Objectives

- Understand how Asynchronous Requests and Promises work in Javascript
  - Make a sequence of Ajax requests where one request depends on the information received from the previous request

JS Promises

- *Promise*: Object that serves as a placeholder for a value
  - Usually the result of an asynchronous operation
- The promise syntax helps to manage the execution order of callbacks
- An async function can immediately return a promise object

Promises (Basic Operation)

- The promise represents a pending computation
- Attach callbacks to the promise instead of passing them as arguments

```
var promise = asyncFunc();

promise.then(function (x) {
   // do something with the result
});

promise.catch(function (e) {
   console.log(e);
});
```

Callbacks

```
function mySequence() {
    $.get("myservlet",{},function (r) {
        show(r);
        $.get("myservlet?chain=" + r.expected, {}, function (r) {
            show(r);
            $.get("myservlet?chain=" + r.expected, {}, function (r) {
                show(r);
                $.get("myservlet?chain=" + r.expected, {}, function (r) {
                    show(r);
                    $.get("myservlet?chain=" + r.expected, {}, function (r) {
                        console.log("Number of steps advanced is " + r.steps);
                        show(r);
                    }, "json");
                }, "json");
            }, "json");
        }, "json");
    }, "json");
}
```

- This function shows how a sequence of requests can be implemented with callbacks and nested functions
- The goal of the exercise is to make the same sequence of requests but with Javascript promises instead of callbacks

Cloning the Web Application

- We have set up a public repository containing a basic Java web application[1]
- You can clone this project and import to Eclipse as you have done previously
- Make sure you are in the Java EE perspective (upper right corner)

[1] https://github.com/marks1024/promise-asynchronous-exercise-361

Task

- Download the starter project and run it on your local servlet container, you will see a page with information about the task
- Complete the implementation of the function `mySequenceWithPromises`
  - Use the `then` method to chain promises and note that you can return a promise from a handler
  - Note that the service responds to requests with a time delay
  - Make sure you keep the Javascript console open while working on this task
- Deploy your application to your local servlet container
- You should only need to edit the front-end code for this exercise