Project Report

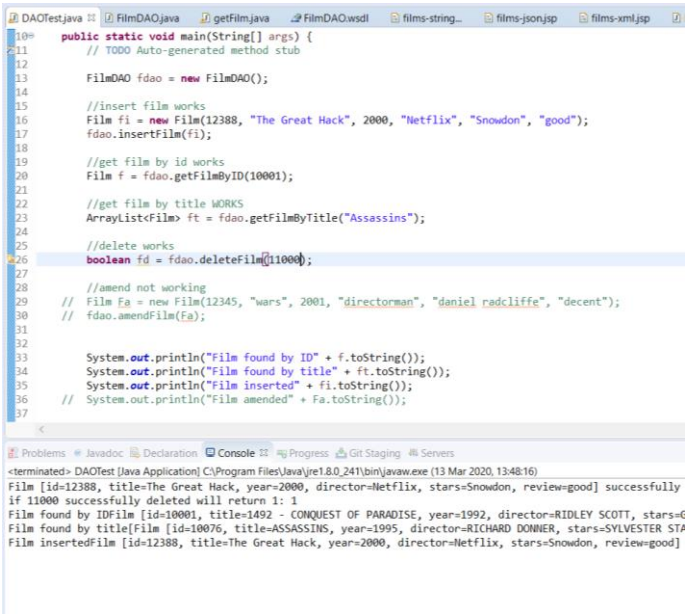
To meet the requirements for criteria 1 (access to the data using a simple http web service calls)

I used a DAO for films with

- getAllFilms() this returns all the films in the database, I limited this in the SQL statement to return the first 20 films.
- getFilmbyID(), at first I couldn't retrieve films from the database based from the title attribute so I used id, this would return film data based off the primary key, the film id.
- getFilmByTitle() this returned an array list of films where the title matched the search parameter
- insertFilm() I used prepared statements to insert a film from the DAOTest java file. If successful the console would print the film details + "successfully inserted", if there was an error, the console would print the details + "film not inserted".
- deleteFilm I would delete films from the database based on their ID number, if the film was deleted the console would print 1 and if the film wasn't successfully deleted would return 0.
- amendFilm this would update the film values based on the id number – if successful the console would return film successfully updated and film not updated if not.

I used both a stand-alone java program and servlets to the test the DAO.

On the stand alone java program titled DAOTest.java I tested the insertFilm, getFilmbyID, get filmByTitle and deleteFilm and amendFilm() operations. See the below screenshots for evidence, and database reflecting the results. I successfully deleted id 11000 from the database and inserted 12388. When trying to amend a film, the console would return an error. "Got an exception!
You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near '('title', 'year', 'director', 'stars', 'review') values (?, ?, ?, ?, ?) WHERE (' at line 1")"

