**Creating the Project**  
There are several options in order to create this project and I opted to create it using the Spring Initializer at <https://start.spring.io>, specifying a Maven Project using Java 11, Spring Boot version 2.3.2, and packaged as a WAR file. Although not initially necessary, I wanted to enhance this application’s functionality and as such, I further requested the following dependencies be included in my project: Spring Web, Jersey, Thymeleaf, Spring Data JPA, and the H2 database. After generating, uncompressed the file and imported the existing Maven project into Eclipse.

**Project Structure**

Insert image

**RESTful Endpoint**

Create a controller annotated with @RestController. This is for Spring MVC to handle requests. It combines @Controller and @ResponseBody and will return data instead of a view. Based on the syntax provided in the example, the application will use Request Parameters rather than Path Variables. As per the requirement, the RESTful web service endpoint will return plain text (i.e. TEXT\_PLAIN).

**City Data**

I’ll model the collection of cities using a graph data structure where each vertex represents a city, and each edge represents a connection between two cities. It should be undirected (e.g. the road between Boston and New York may be travelled from Boston to New York or from New York to Boston). Additionally, it will not be weighted as there is no instruction to determine the shortest path between cities. Had this been the case, it would be possible to design and code a BFS search algorithm for traversing the graph. This implementation will be registered as a Spring Component and be injected by the Spring framework into the CityService.

**Loading the Cities**

Based on best practice, I placed the city.txt resource into the java.main.resources folder, where we can retrieve it via a ResourceLoaderAware implementation. This resource will be loaded from the classpath.

**Assumptions**Security will not be considered.

TODO:  
1. Document all classes

2. Code test classes, Junit 5

3. create README.md

4. Need to set up a GIT repository on GitHub. Map the project name to the repository name. Create a README.md.

Use Constructor Injection (@AutoWired no longer needed on constructor), code to interfaces.

Diff between @Service, @Component, @Controller, @Bean