

Exercises

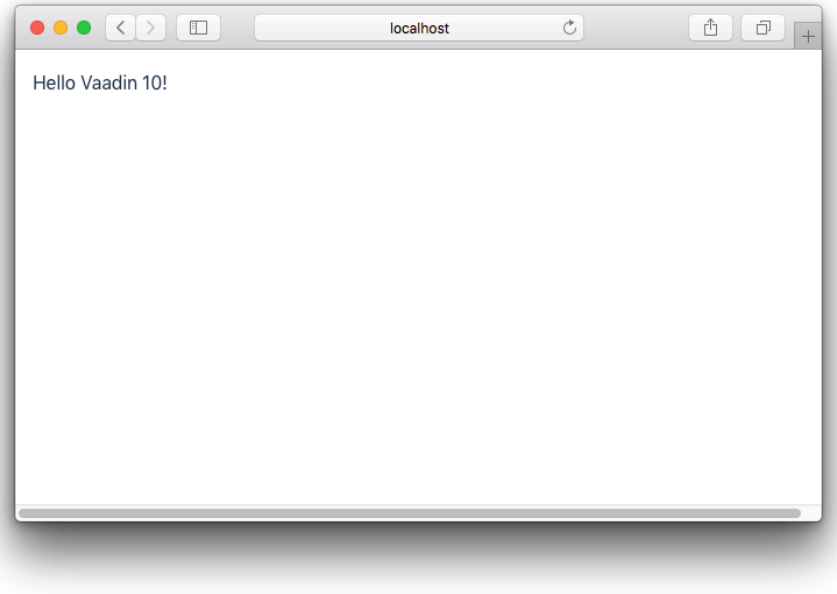
Exercise 1: Hello Vaadin 10!

In this exercise you are getting yourself familiar with Vaadin 10 with a traditional Hello World application.

1. In `MainView.java`, add the following line to the constructor:

   ```
   add(new Paragraph("Hello Vaadin 10!"));
   ```

2. Deploy to your server (or run mvn jetty:run) and then you should be able to see something like this:

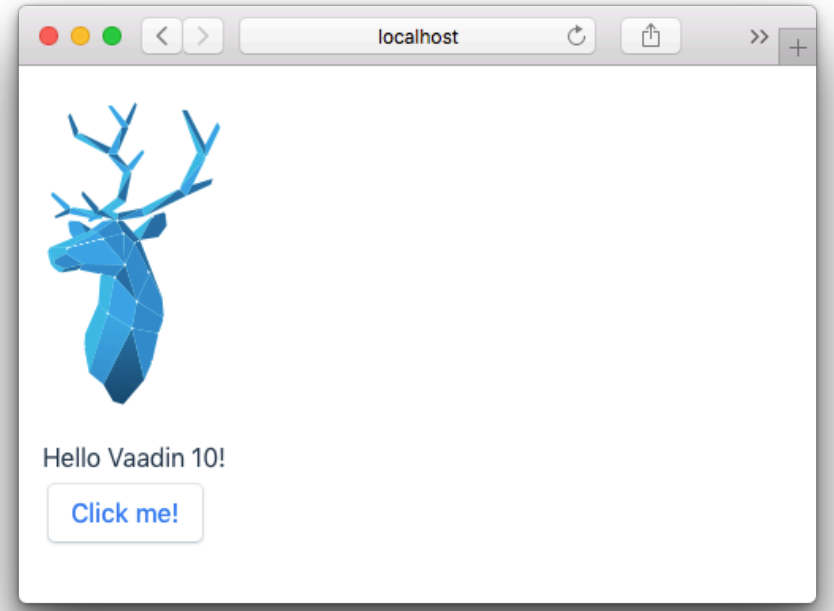


3. Lets add some interactivity by adding a Button
   1. Create a new button component (`com.vaadin.flow.component.button.Button`) and add it to the view. The button caption should be “Click me!”.
   2. Add a click listener for the button. In the listener, you should create a new Paragraph with the text "added from button!”, and add it to the view.

Exercise 2: Creating a Composite Component

As plain text is boring, we also want to show a beautiful reindeer above the text. So let's create a composite component which contains a reindeer image as well as a paragraph which shows the hello text.

1. Create a new class VaadinWelcome.
2. Let VaadinWelcome extend `Composite<Div>`, the Div generic parameter here means that the client side implementation will be a tag.
3. Let VaadinWelcome implement the `HasComponents` interface.
    1. Notice that you don't need to implement anything, since all the methods have been implemented as default methods. By implementing this interface, we can add a component inside this one by just saying `add(Component…)`.
4. Move the hero-reindeer.svg file from the root into your `src/main/webapp` folder. This makes it a public file, accessible from the browser.
5. Create and add an Image component to VaadinWelcome, the image should take the hero-reindeer.svg file as source. The alt text can be whatever.
6. Move the 'hello' text creation into this file from the MainView file.
7. In MainView.java, create and add a VaadinWelcome component as the first content.
8. Now you should be able to see something like this in your browser:



Exercise 3: First taste of Template

Template is a very powerful new feature provided by Vaadin 10. On the client side, you can have any html template, and on the server side, you have a good old java class. The Template class (with help from Flow) can do the magic to connect them. Let's get started and we will reuse the example template provided by the skeleton project.