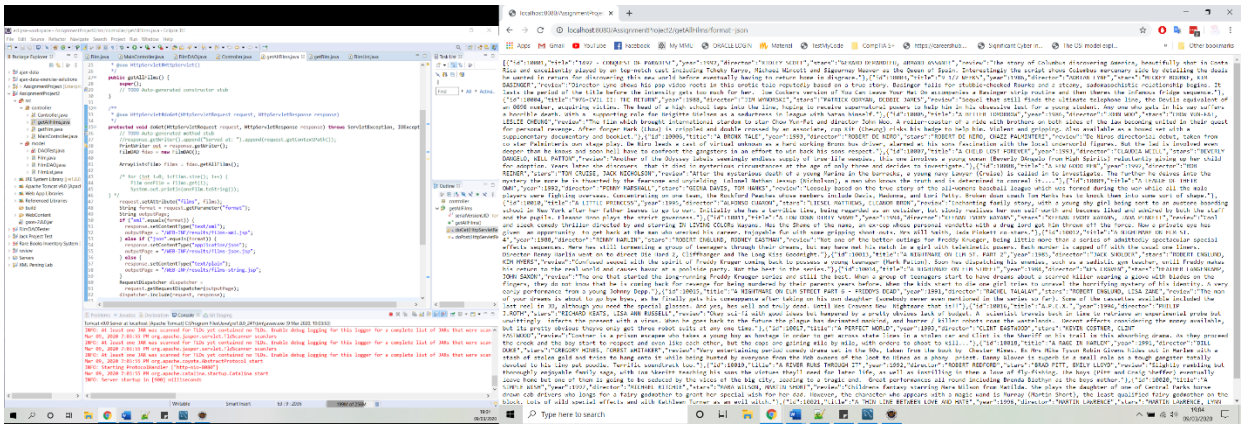


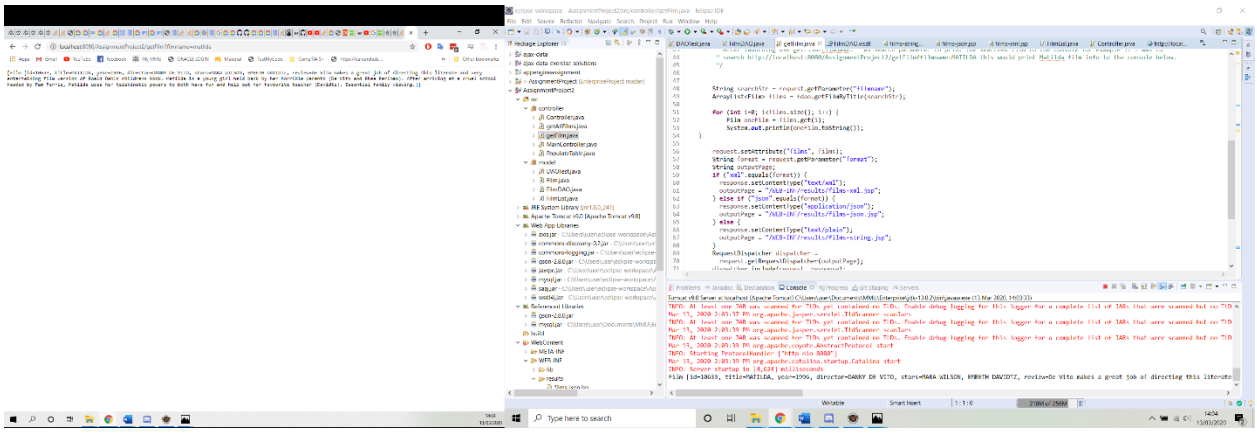
id	title	year	director
10993	TEENAGE MUTANT NINJA TURTL...	1992	STEWART GILLARD
10994	TERMINAL FORCE	1995	WILLIAM MESA
10995	TERMINAL RUSH	1996	DAMIAN LEE
10996	TERMINAL VELOCITY	1994	DERAN SARAFIAN
10997	TERMINATOR 2	1991	JAMES CAMERON
10998	TESS	1979	ROMAN POLANSKI
10999	THAT THING YOU DO!	1996	TOM HANKS
11001	THE ADDAMS FAMILY	1991	BARRY SONNENFELD
11002	THE ADVENTURES OF PINOCCHIO	1996	STEVE BARRON

Result Grid | Filter Rows: | Export:

id	title	year	director	stars	review
12388	The Great Hack	2000	Netflix	Snowdon	good

I tested getAllFilms and getFilmByTitle in servlets named getAllFilms and getFilm. I used the format http://localhost:8080/AssignmentProject2/getFilm?filmname=matilda and because I didn't specify a format, it would print the film as a string.

