Objectives

- Understand declarative data bindings
  - Make an Ajax request and update the UI with knockout js

Knockout JS

- Lightweight JS implementation of MVVM Pattern
- Based on...
  - Observables and dependency tracking
  - Declarative bindings
  - Templating
- *Model*: Domain layer accessed via AJAX
- *ViewModel*: JS representation of the UI data and operations
- *View*: The actual UI, HTML with declarative bindings that link to the view model

Simple Example with KO


```
<script type="text/javascript">
var myViewModel;

$(document).ready(function () {
  myViewModel = { greetingName: ko.observable('Mark')};
  ko.applyBindings(myViewModel);
});
</script>
/////
<body>
<h2>KO Example</h2>
<p>
Hello, <b><span data-bind="text: greetingName"></span></b>.
How are you today?
</p>
</body>
```

- Framework sets up the pub/sub mechanism for us, we only need to call `myViewModel.greetingName("new value")` to update the view

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/data-binding-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
- Recreate the Ajax request and update the UI with knockout instead of JQuery
  - Inspect the first example to understand how declarative data bindings work
  - Add necessary properties to the view model and bind them with new HTML elements
  - 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