1. Copy the example-template.html file from the root into `src/main/webapp/frontend`
2. In the example-template.html file, update the content inside the `<template>` tag to be something like:

```
<div>What's the best UI framework?
    <button on-click="handleClick">Search</button>
</div>
<div>{{value}}</div>
```

3. In example-template.html file, in the <script> section, add a handleClick function inside the ExampleTemplate class.

```
handleClick(){
    this.value = "searching...";
}
```

4. Create an interface called ExampleModel, ExampleModel should extend from TemplateModel, and it has only two methods, getValue() and setValue().

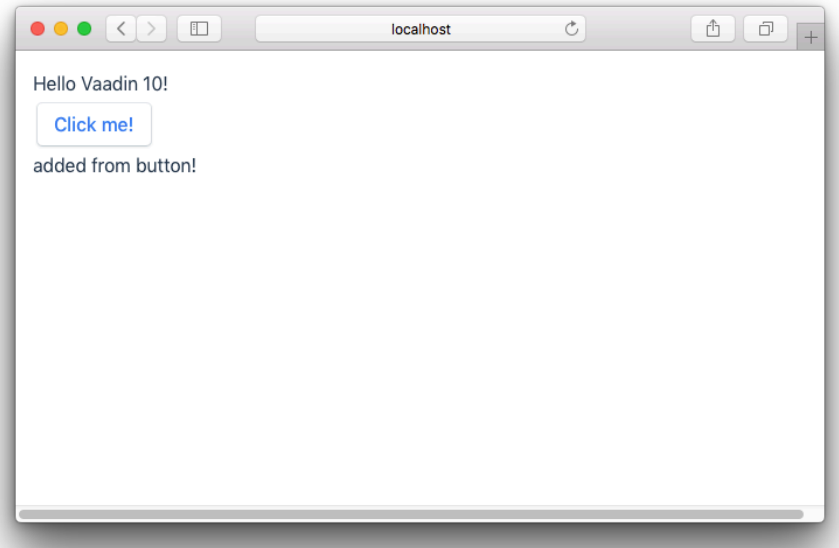
```
public interface ExampleModel extends TemplateModel{
   void setValue(String value);
   String getValue();
}
```

5. That's all for the client side. For the server, create an ExampleTemplate.java class. It should have the following class signature.

```
public class ExampleTemplate extends PolymerTemplate<ExampleModel>
```

6. To connect the Java class to the Template file, add the following annotations to the ExampleTemplate class:

```
@Tag("example-template")
@HtmlImport("example-template.html")
```

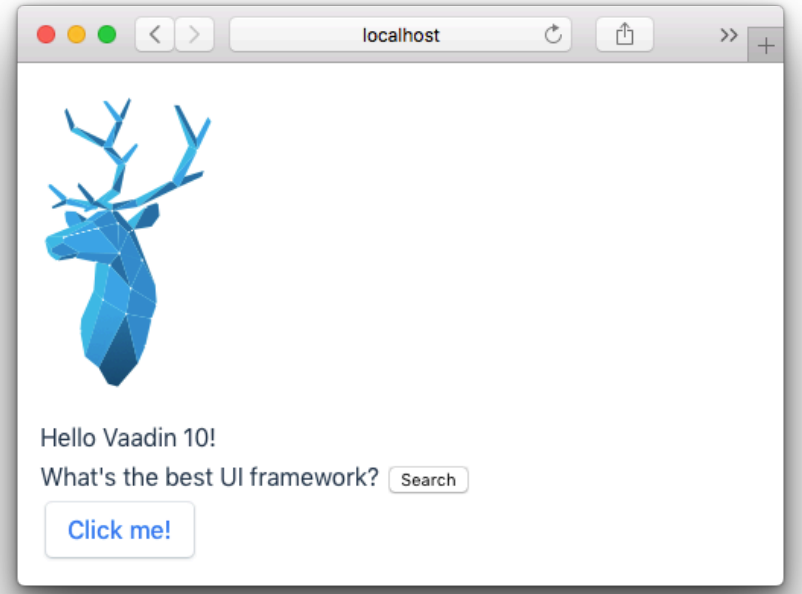


7. In MainView.java, create and add a ExampleTemplate component as the second content.

8. In ExampleTemplate java class, add a new method called `handleClick`. Put an `@EventHandler` annotation on the method, and simulate that it's doing some heavy computation, and then update the model. Something like this:

```
@EventHandler
private void handleClick(){
    try {
        Thread.sleep(2000);
        getModel().setValue("Vaadin!");
    } catch(InterruptedException e) {
        e.printStackTrace();
    }
}
```

9. Now everything is done, you should be able to see something like this:



8. Notice that the button inside the template isn't themed. This is because it's a normal HTML button, not a Vaadin Button. We can change that by changing the tag in the HTML template file from `<button></button>` to `<vaadin-button></vaadin-button>`: