The Scout "Hello World!"

Scout Team

Version 4.0.200-SNAPSHOT

Table of Contents

After installing the Eclipse Scout package you are now ready to implement your first Scout application.

Create a Scout Project

Start your Eclipse IDE and select an empty directory for your workspace. This workspace directory will then hold all the project code for the “Hello World” application. Once the Eclipse IDE is running it will show the Scout perspective with the Scout Explorer view and an empty Scout Object Properties view. To create a new Scout project select the **New Scout Project...** context menu as shown in Figure New Scout Project Menu.

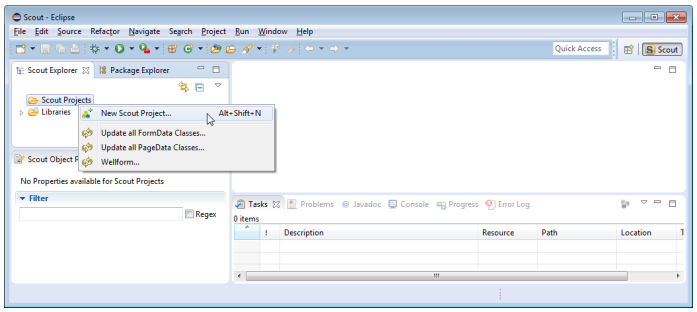


Figure 1. Create a new Scout project using the Scout SDK perspective.

In the *New Scout Project* wizard enter a name for your Scout project. As we are creating a “Hello World” application, use org.eclipsescout.helloworld for the *Project Name* field according to Figure New Scout Project Wizard. Then, click the **[Finish]** button to let the Scout SDK create the initial project code for you.

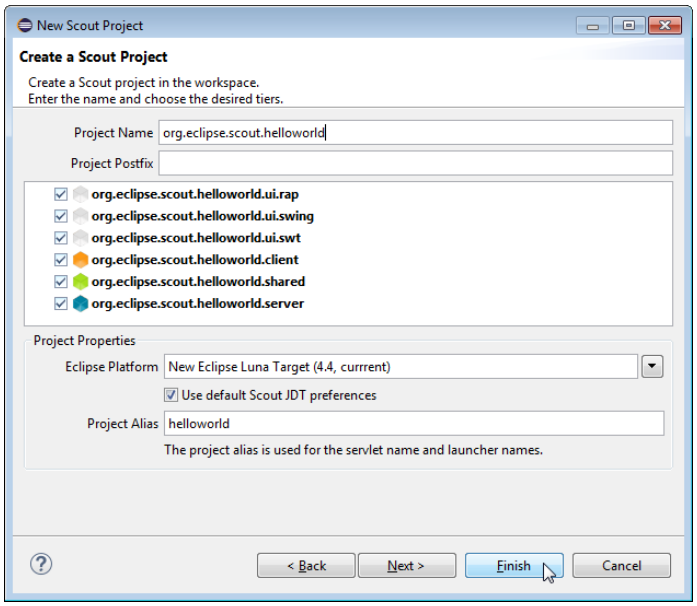


Figure 2. The new Scout project wizard.

Once the initial project code is built, the Scout SDK displays the application model in the *Scout Explorer* as shown in Figure Representation of the Hello World Application. This model is visually presented as

a tree structure covering both the client and the server part of the application. The Scout Explorer view on the left hand side displays the top level elements of the complete Scout application. Under the orange node the Scout client components are listed. Components that are needed in both the Scout client and the Scout server are collected under the green node. And the Scout server components are listed below the blue node in the Scout Explorer view.

