# ASEP Project Sprint 1

## Introduction

This project is about building an application which displays information about stocks and portfolios. You will be given a lecture covering the business side of this in the second week of this sprint. For the first week, all you need to know is that you will be consuming some data from an external REST API, changing that data into a different representation, and returning that data to users in JSON and HTML representations.

## Getting started

1. Check out the project *portfolio-manager-spark* from Gitlab, it will be in your team’s repository.
2. Familiarise yourself with the code. The starting point is PortfolioManagerApplication.java. There are comments throughout the code which explain what it is doing.
3. Start the application by running PortfolioManagerApplication.java. This will start the application on localhost:4567. The available endpoint is localhost:4567/hello.

## What is the sample application doing?

The application that has been given to you is an example of how to build the following process:

1. The user goes to the following URL in their browser: localhost:4567/hello
   1. If you’ve not seen something running on localhost before, it’s simply a way to run a web server on your computer without exposing it to the wider interval. The number after the colon (4567) is the port that the application is running on. It’s just a unique way of accessing a particular application running locally on your machine.
2. The call is then passed to the controller method which is registered to /hello - in our case this is the get() method in HelloController.java.
3. This then makes a request to the PersonApiService class
4. This class uses OkHttp (<http://square.github.io/okhttp/>) to make a call to the external api running at <https://portfolio-manager-api.herokuapp.com>.
5. It calls <https://portfolio-manager-api.herokuapp.com/people> which returns a list of simple representations of people – their name and their age.
6. HelloController then checks if the user has requested HTML or JSON. As you have opened localhost:4567 in a browser, by default the browser requests HTML. We then convert the data we received from the external API to HTML using the Velocity Templating Engine (<http://velocity.apache.org/engine/2.0/vtl-reference.html> )
7. This populated HTML is then returned to the user.

## How do I request JSON from my service?

As mentioned above, when you open localhost:4567 in your browser, you are requesting HTML by default. To request JSON, use a browser extension like Postman for Chrome (search in the Chrome extension store) or RESTClient in Firefox.

In this extension, you can send a custom request to localhost:4567. You simply need to add the following Http Header:

“Accepts: application/json”

## What do I need to do to complete this sprint?

1. Call the external API to get a list of equities (hint: go to <https://portfolio-manager-api.herokuapp.com> to see a list of the available endpoints)
2. Turn the JSON returned by that endpoint into a list of Equity objects, which will have the following fields:
   1. EPIC
   2. Company Name
   3. Asset Type
   4. Sector
   5. Currency
3. Expose two endpoints:
   1. /equities
      1. This returns a list of these equities in JSON & HTML, depending on the “Accepts” header passed with the request
      2. The HTML version should be a table with each of the fields in a column
   2. /equities/:epic , eg /equities/AAL
      1. This returns the details of one specific epic in JSON & HTML depending on the “Accepts” header passed with the request
      2. Details of how to take a named parameter in are found here: <http://sparkjava.com/documentation#routes>
      3. The HTML should be a nicely presented representation of the Equity.
4. Test your code!
   1. Tests on HTTP endpoints can be carried out easily using Wiremock (<http://wiremock.org/>)
      1. See the example in PersonExternalApiServiceTest
   2. Tests on any other code you write can be done using Junit & AssertJ, as shown in HelloMessageServiceTest

## How will I be assessed?

We will be looking at the following factors

* Does your code build & run from the command line?
  + We will run “clean package” to build using Maven, either from the IDE or the command linem and then we will run your application directly from the jar, using the following command: java -jar target/portfolio-manager-1.0-SNAPSHOT-jar-with-dependencies.jar
  + NOTE: “it works from Eclipse / IntelliJ” will not be acceptable
* Do you have an appropriate amount of tests?
* Do they all pass?
* Does your code meet all the requirements
  + Both endpoints implemented
  + Both endpoints can serve HTML or JSON, based on the “Accepts” header
  + The data is pulled live from the remote server (we can change the data being returned by that service, so we’ll know if you’ve just stored it somewhere!)

Note that the marks awarded for the practical part of this course are 30% of the class, and each lab exercise from now on will be worth an increasing percentage of the marks.