Overview
For this project, you will implement metadata and full-text search capabilities for data stored in a MySQL database.

What to do
For this project, do the following:

- In the MySQL account and database that was assigned to you on pearl.ils.unc.edu (e.g., webdb_1), create a table called "p4records" with the following fields and data types:

videoid	int primary key
title	varchar(255)
description	text
keywords	text
creationyear	int
sound	varchar(10)
color	varchar(10)
duration	time
durationsec	int
sponsorname	varchar(200)
contribname	varchar(50)
language	varchar(20)
genre	varchar(50)
keyframeurl	varchar(250)

  When you create the p4records table, you will need to construct one or more FULLTEXT indices to support the requirements below.

- A data set of 200 records from the OpenVideo web site (http://www.open-video.org) will be posted in the Resources section of the Sakai site for you to use for this assignment. The data set will be posted in two formats: 1) a text file with records one per line, fields separated by vertical bar characters '|', and 2) a text file that contains SQL insert statements. You can either load the data using the MySQL bulk loader (as described in class), or using the SQL insert statements. Regardless of which method you choose to load the data, make sure that you have successfully loaded all 200 records into your p4records table.

- In a file called search.html, create a fielded search interface that allows users to:

  - search over just the title field
  - search over just the keywords field
  - perform a full-text search over the combined title, description and keywords fields
  - restrict the search to a range of years (start year to end year)
  - restrict the search to a range of durations in seconds (min to max)
  - restrict the search to only records with sound
  - restrict the search to only records that are in color
  - combine **all** the features above (combine using logical "AND")

  The fielded search interface should be an HTML form with a button to submit the search. You can design the form as you wish, but I suggest that you use textboxes for most input fields, and that you use checkboxes for the sound and color restriction fields. The form action should call search.php.

For the search on the title field, you can use LIKE. For the search on the keyword field, you can also use LIKE. For the search on the combined title, description, and keywords fields, you should use MATCH..AGAINST with a FULLTEXT index.

- The search.php file should do several steps:
    - Initialize variables
    - Clean/Sanitize input using appropriate techniques
    - Construct an SQL query on the p4records table to return matching results.
    - Display a button or link to allow the user to go back to the search.html page to issue a new query.
    - Display the current query back to the user in a form that end-users would understand (i.e. do NOT display the SQL query).
    - Issue the query and display the results in an HTML table.

  You should display the results in an HTML table with the following headings: videoid, title, year, sound, color, duration, genre. The videoid should be a hyperlink to the OpenVideo web page for that record. For example, the OpenVideo URL for videoid 5304 would be: http://www.open-video.org/details.php?videoid=5304

  By default, you should display records in order by videoid. However, users should be able to click any of the column headings to re-sort the records by that field. Clicking a column heading to re-sort the records should preserve the current search parameters.

  You may display all the records that are returned from the search in one long table – you do NOT need to implement a paging mechanism for this assignment (although you may do so if you wish).

  Part of your grade for this assignment will be based on how easy it is to understand and use your advanced search interface.

How to do it

Implement the functionality described above using PHP. Be sure that your files work on ruby.ils.unc.edu with Firefox.

Be careful to make sure that you include a variable initialization block at the top of each file or code segment that initializes all variables used in that section of code. Also be sure that you have sanitized all $_GET and $_POST input including use of addslashes, striptags, and htmlentities AS APPROPRIATE. Your code should run correctly on systems with magicqutoes_gpc ON or OFF.

Develop your system using the two files as outlined above (search.html and search.php).

Be sure to include a `youronyen-p4-readme` file that contains ONLY the URL for your "live" project, followed by a blank line.

All your code and links should work with these files, named as indicated, in the same directory (except your dbconnect.php script if you use one). In other words, if your files were placed together in some other directory, they should still work.

What to turn in and how

Make sure all of your files are in one directory on a Unix machine (for example, ruby.ils.unc.edu). cd into that directory and then type the following command to create a gzipped tar file:

```
tar cvzf youronyen-p4.tgz .
```

Note that after the "tgz" there is a space and then a period.

Check your tgz file with the following command:

```
tar tvzf youronyen-p4.tgz
```

When you are satisfied that your tgz file is okay, you can submit it electronically through Sakai by going to the Assignments area and finding the "P4" assignment. After you think you have submitted the assignment, I strongly recommend checking to be sure the file was uploaded correctly by downloading it from Sakai and verifying it again with the "tar tvzf" command. Keep in mind that if I cannot access your files, I cannot grade them.

If for some reason you need to re-submit your homework file, you must add a version number to your filename. Sakai is configured so that it will only accept 3 total submissions. Use the following file naming convention if you need to re-submit:

Your first submission:	`youronyen-p4.tgz`
Your second submission:	`youronyen-p4-v2.tgz`
Your third submission:	`youronyen-p4-v3.tgz`

Sakai is also configured with a due date and an "accept until" date. Submissions received after the due date (even just 1 minute!) will receive a 10% penalty per day. The "accept until" date is 5 days after the due date. Submissions will not be accepted after the "accept until" date and will have a score of zero recorded.

If for some reason you are unable to submit an assignment to Sakai, as a last resort you may email it to the instructor along with a note about the problem you encountered. Then, as soon as you are able to, it is your responsibility to submit the exact same assignment to Sakai. The email serves as a record that you tried to submit the assignment on time, but to receive credit, your assignment must be uploaded to Sakai.