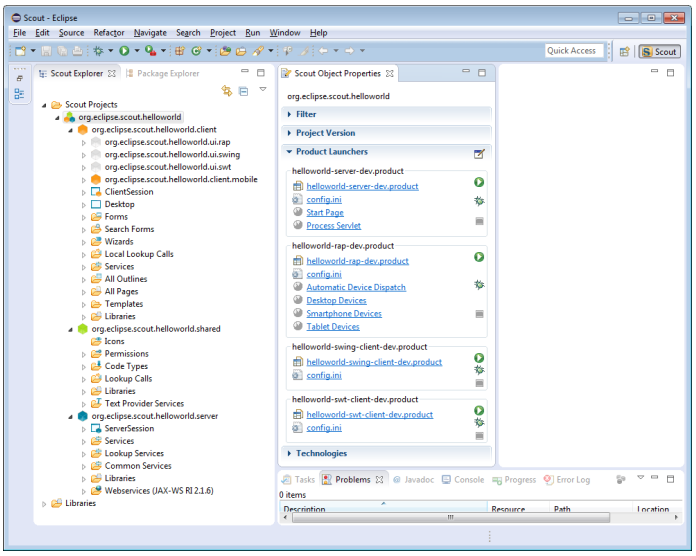


Figure 3. The Scout SDK showing the tree representation of our "Hello World" application in the Scout Explorer. The Scout Object Properties contain the product launchers for the server and the available clients.

Run the Application

After the initial project creation step we can start the Scout application for the first time. For this, we switch to the Scout Explorer view and select the root node org.eclipse.scout.helloworld. This then loads the corresponding controls and the *Product Launchers section* into the *Scout Object Properties* view as shown in Figure Representation of the Hello World Application.

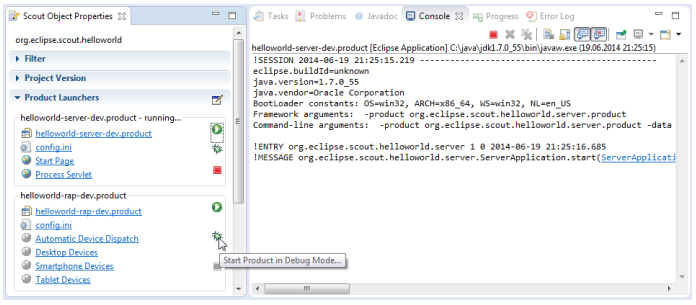


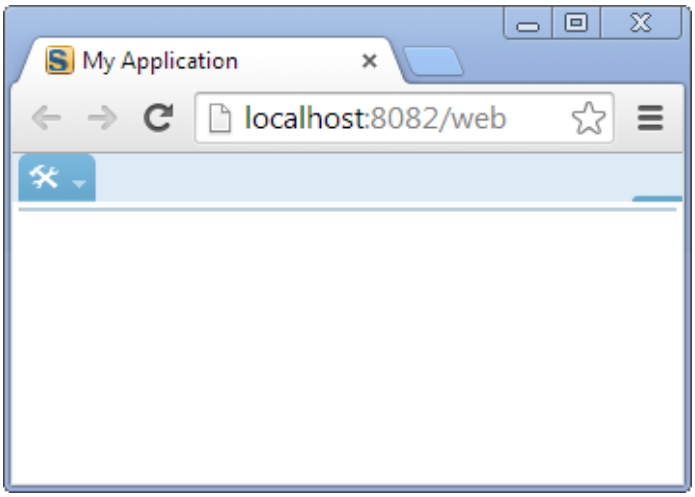
Figure 4. Starting the Hello World application in the Scout SDK using the provided product launcher. Make sure to start the server before starting any client product.

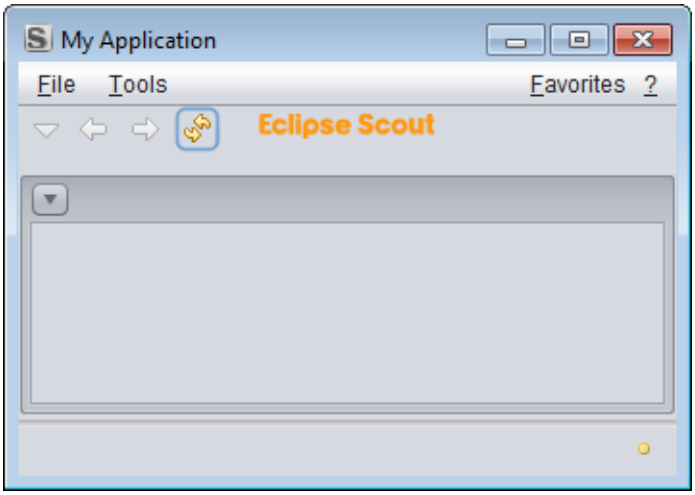
In the product launcher section of the Scout Object Properties view four launcher boxes are available. One launcher box for the Scout server product, and three launchers for the different client products. Each launcher box provides a link to the corresponding configuration and product definition files, as well as the launcher icons to start and stop the corresponding product. The green *Circle* icon starts the

