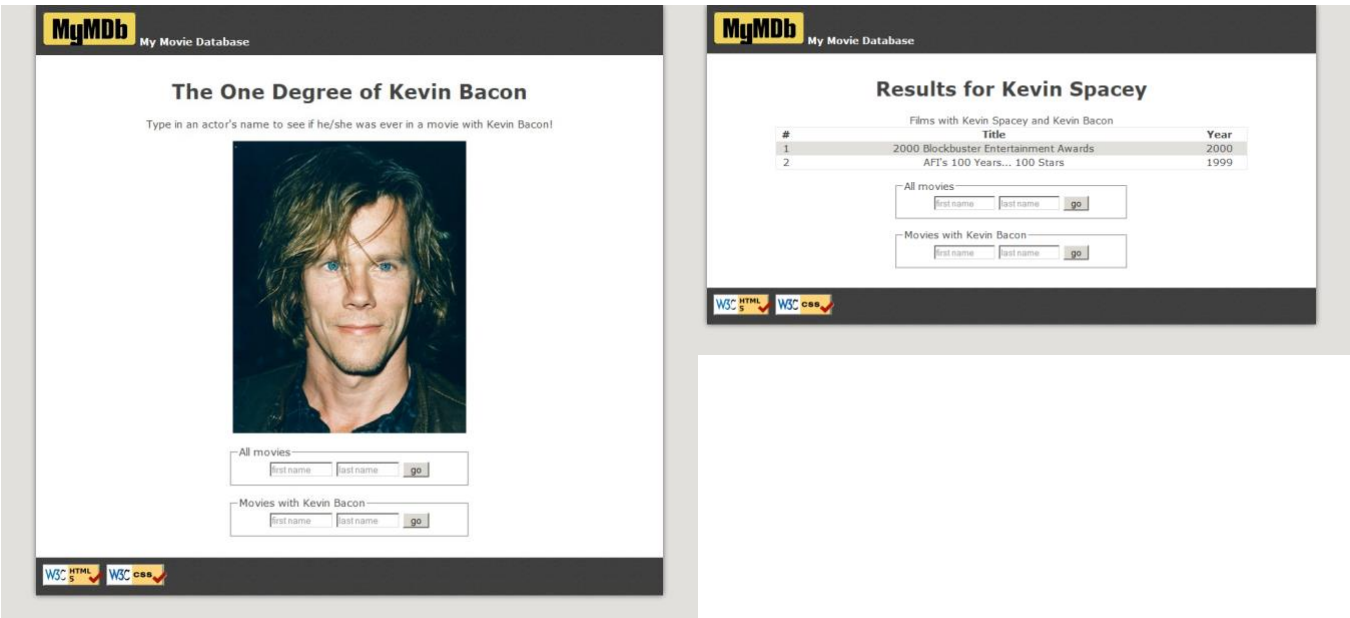
Università degli Studi del Piemonte Orientale
Dipartimento di Scienze ed Innovazione Tecnologica
DiSIT

Esercizio 3: Six degrees of separation
(PHP, Ajax e DBMS MySQL)

Testo dell'esercizio

Lo scopo di questo esercizio è quello di farvi prendere pratica con i database relazionali e, soprattutto, su come connettervi ad essi attraverso l'utilizzo del linguaggio PHP.



Il contesto

La teoria dei Sei Gradi di Separazione è stata introdotta dallo psicologo sociale Stanley Milgram negli anni '60, ed è basata sull'idea che, presa una coppia di individui (per i sociologi: "attori") qualsiasi in una rete sociale, allora ci sarà un cammino minimo medio di lunghezza 6 tra di loro. Questa teoria è stata (inconsapevolmente o meno) applicata per un periodo di tempo ad un gioco diffusosi negli Stati Uniti che tendeva a dimostrare che Kevin Bacon fosse l'attore più "importante" nella rete sociale delle collaborazioni cinematografiche e che egli fosse distante da ogni altro attore di quella rete con cammini di lunghezza non superiore a 6. Questa "diceria" ha dei fondamenti scientifici e matematici seri (ad esempio usando il cosidetto modello "small-world" [Albert-Laszlo Barabasi, Reka Albert, 1999]) e parte del principio dei 6 gradi di separazione è effettivamente validabile su qualsiasi rete sociale, ma è anche vero che Kevin Bacon non sia più centrale di altri attori nella rete sociale: è molto probabile che esista un cammino molto brevi per qualsiasi coppia di attori.

