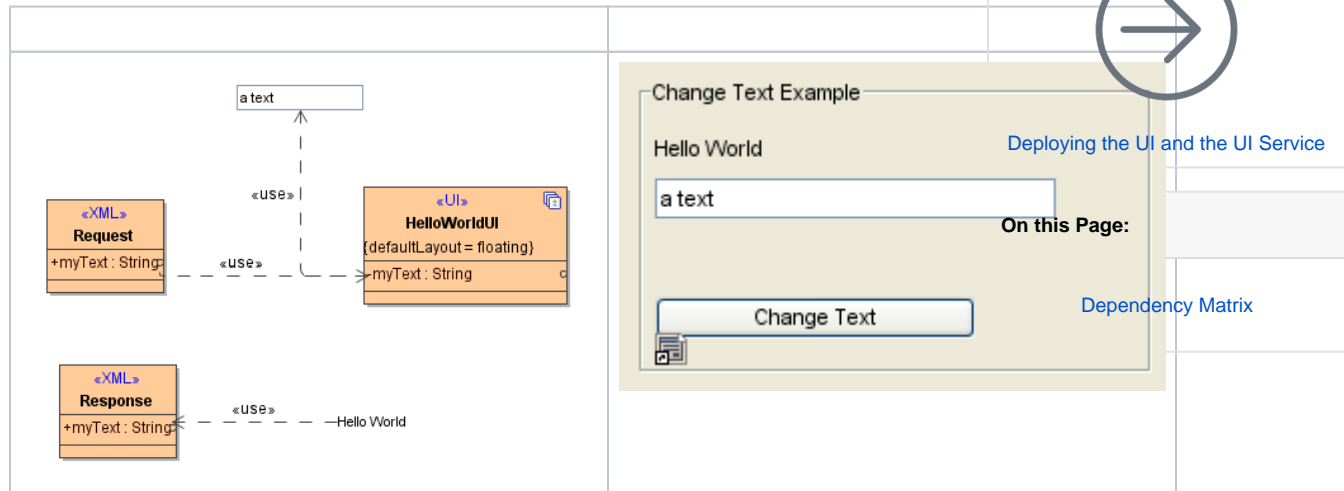
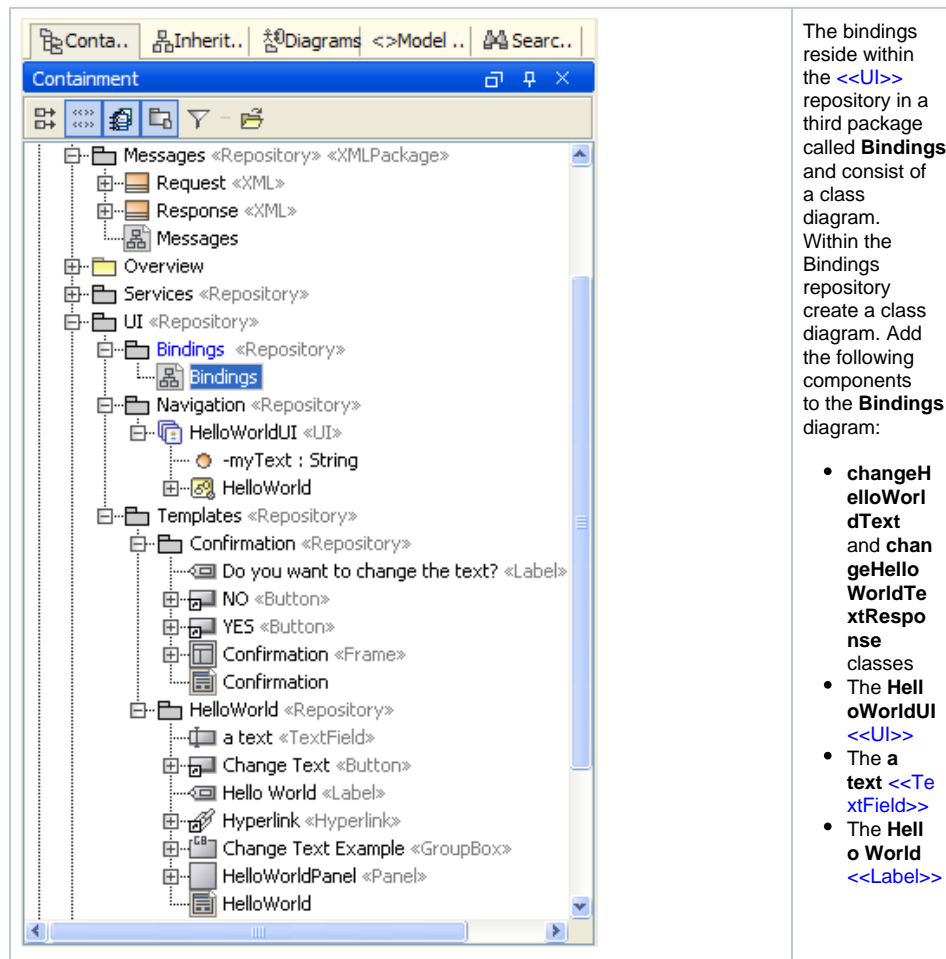


