SSC Coursework Breakdown

**Project Description**

**Front End (Web Service)** – Angular via Angular IDE

**Backend (Web Service)** – Maven + Spring framework via NetBeans

**Server** - Glassfish

**Terminology**

**Ajax -** Ajax is a set of web development techniques using many web technologies on the client side to create asynchronous web applications

**Angular** – This is an open-source web application framework. Angular is based on TypeScript while AngularJS is based on JavaScript. Angular is considered a frontend framework.

**API (Application Programming Interface)** - A software intermediary that allows two applications to talk to each other. The API is not the database or even the server, it is the code that governs the access point(s) for the server.

**ApplicationContext (Spring Term)** - In short, it is an object that loads the configuration (usually an XML file annotation based) and then Spring will start managing the beans and its benefits

**Beans (Spring terms)** - The objects that form the backbone of your application and that are managed by the Spring IoC container are called beans. A bean is an object that is instantiated, assembled, and otherwise managed by a Spring IoC container. These beans are created with the configuration metadata that you supply to the container

**Bootstrapping** – This usually refers to the starting of a self-sustaining process that is supposed to proceed without external input.

**CLI (Command Line Interface)** - A command line interface (CLI) is a text-based user interface (UI) used to view and manage computer files

**Client application** - An application running in a user's machine.

**“Consuming a Web Service”** - "Consume" means that the Web service successfully fulfils the web client's request. Context of Use: An end user performs a task on a web client that requires consumption of a Web service.

**Controller** – A controller in MVC architecture handles any incoming URL request.

**CORS (Cross Origin Resource Sharing)** - This is a mechanism that uses additional HTTP headers to tell browsers to give a web application running at one origin, access to selected resources from a different origin. A web application executes a cross-origin HTTP request when it requests a resource that has a different origin (domain, protocol, or port) from its own. (Good explanation [here](https://howtodoinjava.com/servlets/java-cors-filter-example/))

**DOM (Document Object Model)** – This is a programming interface for HTML and XML documents. It represents the page so that programs can change the document structure, style, and content. The DOM is an object-oriented representation of the web page, which can be modified with a scripting language such as JavaScript.

**Full Stack** - A full stack web developer is a person who can develop both client and server software.

**GET (HTTP Request Method)** - GET is used to request data from a specified resource.

**Grails** - Grails is an open source web application framework that uses the Apache Groovy programming language (which is in turn based on the Java platform).

**JavaScript** - JavaScript is the Programming Language for the Web.

**JAXB (Java Architecture for XML Binding)** - A software framework that allows Java developers to map Java classes to XML representations. JAXB allows storing and retrieving data in memory in any XML format.

**JSON (JavaScript Object Notation)** – This is a lightweight format for storing and transporting data The main difference between this and XML is that XML is a markup language (as it actually says in its name), whereas JSON is a way of representing objects. (Good explanation [here](https://stackoverflow.com/questions/2620270/what-is-the-difference-between-json-and-xml)).

**JSP (JavaServer Pages)** - JavaServer Pages is a collection of technologies that helps software developers create dynamically generated web pages based on HTML, XML, SOAP, or other document types

**Markup Language** - A markup language is a computer language that uses tags to define elements within a document.

**Maven** - Maven is a build automation tool used primarily for Java projects. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information. It does this by configuring your java project. For example, setting source folders, classpath etc. based on the maven poms. it allows you to easily get the required jars your application may need

**MEAN Stack** - The acronym describes four technologies that cover major software product development aspects, specifically MongoDB (NoSQL (non-relational) database), Express (back-end middleware), Angular (front-end framework) and Node.js (runtime environment).

**MongoDB** - MongoDB is a cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with schema.

**Node.js** - This is an open-source, cross-platform, JavaScript runtime environment that executes JavaScript code outside of a browser.

**OIDC (OpenID Connect)** – This is an authentication layer on top of OAuth 2.0, an authorization framework. The standard is controlled by the OpenID Foundation

**Pom.xml** - It is an XML file that contains information about the project and configuration details used by Maven to build the project.

**POST (HTTP Request Method)** - POST is used to send data to a server to create/update a resource.

**REST (REpresentational State Transfer)** – This is an architectural style for providing standards between computer systems on the web, making it easier for systems to communicate with each other. In the REST architectural style, the implementation of the client and the implementation of the server can be done independently without each knowing about the other.

**REST API** - A REST API defines a set of functions which developers can perform requests and receive responses via HTTP protocol such as GET and POST.

**REST Controller** – The RestController annotation is used to create RESTful web services using Spring MVC.

**Servlet** - A servlet is a Java programming language class that is used to extend the capabilities of servers that host applications accessed by means of a request-response programming model.