Il sito proposto per questo Esercizio 3 si ispira a questa idea un po' bizzarra ed un po' vera che ruota attorno a Kevin Bacon.

Il sito

L'obiettivo di questo esercizio di laboratorio è quello di sviluppare un sito, chiamato **myMDB**, usando tecnologie HTML/CSS/PHP/MySQL/Ajax replicare in parte le funzionalità del popolare IMDB. Questo sito serve a mostrare i film in cui Kevin Bacon ha recitato un ruolo in collaborazione di un altro attore scelto dall'utente. Se preferite, potreste anche scegliere un altro attore come ipotetico centro della rete delle collaborazioni cinematografiche, l'importante è che questo attore sia presente nel database fornito come materiale dell'esercizio e che abbia un certo numero di connessioni con altri attori, ovvero abbia recitato in molti film.

I seguenti file li potete trovare sulla pagina DIR del corso di Metodologie per il web:

- top.html, contenente qualche riga HTMLche dovrebbe essere inclusa all'inizio di ogni altra pagina
- bottom.html, contenente qualche riga HTML che dovrebbe essere inclusa alla fine di ogni altra pagina
- index.php, la pagina principale che serve ad accogliere l'utente

La pagina principale index.php contiene due form che possono essere usati dall'utente per digitare il nome di un attore (diverso da Kevin Bacon). L'utente può cercare un film dove questo secondo attore è apparso (che viene passato alla pagina search-all.php), oppure ogni film dove sia l'attore cercato che Kevin Bacon siano apparsi (passando alla pagina search-kevin.php).

Questi che seguono sono i file che dovete scrivere:

- search-all.php, la pagina che mostra i risultati della ricerca di tutti i film, dato un determinato attore
- search-kevin.php, la pagina che mostra tutti i risultati della ricerca di tutti i film dove appaiono sia il dato attore che Kevin Bacon
- common.php, contiene codice di servizio comune che può essere usato da tutte le varie pagine del sito
- bacon.css, lo stile CSS da usare nelle varie pagine