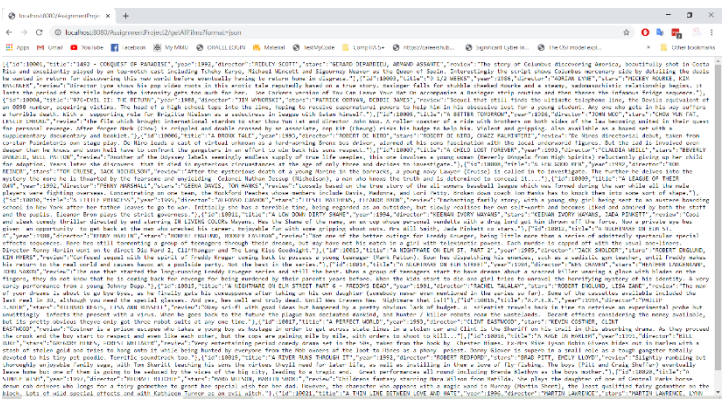




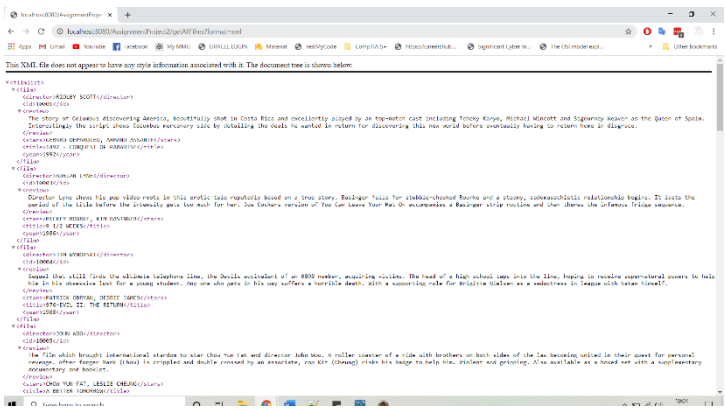
To meet the requirements for criteria 2 (Options to return the data in text, json (the default) or xml)

I used a format = request.getParameter and then if and else statements which would manipulate the film data into JSON, XML or text format.