product in normal mode. The *Bug* icon just below, starts a product in debug mode. To terminate a running product, the red *Square* icon is provided.

Before any of the client products is started, we need to start the server product using the green circle or the bug launcher icon. During startup of the Scout server you should see console output similar to the one shown on the right hand side of Figure Starting the Hello World application.

Once the server is running, you may start the RAP client as shown in Figure Starting the Hello World application. To start the Swing client, or the SWT client use the corresponding green *Circle* icon or *Bug* icon. And with a running RAP client, the Scout client can be opened in a web browser by clicking on the provided *Automatic Device Dispatch* link.





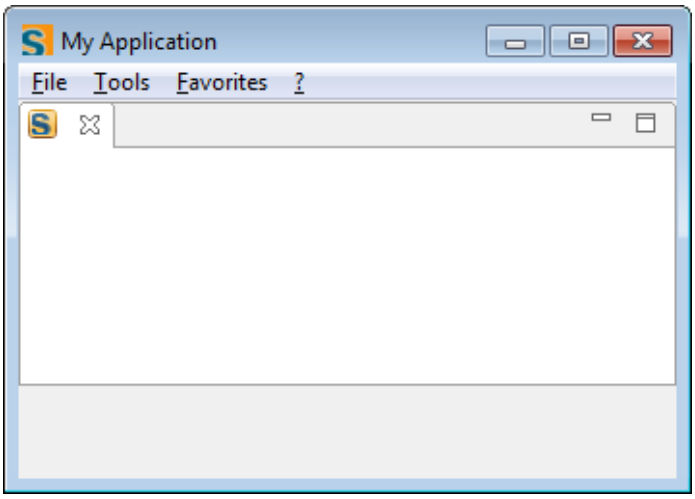


Figure 5. Running the three client applications. Each client displays an empty desktop form. The RAP client, the Swing client, and the SWT client

Having started the Scout server and all client products, the running client applications should look as shown in Figure Running the three client applications.

Add the User Interface Widgets

The project creation step has created a Scout client that displays an empty desktop form. We will now add widgets to the client's desktop form that will later display the "Hello World!" message.

To add any widgets to the desktop form, navigate to the *DesktopForm* in the Scout Explorer. For this, click on the orange client node in the Scout Explorer view. Then, expand the *Forms* folder by clicking on the small triangle icon, and further expand the *DesktopForm*. As a result, the *MainBox* element becomes visible below the desktop form as shown in Figure New Form Field Menu. With a click of the right mouse button over the *MainBox*, the available context menus are displayed. To start the form field wizard select the **New Form Field ...** menu.

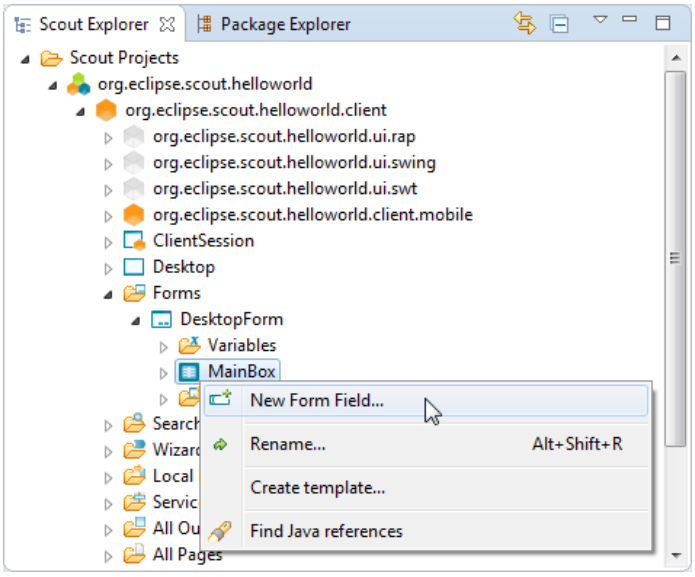


Figure 6. Using the **New Form Field ...** *menu to start the form field wizard provided by the Scout SDK.*

