mymdb.php this area contains content of your choice, including at least one image of Kevin Bacon (or your central actor) and some text about the site. Be creative! In search.php this area contains the actor's movies and all movies in which the actor starred with Kevin Bacon.
- The search.php page should contain a link back to mymdb.php if the user wants to start over.

NOTE: With so much in common between the two pages, it is important to find ways to avoid redundancy. You should use the PHP *include* function with a shared common file included by both pages.

Front Page, mymdb.php:

The initial page, *mymdb.php*, allows the user to search for actors to match against Kevin Bacon. A skeleton is available for you to download on the D2L course web site; you may modify it in any way you like, subject to the constraints in this document.

The form on the page must contain two text boxes that allow the user to search for an actor by first/last name. The end goal is to submit to *search.php*, but it is possible that the name the user types (such as "Will Smith") will match more than one actor. Some names match no one in the database.

To address this problem, you can use the Actor ID web service, which is explained below, to find out which actor matches a given name. You don't need to use the web service if you'd

rather write that functionality yourself. But either way, your page needs to map a name to a single actor.

Actor ID Web Service, actorid.php:

The provided *actorid.php* maps names to IDs. [Note that the file is not included in the provided code package for you to start the assignment. But the service is available for you to check actorid as explained.] For example, to search for partial names "ad"and "pitt":

http://u.arizona.edu/~milazzom/cscv337/sp19/hw6/actorid.php?**first_name=brad**&**last_name=pitt**

This query would return the following XML output. Notice that it contains the actor id:

```
<actor id="376249" first_name="Brad" last_name="Pitt" appearances="59"/>
```

The service issues a 404 error if no actor is found. By default, the service queries a small subset of the database, which is useful for development purposes. For your code, in case that you'd like to use the web service in your implementation, you must also include another query string parameter (imdb=true) to allow the service to query against the full database.

http://u.arizona.edu/~milazzom/cscv337/sp19/hw6/actorid.php?first_name=ad&last_name=pitt**&imdb=true**

Movie Search Page, search.php:

The search.php page performs two queries on the imdb database to show a given actor's movies. To query the database using PHP's mysql_functions, you need to connect to the database using the database credentials (user name + password) I presented at D2L following this assignment link.

The database has the following relevant tables. (The roles table connects actors to movies.)

table	columns
actors	id, first_name, last_name, gender
movies	id, name, year
roles	actor_id, movie_id, role

Your page should perform the following two queries. For each query, you will need to use a **join** between several tables of the database.

A query is to find a complete list of movies in which the actor has performed, displaying them in an HTML table. You may assume that any actor in the actors table has been in at least one movie.

Hint: You will need to join the actors, movies, and roles tables, and only retain rows where the various IDs from the tables (actor ID, movie ID) match each other and the name or ID of your actor.

A query to find all movies in which the actor performed with Kevin Bacon. These movies should be displayed as a second HTML table, with the same styling as the first. This is the hard query and should be done last. If the actor has not been in any movies with Kevin Bacon, don't show a table, and instead show a message such as, "Borat Sagdiyev wasn't in any films with Kevin Bacon."

This query is bigger and tougher because you must link a pair of performances, one by the submitted actor and one by Kevin Bacon, that occurred in the same film. Do not directly write any actor's ID number anywhere in your PHP code, not even Kevin Bacon's.

Hint: You will need to join a pair of actors (yours and Kevin Bacon), a corresponding pair of roles that match those actors, and a related movie that matches those two roles. The query could join 5 tables in the FROM clause and contains 3 conditions in its WHERE clause. However, if you have the searched actor id and Bacon’s id available first, you could simplify the joining over only 3 tables.

The data in both tables should be sorted in descending order by year, breaking ties by movie title. The tables have three columns: A number for each movie, starting at 1; the title; and the year. The columns must have styled headings, such as bolded. The rows of the table must have alternating background colors, sometimes called "zebra striping." (Use PHP to apply styles to alternating rows.) For example:

No.	Title	Year
1.	Private War, A	2005
2.	Reefer Madness	2004
3.	Blind Horizon	2004
4.	Churchill: The Hollywood Years	2004

It is not acceptable to perform query filtering in PHP. Your SQL queries must filter the data down to only the relevant rows and columns. For example, a bad algorithm would be to query all of the actor's movies, then query all of Bacon's movies, then use PHP to loop over the two looking for matches. Each of the two queries above should be done with a single SQL query to the database.

Developing Your Queries

You may work on designing your SQL queries for this assignment using MySQL client application such as the MySQL Workbench or the MySQL command-line client. Database connectivity information will be posted to our D2L course site. If you execute a query that is taking a long time to respond, please cancel it using the Control_C key combination. To develop your queries, please perform them against the imbd-small, which is the small, subset of the full, database instead of the imdb full database to ensure performance is acceptable for other users that might by querying the database simultaneously. (Reminder: Your PHP code should query the full database.)

Turn-in

Create a zip file named <UA Net ID>_hw6.zip containing all files you created or modified for this assignment and submit it to Assignment 6 drop box on our D2L site. You must make your site accessible from your password protected UA web hosting space and submit the URL as well as the password for user lxu to access the pages in your D2L submission notes.

Rubric

This assignment is worth 100 points, broken down as follows:

- 40 points: Upon submitting a first/last name of an actor/actress, two tables are displayed, one for the movies for the specified actor/actress and the other for the movies where Kevin Bacon also performed in the same movie.
- 10 points: Tables displaying the results have styled column headers and alternating background colors for every other row.
- 10 points: Results are retrieved from search.php
- 20 points: Code connects successfully to MySQL
- 20 points: Code performs SQL queries as required