Assignment Brief 2020/21
Unit Title: Enterprise Programming
Unit Code:6G6Z1103, 6G6Z1903
Level: 6
Assignment Title: Developing, deploying, publishing and finding your own web service
Unit Leader: Nick Whittaker
Contact Details: N.Whittaker@mmu.ac.uk
No of Elements in Assignment:
Submission Date: See date on Moodle
Submission Instructions: Detailed in assignment brief. Zip file submitted to Moodle
Feedback Return Information: Three weeks after submission date, Electronic feedback on Moodle
Assignment Task Overview: See separate assignment sheet with full details
Unit Learning Outcomes Assessed Learning Outcome 1: Critically analyse and implement Design Patterns Learning Outcome 2: Create and critically review distributed applications in a suitable language Learning Outcome 3: Apply knowledge of web services technology to create distributed systems Learning Outcome 4: Research and demonstrate knowledge and practical application of current and emerging enterprise level technologies

Assignment Details and Instructions.

In this assignment, you will create a simple web service that is accessible in a variety of ways using different technologies, and critically assess these approaches. The purpose is to demonstrate your ability to create and test differing formats of web service, to retrieve data in a variety of formats and to implement a cloud based web service.

You will develop a simple Films info web service from scratch, deploy it and develop a range of clients to access the data.

Interface

Your Films info API service will have the following (as a minimum) interface:

```
public interface FilmInfo {
  public void insertFilm (Film finfo);
  public void updateFilm (Film finfo);
  public void deleteFilm(int FilmID);
  public Collection listFilm ();
  public Collection retrieveFilm(String searchStr);
}
```

The Film class can have attributes such as: FilmID, FilmName, FilmYear, FilmCredits, FilmDuration, FilmReview and others. You may amend the details of the attributes as needed.

Storage

In order to implement the insertFilm, updateFilm, deleteFilm, listFilm and retrieveFilm operations, you should use a cloud database to store and retrieve film records, though you may decide to use a local storage option (such as mysql) to test your web services initially.

You are to create a simple web based app to manage the above interface, and provide a variety of methods to access the data from a client app, as detailed below

Technologies to use

You should demonstrate proficiency in using the following technologies as part of your web service:

- Access to the data using simple http web service calls (10 marks)
- Options to return the data in text, json (the default) or xml (10 marks)
- Google App Engine, Microsoft Azure or other cloud based service to implement the application on a remote cloud based server. (20 marks)
- A WSDL description of the interface to the web service (5 marks)
- Access to the data using REST type interaction (10 marks)
- 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)

Evaluation

You are to give a critical analysis of your work and the techniques you have used. Here, you should evaluate the pros and cons of each approach, and design/implementation decisions taken. This section forms a major part of the assessment. You need to demonstrate that your code is professional, and follows sound Software Engineering techniques. You need to show where they are used in your code and why, giving an evaluation and the benefits of using these techniques. This will show your true understanding of both your code and the techniques used. You will need to show how your code encourages a team based approach to software production, and indicate aspects of your code and design which demonstrate this. (30 marks).

Deliverables

You should submit a zip (not rar or other formats) file containing the following documents:

- A PDF (not odt, doc or other formats) file showing your code working, using screenshots, and suitable descriptions. **You should clearly indicate which sections you have completed**.
- A README file describing the steps to invoke your programs for deploying, publishing and finding your Web service
- Commented code listings in the Eclipse project code (no need to show separately)
- An Eclipse project(s) with the code ready to compile. You may have more than one project for client and server if you wish. This is a good idea as you don't need one all-encompassing project that does everything – it's not realistic.

The single file should have a filename of your name and student number (surname first), e.g. WhittakerNick02386456.zip

Please note that the submission inbox on Moodle will not accept submissions larger than 100MB. Your zip file is unlikely to be this large unless you have used very large image files. Please check in good time that your work will fit within the size limit. If you need to go over 100Mb, provide a link in your documentation to the external site where you have stored the data.

CAUTION:
There are a number of online implementations of this and similar concepts. While it is OK to review these for ideas, it is not OK to copy whole or parts of these code examples and pass them off as your own. We will use an AUTOMATED PLAGIARISM CHECKER to compare your submissions against other students' work and online examples.

NOTES AND HELP
Assignment Discussion – Detailed description of what is expected

The learning outcomes of the assignment are that you should demonstrate that you know how to deploy a web service in several ways, that you can return data from a service in a variety of data

formats, and make use of cloud services to deploy apps.

You are demonstrating that you can create software that delivers these objectives. You will be developing a series of classes, effectively your own API, to offer a web service. Most of the assignment isn't concerned with the client front end, you are not writing software for the general public. You are writing software that other programmers can use, and for machine-to-machine communication. The front end will be some simple user programs that demonstrate the API is working, rather than something for a normal user.

In order to show that you can write a web service, you are going to offer access to a simple plain old java object (pojo). In this case, it is a Film object. You will offer simple access to perform the basic CRUD operations for Film objects. The objective here is to show that you can offer web services in a variety of formats, and return data in a variety of data formats. We are deliberately keeping the actual object simple to reflect this.

The Film objects are to be based on a server, and there should be a collection of them. How the collection is stored will be up to you, but it should be cloud persistent in some way (i.e. not entirely memory based). Good solutions will implement a Data Accessor Object (DAO) to access the objects through a simple interface to the main server program. The DAO will hide the implementation of the persistence.

The final version should have cloud-based server code, using Google AppEngine, Azure or similar cloud service, depending on the implementation language of the server code. There are several methods of persistence, and you should choose an appropriate one. Remember that part of the marks is for your coding style and good use of software engineering techniques. You will have practice in using Google AppEngine and Design Patterns in lab sessions, and have reference to other methods.