In the first step of the form field wizard shown on in Figure Add the DesktopBox field choose GroupBox as the form field type and click on the **[Next]** button. In the second wizard step, enter 'Desktop' into the *Class Name* field before you close the wizard with the **[Finish]** button. The Scout SDK will then add the necessary Java code for the DesktopBox in the background.

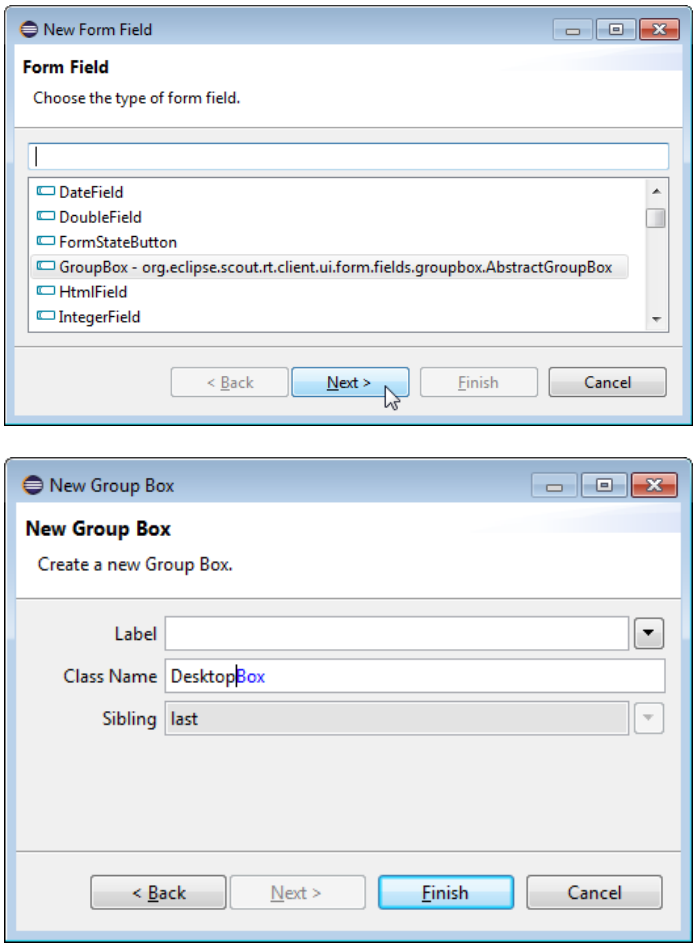


Figure 7. Adding the DesktopBox field with the Scout SDK form field wizard.

We can now add the text field widget to the group box just created. To do this, expand the *MainBox* in the Scout Explorer view to access the newly created *DesktopBox* element. On the *DesktopBox* use the **New Form Field ...** menu again. In the first wizard step, select *StringField* as the form field type according to Figure Add a StringField. To select the *StringField* type you can either scroll down the list of available types or enter "st" into the field above the field type list. In the second wizard step, enter 'Message' into the *Label* field. As we do not yet have the text 'Message' available in our "Hello World" application the wizard prompts the user with the proposal `New Translated Text` .

