Enterprise Programming Web Services Assignment – How to get started

Quite a few hints/help are given in the assignment brief itself. What follows here is enough to get you started and achieve a good passing mark in most sections. You don't need to submit one single project for the assignment, you may prefer to submit a project for different sections (this is definitely easier), so long as you demonstrate your capability in each of the objectives.

The Basic Application – Creating the basic web service

This is made much, much easier if you have done the labs. A lot of the code in the labs will give a good idea how to create the different sections, and the notes here will refer to certain labs/classes.

Step 1 – Creating the basic classes/tables/projects needed The Model in the Model-View-Controller (MVC)

- Create a Dynamic Web Project for the basic web service inside Eclipse.
- Create a package(s) inside the src folder to hold various linked classes – the model, the Data Accessor Object etc
- Create a basic data class, a java bean, in the project to represent a Film, with suitable attributes. Create constructors/getters/setters/toString (use the Source—Generate to do this for you).
    - Look at **City.java** and **Customer.java** in the ajax-data and ajax-data-exercises projects for ideas.
- Create a basic database table in mysql to hold your Film data, with columns for the attributes. You should use mysql or Oracle really, as these will be compliant with all platforms. Populate the table with a few sample Films. You may find it easier to use the createfilms script that was used in the XML parsing exercises to create a films table and populate it with a few hundred lines of data automatically (this will save you some time!)
- In mysql (or Oracle whatever), work out the SQL to retrieve/insert Films. Test them out to ensure you know what it should be. Create 2 selects, one to select all Films, one to select Films given an id (or name or other field), also create an insert statement to insert a new Film. Test them out interactively in mysql.

Step 2 – Create a Data Accessor Object (DAO) to access your data store
See **CityUtils.java** and **CustomerUtils.java** in the ajax data exercises.

- Create a class (FilmDAO), not a servlet, with several methods (you will find a starter piece of code for this in the XML parsing labs)
    - getAllFilms() – returns an ArrayList of Film Objects
        - Create an empty array list of Film, ready to return later
        - Using suitable JDBC methods, load the driver, open a connection to the database, and create a String for the select statement to select all Films from the table.
            - Print the query string as a debug message to the console to check later
        - Use executeQuery(selectStatement) to return a ResultSet of values
        - Iterate over the ResultSet, each time create a new member of Film object, and add it to the array list of Films.
            - Eg String filmName=rs.getString(1), where filmName is in column 1 of the result set, do for each attribute
            - Film f = new Film(….) or whatever your constructor is, then add to the array list
            - Print out each result to the terminal as a debug to see if you got the data (use your toString method for the Film object to help you)
    - getFilm(String Film) – returns an array list of Films with matching name (or substring) (make sure your test data has some values to match this)
        - Similar to getAllFilms()
        - Need a WHERE clause in the select statement.

        - Print out the generated select statement to debug test – common mistakes are made here
    - insertFilm(Film f) – input is a Film object to insert in the database, return value is an integer representing how many were inserted successfully (1 or 0)
        - Given a Film object, extract the attributes with the getters, and create an insert statement string.
        - Definitely print the debug insert string for the insert statement. Common error position
        - Use executeUpdate(insertStatement) to insert the data. Check the return value, 1 for inserted 1 value, 0 if a problem.
    - Later (or now) you should also create suitable amend/delete methods to complete the CRUD set of operations.

Step 3 - Test your DAO – The Controller in the MVC

You now need to test your DAO to see if it works. You will need a test harness, either a servlet of a standalone Java program, either of which will create and use the DAO from step 3. Similar code is shown in **ShowCities.java** and **CustomerList.java** in the exercises