La pagina principale, index.php

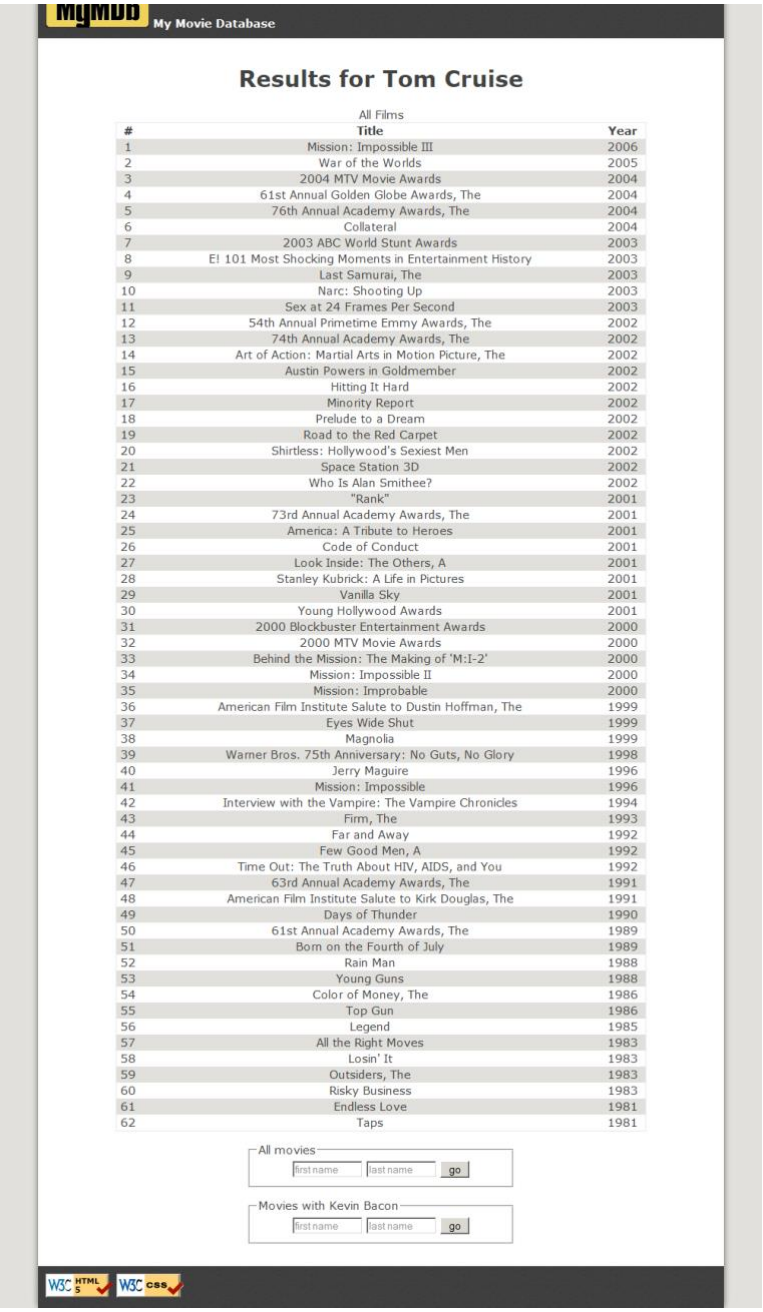
La pagina principale, index.php, consente all'utente di cercare gli attori. Questo file è già messo a disposizione su DIR; potete modificarlo come volete, o potete lasciarlo cosí com'è. I moduli presenti nella pagina contengono due campi testo che permettono all'utente di cercare l'attore tramite la coppia nome/cognome.

- firstname per il nome di battesimo dell'attore
- lastname per il cognome dell'attore

Pagine di ricerca dei film, search-all.php e search-kevin.php

Le due pagine "search" eseguono delle query nel database imdb (già fornito su DIR precedentemente) per mostrare i film di un dato attore. Eseguite le query nel database usando le librerie **PDO di PHP cosí come abbiamo visto a lezione**. Connettetevi al database usando le vostre credenziali.

I dati in entrambe le tabelle dovrebbero essere ordinati per anno in senso decrescente e nel caso di film distribuiti nello stesso anno, questi dovrebbero apparire ordinati per titolo in senso crescente. I dati possono essere visualizzati in HTML tramite <table>, considerando tre colonne: un numero d'ordine che parte da 1; il titolo del film; l'anno di distribuzione. Le colonne devono avere delle intestazioni, alle quali applicare un qualche stile, ad esempio il grassetto. Un esempio di risultato è mostrato a dx.



Database e query

Il database ha tra le altre, le seguenti tabelle (la tabella roles è un'associazione tra le entità attori e film):

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

Le pagine 'search' eseguono le seguenti query. Per qualche query dovrete usare un join su diverse tabelle del db.

1. **search-all.php - restituisce la lista di tutti i film di un dato attore**: con una query cercate la lista completa dei film nel quale un attore recita una parte, mostrando i risultati in una tabella HTML costruita dinamicamente. Se l'attore non esiste nel database, non create la tabella, ma invece mostrare un messaggio tipo: "Actor ***Borat Sagdiyev*** not found." Se invece l'attore è presente nel db, potete assumere che ci sia almeno un film in cui l'attore ha recitato.
    a. Per scrivere la query corretta, avrete bisogno di fare un join almeno tra le tabelle actors, movies e roles. Tenete solo le righe dove le ID delle varie tabelle combacino e dove l'attore è quello cercato