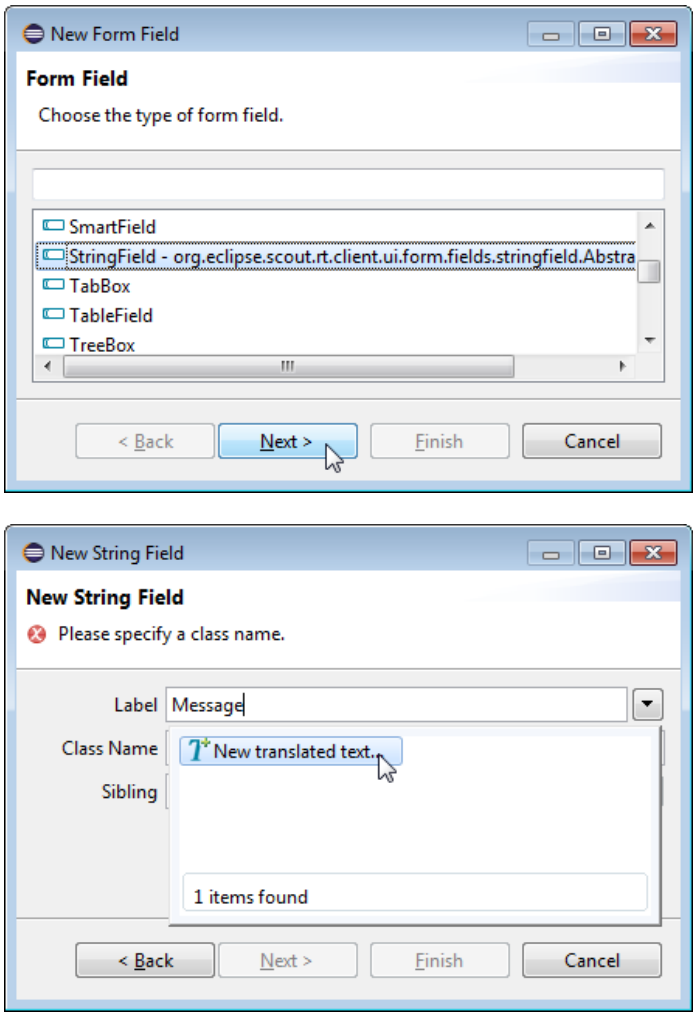


Figure 8. Adding a StringField and providing a new translation entry.

With a double click on this option a new text entry can be added to the application as shown in Figure Add a new translation entry. Once an initial translation for the message label is provided, close the translation dialog with the **[Ok]** button. Finally, close the form field wizard using the **[Finish]** button.

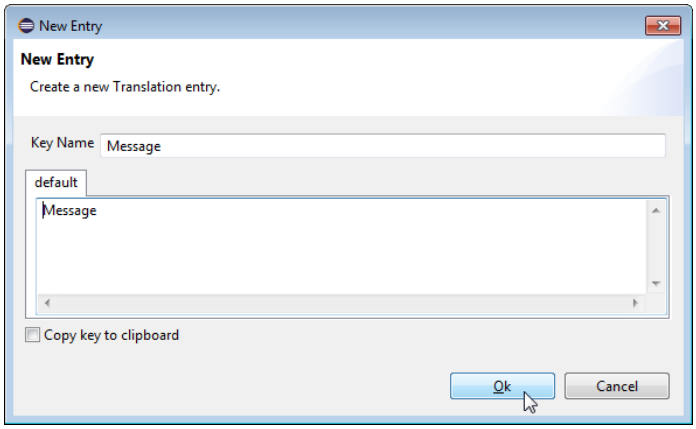


Figure 9. Adding a new translation entry.

By expanding the *DesktopBox* element in the Scout Explorer, the new message field becomes visible. Now, double click on the message field element to load the corresponding Java code into an editor and displays the message field's properties in the Scout Object Properties as shown in Figure showing

MessageField. This is a good moment to compare your status with this screenshot. Make sure that both the Java code and the project structure in the Scout Explorer look as shown in Figure showing MessageField.