- **Standalone Java program**
    - Create a java class with public static void main etc
    - Create an instance of your FilmDAO, and an instance of a Film object, f, with some test values in it
    - Invoke dao.insertFilm(s) and see if the result is 1 or 0. Look in your database Film table to see if inserted. Errors in the sql will be in the console screen along with your sql string debug messages etc.
    - Create an empty array list of Film, fList
    - Invoke fList = dao.getAllFilms()
        - Iterate over the array list and print each Film to the terminal (use your toString() method for Film objects). Ensure the data is correct
    - Invoke fList = dao.getFilm("Wars") or similar to retrieve a subset. Test the results as previous.
- **Create a servlet to access the DAO**
    - In the web project, create a new servlet getAllFilms
    - No input parameters to access
    - Create instance of DAO, and an empty array list for Film results returned
    - Populate the Film list with dao.getAllFilms()
        - Iterate over the list and print to console as debug as in standalone. This will also be useful debug info later
    - Create another servlet getFilm
    - Input parameter is a search name
        - String searchFilmname = request.getParameter("filmname");
    - Invoke fList = dao.getFilm(searchFilmname)   or similar
    - Iterate over/print results as previous to console
    - Create another servlet insertFilm
        - String Filmname = request.getParameter("Filmname") etc. for each attribute, get its associated parameter value
        - Create a new Film object, f
        - Invoke dao.insertFilm(f)
        - Debug print the result number of records inserted
    - Test each servlet from the browser by passing appropriate parameters
        - localhost:8080/myProject/getAllFilms
        - localhost:8080/myProject/getFilm?Filmname=Wars

    - Look in the console screen for debug prints of the data retrieved and the database table to check inserted.
- **NOTE:** At this point you are not sending the results back to the browser, just testing the code works.
- **At this point, you will have completed the section for the basic web service, and tested using http calls.** Simple prints of the results displayed will show that the servlets work.
- These sections are critical, as they hold the basic web service for later code

Step 4 – Formatting the output – The Viewer in the MVC
Now that the basic Model and Controller are in place, you need to decide how to format the output to send it back to the user in the browser. You will send back as a choice of xml, json or text depending on the format parameter supplied from the user browser.

The basic idea is contained in the demo code shown in **ShowCities.java** and **CustomerList.java** in the exercises. These are where the controller will invoke the DAO to get the basic results, then, depending on the format parameter, will package the data up in the request object and pass that on to a Viewer, a jsp program designed to format in the appropriate manner.

In your controller code for each servlet…. (compare to the demo code indicated)

- Access the parameter for format, allowing for a default value if none is sent
    - String format = request.getParameter("format");
    - if (format==null) format = “json”;
- Get your results from the dao
    - fList = dao.getFilm()
- Place them into the request object
    - request.setAttribute(“Films”, fList)
    - These will be accessed later in the viewer
- Invoke the appropriate viewer according to the value of the format

```
if ("xml".equals(format)) {
    response.setContentType("text/xml");
    outputPage = "/WEB-INF/results/film-xml.jsp";
  }
```

- The film-xml.jsp will access the array of Films and format in xml. See the demo example for how this is done. Note that the demo just takes 5 members and displays. You will need to amend the code to iterate over the Film list and generate xml tags etc. You should ideally use libraries to auto generate xml here (e.g. JAXB).
- film-json.jsp will generate json code. You should definitely use the json library utilities here that generate json from a java object

You can test your code from a blank browser as before, just to see the formatted output. This code you created is now the section on generating formatted output. You should capture the formatted output as proof that json/xml/text is being generated. Later sections will take these formatted data in the browser and extract values to be displayed within JavaScript. What you have now is enough for the section of formatting the output from the web service to a standard data format.

Step 5 – The browser side – The client manipulating the data from the web service

This section looks at the browser client, and involves creating JavaScript and Ajax calls to invoke your servlets and manipulate the results that come back to the browser.

Initially, you should create a button on a web form that will create a url in the format you used to test the web service, e.g.

- localhost:8080/myProject/getAllFilms
- localhost:8080/myProject/getFilm?Filmname=Wars

The values for the parameters come from data entry boxes.

Initially, just create a simple button that retrieves the data. You can format it later. Perhaps to make it simple, have 3 buttons for each service to retrieve the data in json, xml or text

At this stage, you should start making use of the Ajax lab examples to make your calls to the server from button presses and data entry boxes.

There are examples that take the data coming back in xml, json or text and format the results into html. Use these to aid you in creating your own layout.

Ideally, you should be using an Ajax library such jQuery to help automate your Ajax calls and make it simpler for you. Take a look at the jQuery labs, and the jquery.com site for reference

You should also look at and use the jquery-ui labs and web site http://jqueryui.com/ . This will greatly aid in layout.

The jquery/jqueryui is the correct technique to use for the assignment (or similar library), rather than painfully writing your own utilities. You will be graded upon the extent to which you use the correct tools for the task in hand.

This section will take some time, but will be greatly aided with the use of jquery (or similar) and doing the labs.

At this point, you will have done another section of the assignment, the client Ajax side.