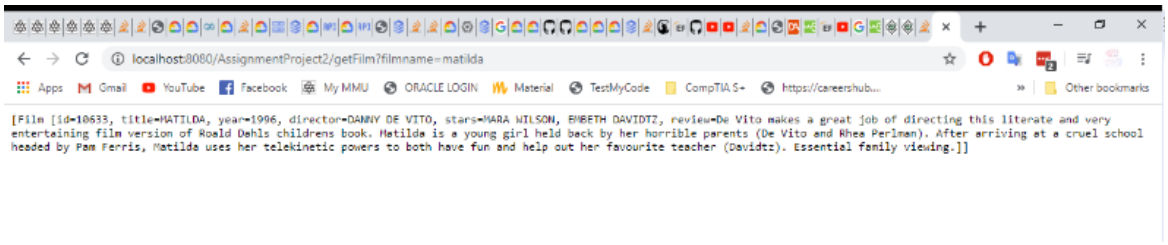
JSON



XML



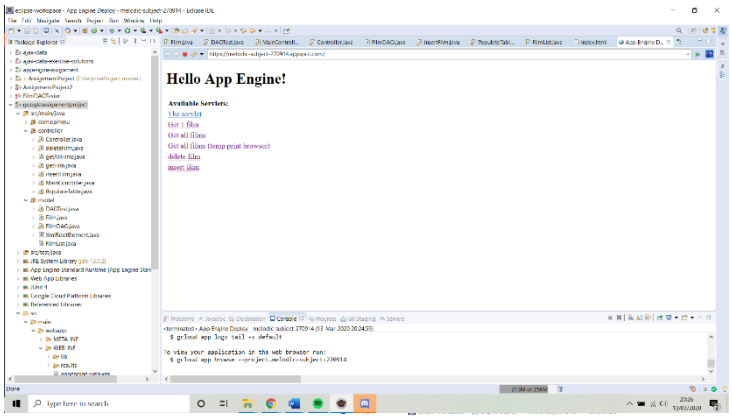
Text

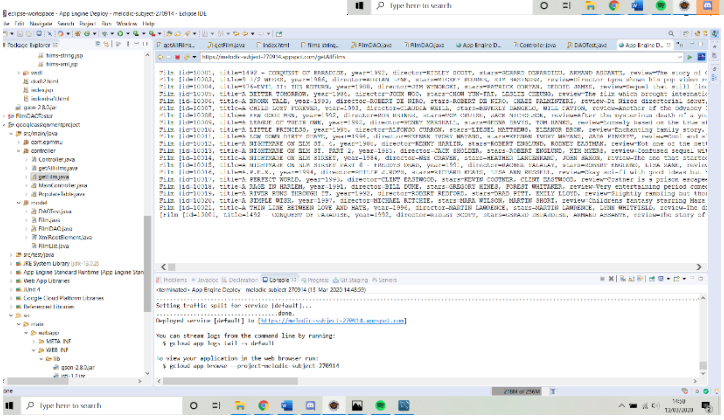


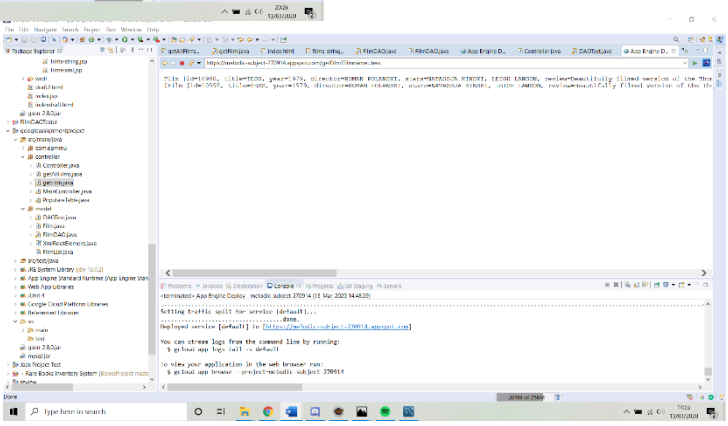
<u>To meet the requirements of Criteria 3 (Google App Engine or Microsoft Azure to implement the application on a remote cloud based server)</u>

I created a new Google App Engine Standard Java project, called googleassignmentproject, I created a controller package and a model package. I placed the FilmDAO and Film class in the model and the getAllFilms and getFilm servlet classes into the controller package.

I ran a version of the google app engine on googleassignmentproject joined to the mudfoot mmu database when I ran the getAllFilms servlet from the index, it would return all films to the browser. I also did this with the getFilm servlet using the parameter ?filmname=tess which would return the film to the browser also.