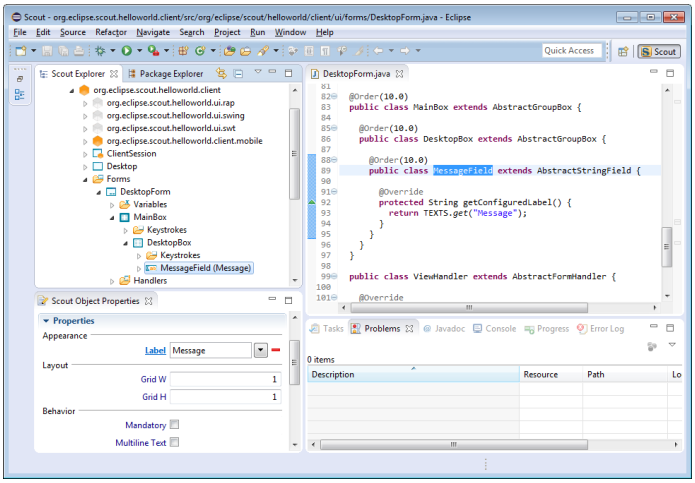


Figure 10. Scout SDK showing the MessageField

Having verified your status of the “Hello World” application start the start the server and a client of the application as described in the previous section. The client applications will then display your message widget. However, the text widget is still empty, as we did not yet load any initial content into this field. This is the topic of the next section.

Implement the Server Service

The responsibility of the Scout server in our “Hello World” application is to provide an initial text content for the message field in the client’s user interface. We implement this behaviour in the load method of the server’s DesktopService. An empty stub for the load method of the DesktopService service has already been created during the initial project creation step.

To navigate to the implementation of the desktop service in the Scout SDK, we first expand the blue top-level *server* node in the Scout Explorer. Below the server node, we then expand the *Services* folder which shows the *DesktopService* element. Expanding this *DesktopService* node, the load method becomes visible as shown in Figure showing Server node.

Figure 11. The Scout Explorer showing the blue server node expanded with the Services folder. In this folder the load method of DesktopService is selected and its initial implementation is shown in the editor on the right side.

The DesktopService represents the server service corresponding to the DesktopForm on the client side.

This initial setup represents Scout's default where client forms and server services typically come in pairs. Whenever the client's user interface displays a form to the user, the client connects to the server and calls the load method of the corresponding server service. We just need to add our business logic to the load method of the server's DesktopService.

According to the signature of the load method, a formData object is passed into this method that is then handed back in the return statement. To complete the implementation of the load method it is sufficient to assign the text 'hello world!' to the message field part of the form data. The complete implementation of the load method is provided in Listing load() implementation.

Listing 1. load() implementation in the DesktopService.

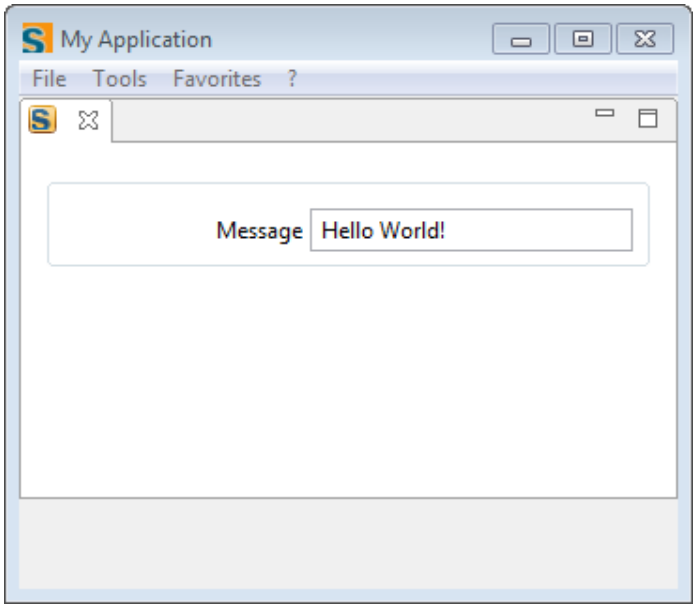
```
@Override
public DesktopFormData load(DesktopFormData formData) throws ProcessingException {
  formData.getMessage().setValue("Hello World!"); // <1>
  return formData;
}
```

① assign a value to the value holder part of the FormData corresponding to the message field

With this last element we have completed the Scout "Hello World" application.