**Spring** - The Spring Framework is an application framework and inversion of control container for the Java platform. The framework's core features can be used by any Java application, but there are extensions for building web applications on top of the Java EE (Enterprise Edition) platform. Spring Boot is an open source Java-based framework used to create a micro Service. There are extensions to use Spring on top of J2EE and you technically can develop a front-end using Spring, but typically Spring is only used to write your back-end services.

**TypeScript** - TypeScript is an open-source programming language.

**Web Service** - A web service is an Application Program Interface (API) that runs on the server (Glassfish), which provide data to the client over http through a standardized messaging system.

**WSDL (Web Services Description Language)** - An XML-based file that basically tells the client application what the web service does. The WSDL file is used to describe in a nutshell what the web service does and gives the client all the information required to connect to the web service and use all the functionality provided by the web service.

**XML (eXtensible Markup Language)** - Designed to store and transport data.

A - Core Web Service (weighting - 35%)­

Implement Shares Brokering service

**Part A (Done)**

Current shares on trade should be held in an XML file based on an XSD schema containing:

* company name (**DONE**)
* company symbol on the stock exchange (**DONE**)
* number of available shares (**DONE**)
* a complex ‘share\_price’ element containing currency and value (**DONE**)
* date of the last-update of the share price. (**DONE**)

You can opt to use conventional structures to store the data (arrays, Lists, etc.) instead of XML objects. However, this will affect the standard of your work and also progression into Section-B. (***Our structure stores the data in an XML file using an ArrayList of type (XML DATA) – potentially known as the POJO structure.)***

**Part B (Done)**

The web service should allow users to:

* list shares details (**DONE** (in form of table))
* update the number of shares on offer when a purchase is made. (**DONE**)

You might opt to utilise JAXB to generate helper Java classes, which allow reading and populating the XML documents.

You may choose to implement the core Web Service using as a RESTful service, but for full marks you need to evidence that you can build & communicate a Shares data structure similar to what can be provided by an XML Schema as in 1.a) above.

**Part C**

Implement a search functionality allowing customers to list shares’ offerings using various criteria such as company details, highest price, etc. More sophisticated search functionality will merit higher marks.

* The choice of client is your decision. Java GUI Apps or JSP are more appreciated but should not be attempted at the expense of providing core functionality.

*Old Links*

*Reading and writing XML (*[*Link*](https://stackoverflow.com/questions/7373567/how-to-read-and-write-xml-files)*)*

*Marshalling data (*[*Link*](https://stackoverflow.com/questions/21751624/how-to-write-data-into-xml-file-using-jaxb)*)*

*More Marshalling (*[*Link*](https://www.intertech.com/Blog/jaxb-tutorial-how-to-marshal-and-unmarshal-xml/)*)*

*Even more Marshalling (*[*Link*](https://stackoverflow.com/questions/13788617/jaxb-marshalling-java-to-output-xml-file)*)*

*JAXB and Root Elements (*[*Link*](http://blog.bdoughan.com/2012/07/jaxb-and-root-elements.html)*)*

*Writing data to file (*[*Link*](https://stackoverflow.com/questions/29473055/how-to-write-to-the-next-blank-line-in-a-document)*)*

*JAXB Marshaller Overwriting data (*[*Link*](https://stackoverflow.com/questions/29135755/jaxb-marshaler-overwriting-file-contents)*)*

*Marshalling multiple pieces of data (*[*Link*](https://howtodoinjava.com/jaxb/jaxb-exmaple-marshalling-and-unmarshalling-list-or-set-of-objects/)*)*

Daniel’s Tutorial for Angular ([Link](https://www.techiediaries.com/angular-tutorial-example-rest-api-httpclient-get-ngfor/))

Netbeans Spring Guide ([Link](https://netbeans.org/kb/docs/web/quickstart-webapps-spring.html))

Angular 8 + Spring Boot Application guide ([Link](https://www.youtube.com/watch?v=PvdFCjWD4Bw))

Angular + Spring tutorial ([Link](https://learning.oreilly.com/videos/learn-full-stack/9781789530742/9781789530742-video1_1))

Angular 8 + Spring Boot Example Tutorial ([Link](https://www.javaguides.net/2019/06/angular-8-spring-boot-2-example-tutorial.html))

Spring XML GET guide ([Link](http://websystique.com/spring-boot/spring-boot-rest-api-example/))

Angular Parsing XML data guide ([Link](https://www.c-sharpcorner.com/article/reading-xml-file-in-angular-8/))

Without Maven

*JAXB Tutorial (*[*Link*](https://www.vogella.com/tutorials/JAXB/article.html) *1)*