The server code should offer a variety of methods to allow access to the data. Remember that you are implementing code for other programmers, so you are making it easy for others to invoke your web services.

You should implement a simple http type method of accessing the data using, for example, a servlet with appropriate parameters passed.

e.g. **http://serverlocation/Filmserver/retrieveFilm?name=Wars**

This would return a list of Films containing "Wars" in the name. The list of Film details should be in some easily processed format, xml, json or structured text fields (using suitable delimiter). It should be possible for the user to select the format, with a default if none supplied, e.g.

http://serverlocation/Filmserver/retrieveFilm?name=Wars&format=json or
http://serverlocation/Filmserver/retrieveFilm?name=Wars&format=xml

The server code will extract the data from persistent storage, and return the data in the prescribed format. Remember this is a web service returning just the plain data in a structured format. It is not sending it back formatted for viewing immediately. That is up to the client. You will look at that

later with an Ajax based client. Make use of suitable libraries here to convert to/from json/xml/pojo.

You need to demonstrate that you can upload the data in json/xml format to, via a POST call. Data from the client will be wrapped in json or xml and sent to the server (eg for a new Film entry). In the case of xml, this would probably be a SOAP call.

You should demonstrate calls to your web service in this manner, sending/retrieving different formats.

You are also to implement a RESTful interface to your server data, and demonstrate this working in a similar manner. Note that this should be a true RESTful implementation, making full use of HTTP methods POST, PUT, GET, DELETE.

All you need to show is the http calls to the web service and the data received for all combinations. You need to include a wsdl to describe the web services for potential users (remember you can generate one of these automatically once you have your web service)

The final bit of coding is to create a client side app using Ajax type calls, which will invoke the web service in the same way as above, but take the response data and display it on screen in a user-friendly way in the browser. Make use of suitable libraries to help you here (eg jQuery). Show both upload and retrieval of data from meaningful user screens in a browser, using suitable widgets from your JavaScript library.

It will be useful to you, and less time consuming, to use suitable libraries and demos from lab examples that you should have completed. Remember that many parts of this can be auto-generated (e.g. a SOAP call using XML, which has an appropriate testing interface ready generated - see the lab when we did this for the temperature conversion example).

The final section, the critical analysis, is where you should discuss the code quality and use of suitable structures, why you made the design decisions that you did. e.g. is it MVC based (you should show it if so), do you use DAOs, is your code modular and easily amended. What other design patterns or techniques have you used? Look at examples of good software engineering practice and demonstrate that you are using them. Commentary on which api (SOAP, REST, http) is useful to different users, and why, is needed. Have you used the correct API for the task? Is your code easily maintainable? (and why?).

There will be extra assistance with the assignment in the form of an online guide on Moodle, as well as online sessions devoted to assignment discussion. You should also (if needed) discuss your approach with your tutor, or personally after arranging an appointment at the availability times shown on the Moodle module home page.

Resources

Group Work Guidelines (If applicable, see Moodle)
Unit Specification – see Moodle

Assessment Marking Criteria

Grading Criteria

Component	Fail (0 to 39%)	3rd Class (40 to 49%)	2nd Class: 2 (50 – 59%)	2nd Class: 1 (60 – 69%)	1st Class (70-100%)
Criteria 1 Access to the data using simple http web service calls (10 marks)	No real attempt at creating a web service, or not working	Basic web service working, but not all CRUD operations implemented. Data not on persistent store	Persistent storage used, not all CRUD working	Persistent storage and CRUD implemented, errors not fully dealt with.	Full implementation working.
Criteria 2 Options to return the data in text, json (the default) or xml (10 marks)	No real attempt made at data format option	At least one of json or xml formatting included	All formatting included	Use of standard libraries to create formatting for json and xml	Clear and understandable usage of formatting using libraries and able to cope with range of data objects
Criteria3 Google App Engine or Microsoft Azure to implement the application on a remote cloud based server. (20 marks)	No real attempt made	Basic web service implemented on cloud server, not fully working	At least 3 CRUD operations implemented	All CRUD operations implemented	Fully tested and demo'd CRUD operations on cloud server
Criteria 4 A WSDL description of the interface to the web service (5 marks)	No real attempt made	Mention made of WSDL but not clear	Full WSDL implemented	Full WSDL implemented and documented	Full WSDL implemented and documented
Criteria 5 Access to the data using REST type interaction (10	No real attempt made	Basic REST working, but not all CRUD operations implemented. Data not on	Persistent storage used, not all CRUD working	Persistent storage and CRUD implemented, errors not fully dealt with.	Full implementation working. Data transactions sent in

marks)		persistent store		RESTful ops not correctly implemented.	correct RESTful way for all CRUD operations
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)	No real attempt made.	Basic attempt using in class examples, basically unchanged	Clear use of Ajax functions. Use of standard library such as jQuery used to a limited extent.	Ajax standard library used, code well organised and clear. Some attempt at use of UI library.	Excellent use of standard Ajax library and UI functionality.
Criteria 7 Critical analysis of your work and the techniques you have used. (30 marks)	No real attempt made, just listing of what was done	More detailed discussion of what was done. Mention of some SE techniques and Design Patterns but not tied into work done.	More evidence of SE techniques used and DPs, some attempt made of indication of where they are implemented in project work.	Clear evidence of SE techniques and DPs discussed. Some attempt made at evaluation of approaches.	Full discussion of SE techniques used and DPs with comparative evaluation.