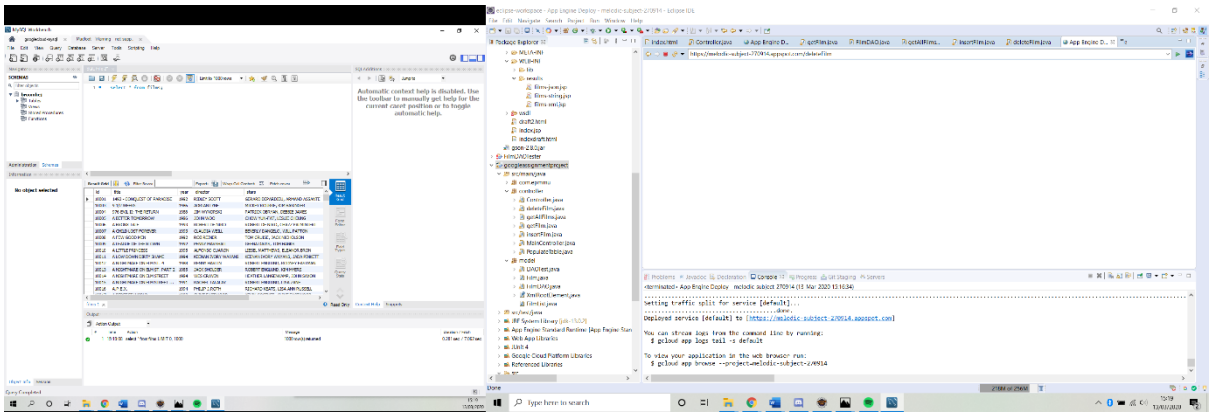




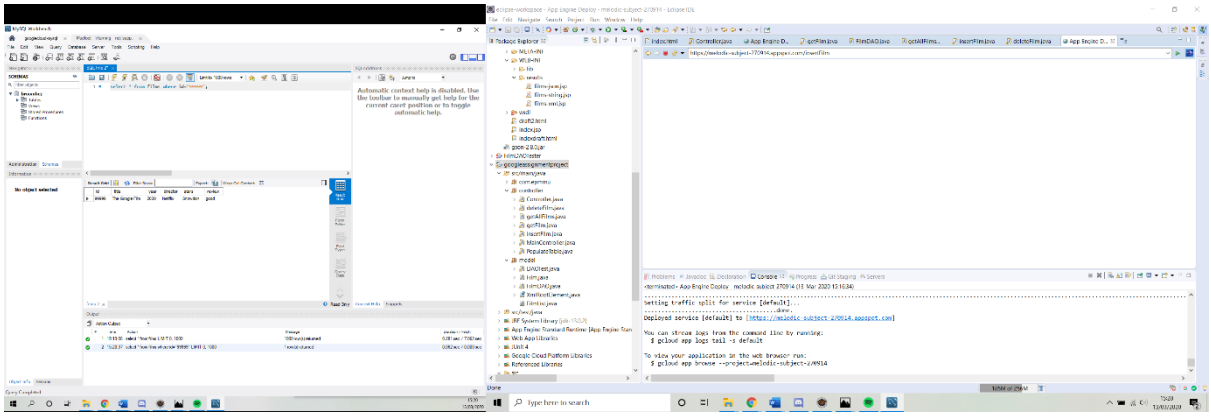


On the google app engine I created two more servlets, one called deleteFilms with deletefilms(10002) and one called insertFilm with the parameters 99999, “The Google Film” etc, when I ran these on the google app engine they didn’t return anything but they did manipulate the mmu database, as shown in the evidence below.