Step 6 – Cloud based implementation
Here you will take your existing web service and migrate it to the cloud, ideally on Google AppEngine. Ensure you have done the labs on creating projects and using data stores., as this will make this section so much easier. Create a Web Application Project, and migrate over your classes from the original web service as needed.

Basically, you can use just about all the code you have already created for the web service. The only real part you need to change is how the DAO stores/retrieves its data. You need to remove references to using a local (ie mmu) mysql (or whichever database you used) and use a cloud datastore instead.

Initially you can use the mysql at mmu to test the cloud app, but a true cloud app would not behave this way, instead using a cloud based database. You should look at Google or Amazon to supply one of these in most cases. Most of your existing web service code should be re-used, just changing the DAO.

This can be a tricky section, but if your code earlier is well written, it will make the process much easier.

You can use any cloud based datastore in this section, so long as it is a true cloud based storage.

These notes should give you some confidence on getting a basic working web service in action. There is quite a lot of extras to fill in, and to refactor your code to make it reusable, expandable, stress tested etc. You should also be using appropriate design patterns, certainly MVC, DAO, Transfer Objects, as well as perhaps Singletons, Database Pooling. You should also refactor the code as needed. These are the factors that differentiate good assignments from excellent ones, and the marker will be evaluating your code based on the use of sound software engineering techniques.

The Evaluation/Critical Analysis

Please don't underestimate the critical analysis section of your assignment. It's worth 30 marks, almost 1/3 of the assignment total.

In this section you are commenting on the style of code you have developed, the SE techniques used. the options you decided upon and why. It is in this section that we will be marking the *quality* of your code, not just whether it does the job. We are looking more at *how* it does the job, and whether you are using the best techniques.

It is perfectly possible to create a working assignment just based on the lab examples, as they quite closely follow what is needed. However, the assignment goes beyond the labs, encouraging you to use better techniques. These are the things we are looking for, to indicate the professionality of your code.

What techniques are there? Basically the summation of your programming career at MMU or employer location, picking up on good techniques mentioned/covered in all your modules. There isn't a definitive list of every one of them, you will just be credited in this section with each valid technique you show a thorough understanding. Some examples are given here, but it's not an exhaustive list.

Design Patterns: Are you using any? If so, which ones? Why is each one a good idea? Where is it used in your code? Only talked about MVC? How about other design patterns you used: Singleton, Factory classes, Transfer Objects, Connection pooling and many others....

Refactoring: Have you refactored your code to perhaps place some useful utilities into your own package of classes? If you are repeating code, you should ideally place it into a common method, perhaps with parameters.

Is it obvious that you have just hacked the demo/lab code to get your own work running? You will need to do some extensive tidying up, refactoring, renaming etc.

Have you made conscious decisions to use one method of coding over another? eg using an api library call rather than writing your own code. Making sure the api is one that fits in with what you needed to do (think of, for example, using apis to generate json rather than writing your own, or using jquery for ajax calls rather than your own). What benefit does this have? Code libraries for the gui?

Is your code readable? Is it easy to maintain, good use of variable names, self commenting, javadoc compatible etc?

Are you making good use of objects? are they well designed? Do your java data classes have the correct naming convention, getters/setters/toString etc?

What makes good code? Hopefully you should have an idea after attending 2 and a half years of programming classes, and listening to those lecturers drone on about stuff. Here's your chance to show what you know and justify your coding style. If your want to find out some good practices or you have just forgotten, simple google "good programming practice" - lots of recommendations for documenting code, using good techniques etc

In writing your report here you need to show that you have used these technique you mention, and where. Point to code in your files, screenshots etc.

It's a good idea to keep your analysis separate from the proof that your code works in the first place (ie the screenshots to show your code working)

Random set of useful links (more easily found)

http://code.tutsplus.com/tutorials/top-15-best-practices-for-writing-super-readable-code--net-8118

https://msdn.microsoft.com/en-us/library/aa260844%28v=vs.60%29.aspx

https://github.com/thomasdavis/best-practices

https://en.wikipedia.org/wiki/Best_coding_practices

Martin Fowler has many good ideas on techniques to use for design and refactoring:
http://www.martinfowler.com/

http://people.cs.aau.dk/~torp/Teaching/E02/OOP/handouts/design_patterns.pdf shows some good "dos and don'ts". This is not the only source, you should research round.