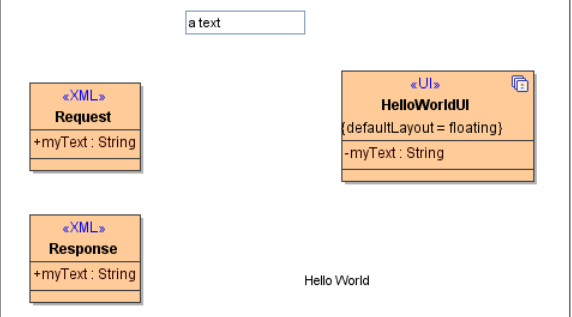
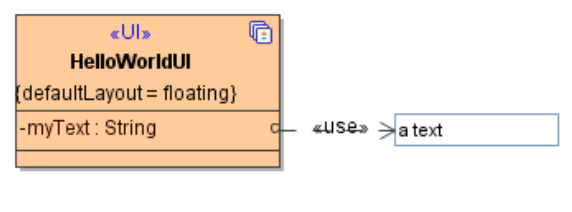
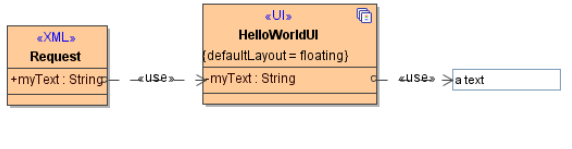
Defining the Data Binding

Now that all required components are modeled, the data needs to be bound and mapped to the user interfaces.



For this tutorial there are two UI elements, the HelloWorld label and the text input field, which need to be bound to the data model. The data model consists of the classes within the Messages repository, **change HelloWorldText** and **change HelloWorldTextResponse**. Due to the fact that there is the **myText** global variable defined within the **HelloWorldUI** it becomes part of the data model itself.

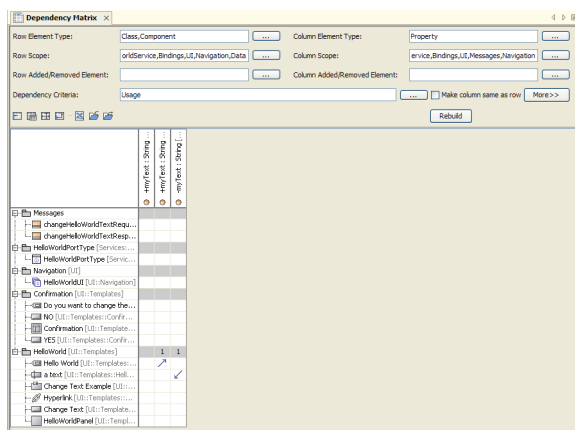


<p>package Class Diagram Bindings [Bindings]</p>  <p>The diagram shows three classes: Request (XML), Response (XML), and HelloWorldUI (UI). Request and Response both have a public attribute <code>+myText : String</code>. HelloWorldUI has a public attribute <code>-myText : String</code> and a stereotype <code>{defaultLayout = floating}</code>. A text element <code>a text</code> is shown at the top. The text "Hello World" is written at the bottom.</p>	<p>The class diagram should look like this. All components holding data are now available for the binding using <code><<use>></code> dependencies.</p>
 <p>The diagram shows the HelloWorldUI class (UI) with a public attribute <code>-myText : String</code> and a stereotype <code>{defaultLayout = floating}</code>. It has a <code><<use>></code> dependency on the text element <code>a text</code>.</p>	<p>Following the flow of the data, the first place where data is changed is the <code><<TextField>></code> element. To have the custom text input of the user available at any time and from other user interfaces, the data is bound to the global variable myText within the HelloWorldUI class.</p>
 <p>The diagram shows three classes: Request (XML), HelloWorldUI (UI), and a text. Request has a public attribute <code>+myText : String</code>. HelloWorldUI has a public attribute <code>-myText : String</code> and a stereotype <code>{defaultLayout = floating}</code>. Request has a <code><<use>></code> dependency on HelloWorldUI, and HelloWorldUI has a <code><<use>></code> dependency on a text.</p>	<p>Further, the custom text input will be used as the input parameter of the SOAP operation and therefore needs to be bound to the Request object itself. The custom text input is now bound to the global variable myText, from which the Request SOAP operation parameter will get the data from.</p>



Dependency Matrix

In real world projects, a given UI component might hold confusingly many bindings. An elegant way to get an overview of the dependencies between elements is to use the Dependency Matrix Diagram.



The Dependency Matrix shows all user interface components and their connection to the class properties of the data model. The direction of the arrows show the usage dependencies directions. The **Hello World** label (having an up arrow) uses the data from the **changeHelloWorldTextResponse** class attribute **myText**. On the other side the global variable, residing in the State Machine, is getting its data from the **a text** text field component.

The Dependency Matrix Diagram parameters can be changed to achieve the result that is needed. The Dependency Matrix Diagram is a view only diagram, meaning Bindings can not be edited.

To create the Diagram choose **Diagrams > Custom Diagrams > Dependency Matrix...**