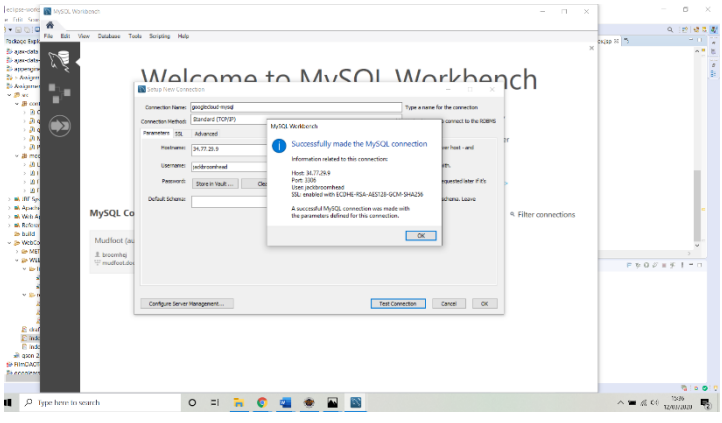
Deletefilm deleted the 10002 entry



insertFilm inserted “The Google Film” id: 99999

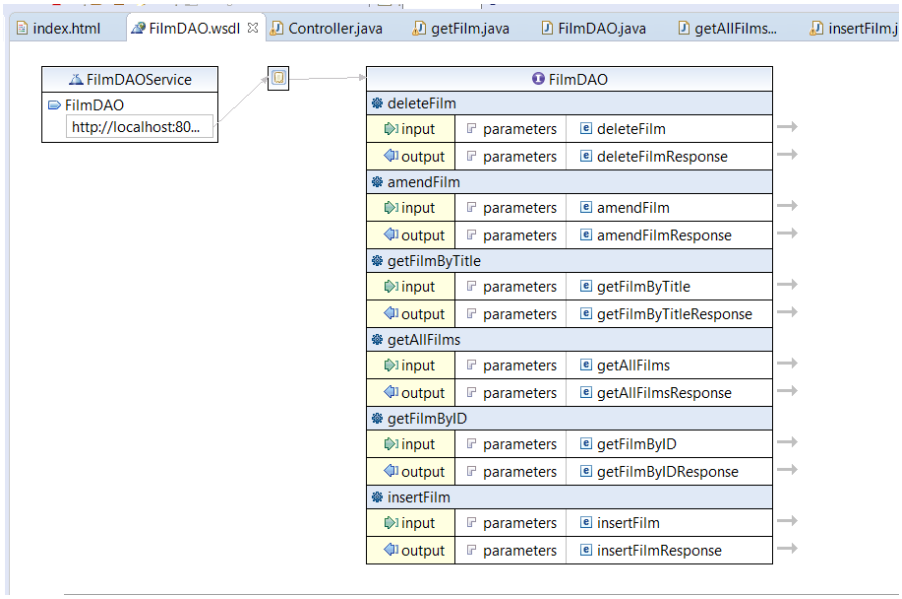


I next wanted the googleassignmentproject to use a google cloud mysql database to be a truly cloud based application. I created a cloud based database using the createfilms.sql file and I linked this to mySQLworkbench so I could view any changes made using crud operations. However, I had difficulty linking the google app engine project to the database, I tried using the public ip address (`34.77.29.9) with the port :3306` and the instance connection name (`melodic-subject-270914:europe-west1:mysqlfilmsdatabase`) but could not get the DAO to connect, given more time this is something I would like to get to work.



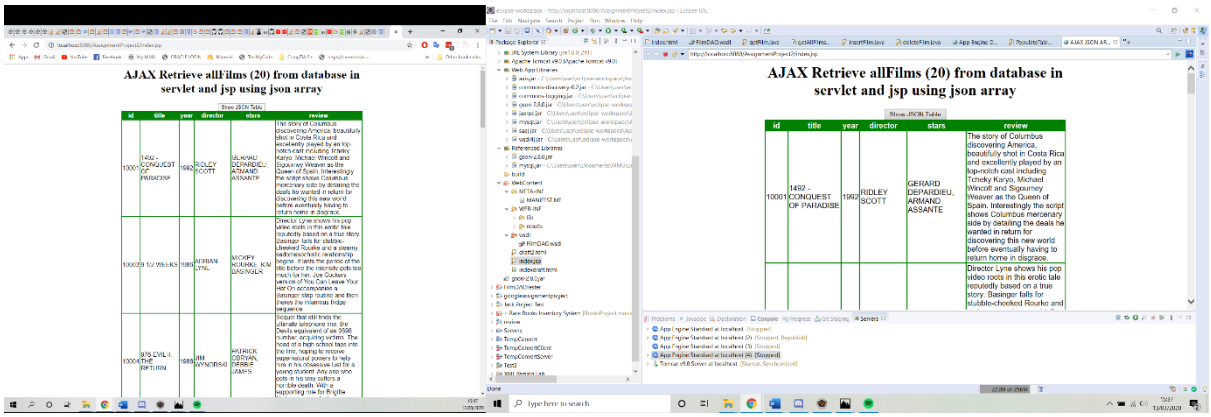
<u>To meet the requirements of Criteria 4 (A WSDL description of the interface to the web service (5 marks)</u>

I created a WSDL interface based of the DAO class in AssignmentProject2. This can be found in WebContent, then wsdl folder.



<u>To meet the requirements of Criteria 6 (An Ajax based web front end to retrieve the data and display in a suitable format using library based routines for an enhanced user interface (15 marks)</u>

For this I created a new class called PopulateTable, this would call get all films and print them as a json array. Then I created an index.jsp file, which when ran on the local host would create a json table with the data from the first 20 films.



Criteria 7 is written in a separate file, critical analysis.