2. **search-kevin.php - restituisce la lista di tutti i film che il dato attore ha fatto insieme a Kevin Bacon.** Questi film dovrebbero essere mostrati in un'altra tabella HTML, con lo stesso stile della prima. In questo caso la query è leggermente più complicata e vi consiglio di scriverla dopo l'altra. Se l'attore non esiste nel db, non mostrare una tabella, ma un messaggio tipo: "Actor ***Borat Sagdiyev*** not found.". Se invece l'attore è presente nel db, ma non ha fatto neanche un film con Kevin Bacon, allora scrivete un messaggio tipo: "*Borat Sagdiyev* wasn't in any films with *Kevin Bacon*."
    a. dovete eseguire un join su una coppia di attori (quello che l'utente ha scelto e Kevin Bacon), una coppia di ruoli relativi a quegli attori, ed un film che è associato a quei ruoli. Se viene fuori una join tra 5 tabelle e tre condizioni WHERE, non vi spaventate: potrebbe essere la query giusta.

3. **entrambe le pagine - trova l'ID per il dato attore**: Una cosa che rende questo progamma leggermente complicato è il fatto alcuni attori hanno lo stesso nome. I dati imdb risolvono questa ambiguità dando a questi attori dei nomi leggermente diversi, ad esempio: "Will (I) Smith" vs. "Will (II) Smith". L'utente probabilmente non capirà la differenza tra i due ed è legittimo aspettarsi che digiterà semplicemente "Will Smith", aspettandosi che il programma farà poi la cosa corretta. Ma se il codice cerca esattamente "Will Smith" nel database, non lo troverà!
    a. Per risolvere questo problema, avete bisogno di scrivere una terza query che cerca il **miglior risultato relativo** al nome dell'attore che è stato digitato. Questa query cerca e restituisce l'ID dell'attore il cui cognome è esattamente quello cercato dall'utente, ed il cui nome di battesimo inizia con quello che è stato digitato. Se esiste più di un attore che corrisponde a questo criterio di ricerca**, considerate soltanto l'attore che ha fatto più film**. In caso di parità, scegliete

l'attore con il numero di ID più basso.

Potete capire in quanti film un attore ha recitato usando una serie di join tra tabelle, ma questo potrebbe risultare troppo inefficiente.
Per fare qualche esempio, se scrivete la query correttamente e cercate nel database più grande (imdb) l'attore "Will Smith", vi verrà restituito quello con ID 444807. Per "David Cohen" avrete quello con ID 90749. Per "Elizabeth Taylor" avrete 809516. Se usate il database più piccolo imdb_small, la ricerca di "Chris Miller" restituirà l'ID 321300.

Non avete bisogno di usare alcuna JOIN qui, perché tutte le informazioni sono contenute nella tabella "actors". Se all'inizio non volete scrivere questa query, potete fissare l'ID di un attore nel codice, o semplicemente scrivere una query che restituisce il primo attore che risponde alla coppia nome/cognome, che sarebbe anche la risposta corretta quando non ci sono ambiguità da risolvere. Il comportamento della pagina non è definito se l'attore che viene cercato è lo stesso Kevin Bacon.

Estensione Opzionale (uso di sessioni per login/logout)

Potete provare a modificare il vostro sito in modo da prevedere un login/logout.

Questo significa che, ispirandovi anche al caso di studio visto a lezione (User Login Session), dovrete:

- Modificare il database inserendo una tabella user che includa almeno i campi user.id, user.username, user.password, dove andrete a mettere i dati per almeno due/tre utenti fittizi, cifrando le password.
- Modificare le pagine index.php e top.php per fare in modo che venga verificato se una sessione è già attiva ed un utente ha già fatto login. Nel caso positivo, index.php mostrerà il form per cercare l'attore nelle modalità descritte sopra. Nel caso negativo, sarà mostrato un form che chiede all'utente di effettuare il login fornendo username/password. Suggerimento: possibilmente aiutatevi con una pagina login.php che contiene questo ultimo form e che viene "richiamata" solo all'occorrenza, sostituendosi ad index.php.
- Modificare le pagine search-all.php e search-kevin.php perché non si possa accedere direttamente ad esse se l'utente non ha fatto prima login. Nel caso in cui un utente accedesse direttamente ad esse senza che una sessione sia attiva in modo corretto, allora il browser sarà redirezionato verso index.php mostrando il form che chiede in input i dati di login.
- Modificare la pagina top.php affinchè mostri, possibilmente in alto a destra, un link alla funzione di logout, che chiuda la sessione ed esegua una redirezione alla pagina login.php. Tale link deve essere mostrato solo a sessione attiva a seguito di un login andato a buon fine.