Run the Final Application

We are now ready to run the completed "Hello World!" application by first starting the server and then the clients. This results in running clients as shown in Figure Running the complete Hello World application. The mobile version of the client can be started from the Scout SDK by clicking on the *Smartphone Devices* link in the product launchers section. Alternatively, manually change the applications URL from http://localhost:8082/web to http://localhost:8082/mobile.



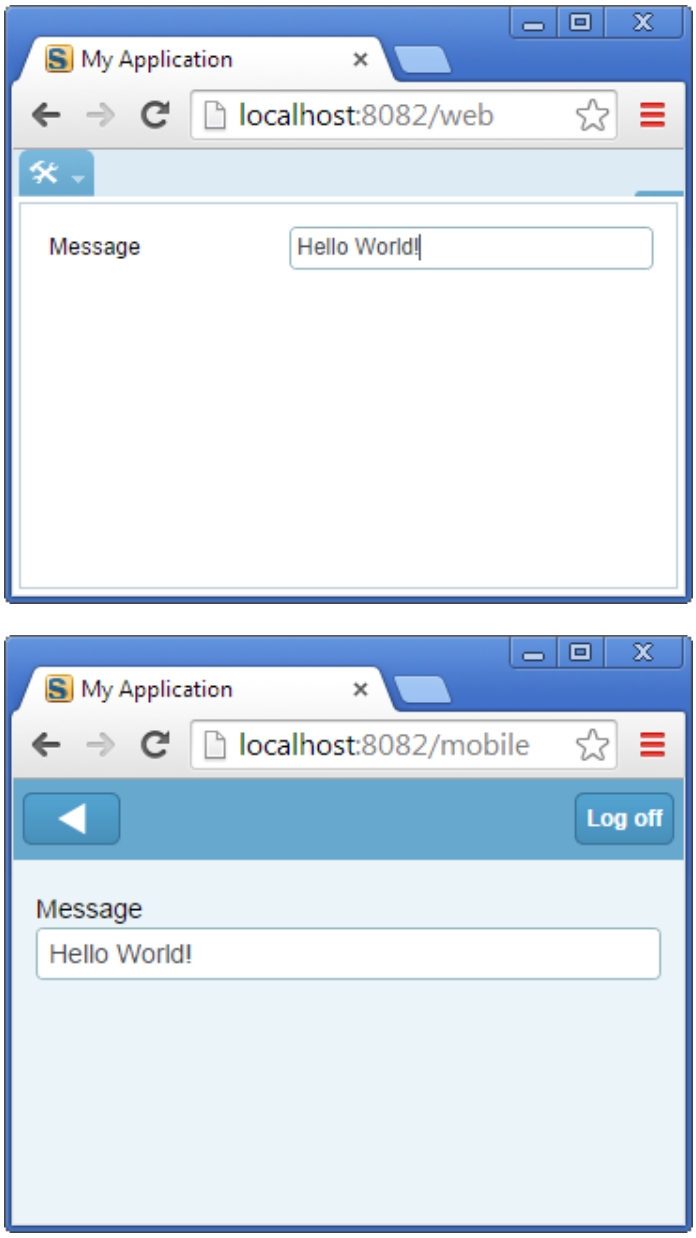


Figure 12. Running the complete “Hello World!” application with an SWT client, as a web application and a mobile application.

Congratulations, you just have implemented your first Scout client server application!

What’s Next?

Now that you have successfully created your first Scout application, you might want to learn more about Scout. To gain experience with Scout, you can follow more tutorials and start to read in the Scout books. If you prefer "Learning by doing" browse the available Wiki tutorials and go for the subset that matches your interests.

http://wiki.eclipse.org/Scout/Tutorial

If you are interested in Scout’s concepts, architecture and features you probably want to start reading.

For this, we are writing the Scout books.

http://wiki.eclipse.org/Scout/Book

In case you should get stuck somewhere and need help, try to get answers by searching the web. And if despite reasonable efforts this approach does not help, contact us on the forum. Should you have solved issues on your own, please consider sharing your findings in the Scout forum as this can help other folks too.

http://www.eclipse.org/forums/eclipse.scout

We wish you all the best on your journey with Scout and hope to hear from you in the Scout forum.