Vincoli sulla presentazione (su tutte le pagine)

Le tre pagine PHP dovranno adottare dei criteri di presentazione precisi, che sono specificati qui. Oltre a questi, qualsiasi altri aspetto legato alla presentazione potete sceglierlo voi, a patto che non entri in conflitto con quanto richiesto esplicitamente. L'intenzione è quella di consentirvi una certa flessibilità in modo da provare la vostra creatività in vista anche del progetto finale, in cui dovrete compiere molte scelte in autonomia; nello stesso tempo vale la pena tornare ad esercitarsi un po' con CSS, per non trascurarlo.

Includete le immagini nel vostro sito tramite dei riferimenti assoluti e non relativi.

- Tutte le pagine devono iniziare con il contenuto presente in top.html e terminare con quello presente in bottom.html, inclusi i riferimenti al "favicon", al logo MyMDb, ai validatori W3C ed i due form che consentono la ricerca dei film. È importante non modificare i nomi dei parametri usati nei form, come ad esempio **first_name**.
- La sezione principale del contenuto della pagina deve essere all'interno di un'area centrata più stretta rispetto alla pagina complessiva. Questa sezione principale dovrebbe avere un colore di background diverso di quello del body dietro di essa, per creare maggiore contrasto. L'esempio in figura usa un width del e un background bianco, mentre il body ha un background con colore.
- Ogni pagina deve contenere un'intestazione descrittiva di livello 1 (h1) che spiega il contenuto della pagina (Ad esempio: "Results for ***Kevin Spacey***" oppure "**The One Degree of Kevin Bacon**".)
- Le aree dei banner in alto ed in basso contengono il logo MyMDb e le immagini W3C che dovrebbero avere un colore comune e/o un'immagine di background per creare maggiore contrasto tra loro ed il resto della pagina. L'esempio mostrato nelle figure in alto usa come immagine di background il file e testo bianco.
- Il sito dovrebbe avere sia una colorazione che uno schema di font consistenti. Il vostro CSS dovrebbe essere strutturato in modo che sia facile cambiare il colore o i font senza che sia necessario modificare i colori ed i font in ogni singola pagina. Nelle figure di esempio sono stati usati per il testo, nero-su-bianco per l'area principale e centrale e per la sfumatura di grigio presente nel background.
- Tutto il contenuto dovrebbe poter essere ridimensionato in modo ragionevole, definendo il padding ed il margine in modo tale che il contenuto di una parte della pagina non debba sovrapporsi ad altro contenuto presente. Il contenuto dovrebbe essere anche allineato in modo ragionevole per consentire una lettura semplice. Nell'esempio nelle figure in alto, gli elementi sono stati centrati; i form hanno width 24em; molti elementi hanno 1em di padding o margine per le separazioni rispetto agli elementi circostanti.
- Ogni risultato della query dovrebbe essere mostrato in tabelle con 'caption' adeguate che descrivano il contenuto e con intestazioni che descrivono ogni colonna. I bordi dovrebbero essere 'collassati'. Nell'esempio in figura, ogni riga pari ha un background con colore, a partire dalla seconda.

Consigli per lo sviluppo, Deploy e test

Sebbene sulla pagina DIR troviate sia il database grande (imdb) che quello più piccolo (imdb_small), vi consigliamo di usare il secondo per sviluppare e di caricare quello più grande solo alla fine, quando siete sicuri che tutto funzioni bene, cambiando opportunamente i parametri di connessione PDO. Questo dovrebbe rendere la prova delle query più veloce.

Potete consegnare l'esercizio anche se il db usato è solo imdb_small, nel caso in cui il caricamento del db completo vi desse problemi.

Catturate le eccezioni generabili dal vostro oggetto PDO. Durante il debugging, stampate (print) le query che avete costruito per essere sicuri di eseguire le istruzioni SQL corrette.

Attenzione agli errori frequenti (apici, variabili, sintassi SQL...). Usate la funzione **quote**.

Infine, caricate tutti i file su una cartella **Esercizio3** all'interno del contenitore specificando un

file README.txt i componenti (max 2) del gruppo come <matricola>_<nome>_<cognome>. Non dimenticate di caricare anche i file top.html, bottom.html, index.php e nerdluv.css anche se non li avete modificati.
Potrete poi controllare il vostro sito semplicemente usando l'URL:

http://localhost:8080/Esercizio3/index.php

Valutate anche di usare la configurazione di XDebug per il vostro IDE. In ogni caso, ricordate che gli errori lato server vengono SEMPRE memorizzati nei file di logs (cercate dove il vostro sistema li ha installati...).

Implementazione e valutazione

L'output HTML per tutte le pagine dovrebbe superare il test di validazione W3C. Ovviamente il test non coinvolge il codice PHP, ma solo l'HTML da esso generato. Non usate tabelle HTML per definire il layout, ma solo per contenere dati ed informazioni (<table> serve per organizzare il contenuto, non per organizzare la presentazione e lo stile). Dato che stiamo usando form HTML, scegliete i controlli corretti ed i loro attributi di conseguenza. Scegliete GET e POST per spedire i dati al server come richiesto nel testo dell'esercizio. Il vostro codice PHP non deve generare **warning** o **errori** (possibilmente neanche **notice**). Minimizzate l'uso di variabili globali (a volte sono necessarie, ma abituatevi ad usare funzioni parametrizzate), usate correttamente l'indentazione e la spaziatura, evitate linee più lunghe di cento caratteri che rendono il codice difficile da leggere.

Alcune sezioni HTML sono in comune nelle varie pagine, ad esempio quelle presenti nei file top.html e bottom.html. Includete questi file in modo appropriato nel resto delle vostre pagine usando la funzione PHP include (o require). La ridondanza rende il codice difficile da leggere e complicato da riutilizzare. Usate funzioni, parametri, return, file esterni da includere, cicli, variabili, etc. per evitare ridondanza. Se avete del codice PHP (ad esempio una funzione che usate in diversi punti del sito) considerate la possibilità di inserirlo in un file common.php che poi potrà essere incluso nelle altre pagine.

Non filtrate i vostri dati con PHP: usate le giuste query SQL piuttosto. Se fate così, potrete trattare i vostri dati molto più semplicemente ed efficientemente. Ad esempio, è decisamente un esempio di pessimo stile di programmazione recuperare tutti i dati di una tabella dal db e poi eseguire complesse operazioni sull'array ottenuto per cercare le informazioni che corrispondono alla richiesta dell'utente. SQL è un linguaggio potente: usatelo!

Per fare tutto alla perfezione, riducete la quantità di codice PHP nel mezzo del vostro codice HTML. Il codice ridondante può essere inserito in una funzione, da dichiarare dentro il già citato file common.php. Ricordatevi di limitare il più possibile le istruzioni PHP print ed echo: usate piuttosto le espressioni blocco <?= ... ?>, così come abbiamo visto durante il corso.

Un altro aspetto importante è legato all'uso dei **commenti** PHP. Ad esempio, all'inizio di ogni file, di ogni funzione e di ogni blocco PHP, ricordatevi di mettere dei commenti descrittivi ed esplicativi.

Cercate di strutturare il vostro codice in un modo simile agli esempi visti durante il corso, facendo particolare riferimento ai diversi casi di studio che abbiamo analizzato. Usate spazi bianchi ed intestazioni in modo appropriato. Non mettete più di un elemento blocco per ogni linea della vostra pagina HTML.

In questo ultimo esercizio, è stato lasciato volutamente spazio alla scelta di usare o meno le tecnologie Ajax e JQuery con i meccanismi di chiamate asincrone. *La valutazione terrà conto*

positivamente nel caso in cui venisse scelto di usarle per implementare particolari funzionalità.