IBM HATS PORTLET MESSAGING

IBM WebSphere **Host Access Transformation Services** is an easy-to-implement Web-to-host solution that delivers HTML to users' Web browsers, extending legacy applications to end users on the Web.

HATS applications can run directly within WebSphere Portal as portlets. By integrating HATS with WebSphere Portal server, data can flow freely between HATS, WebSphere Portal, and other portal applications. IBM WebSphere® Host Access Transformation Services (HATS) portlet messaging is an integration of HATS, WebSphere Portal, and the IBM Portal Toolkit.

This article provides an overview of the implementation of HATS portlet messaging.

We shall discuss two applications:

MessageSender Portlet

MessageReceiver Portlet

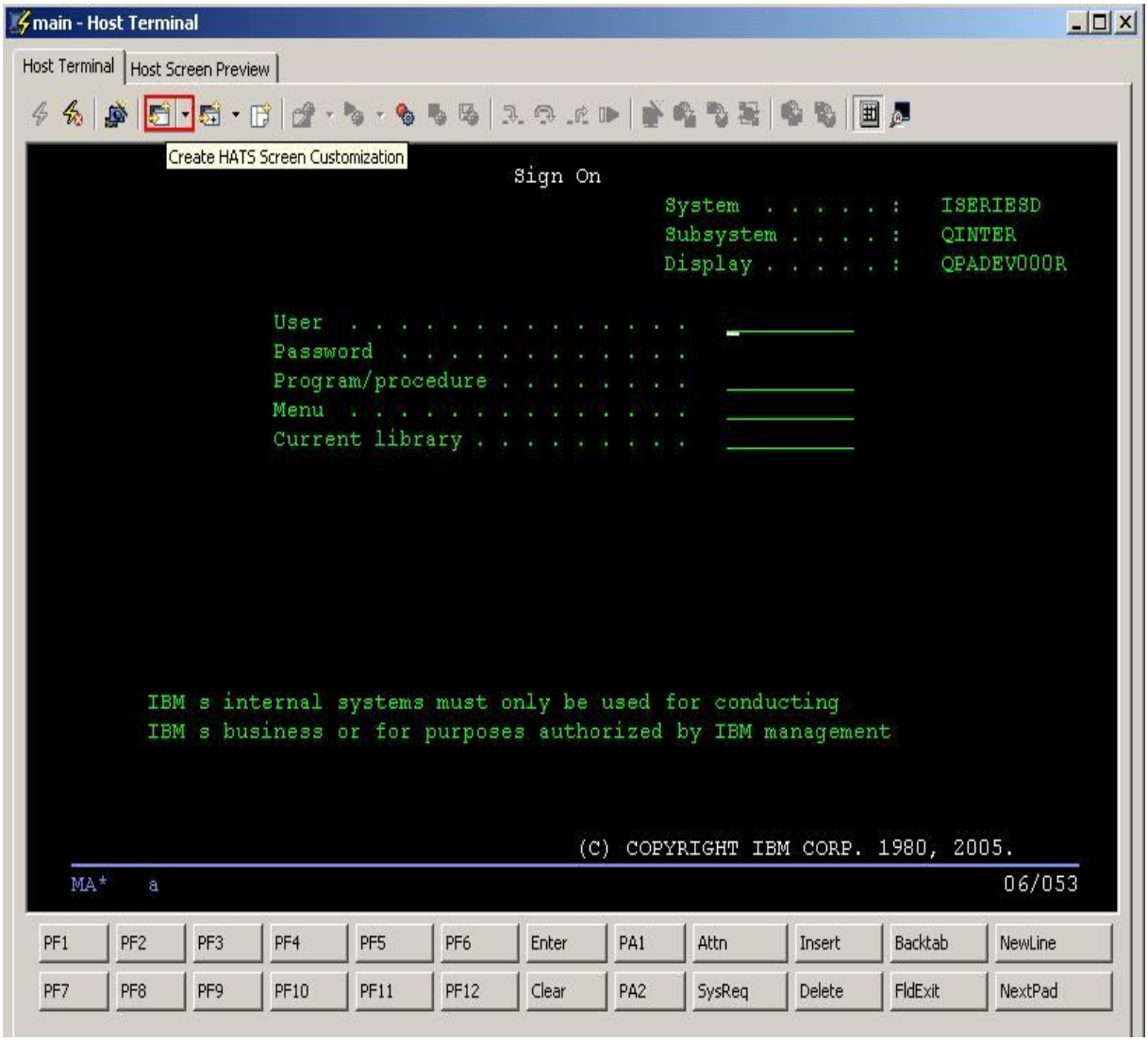
Let us consider the **MessageSender** Portlet first:

CREATING THE *MessageSender* PORTLET:

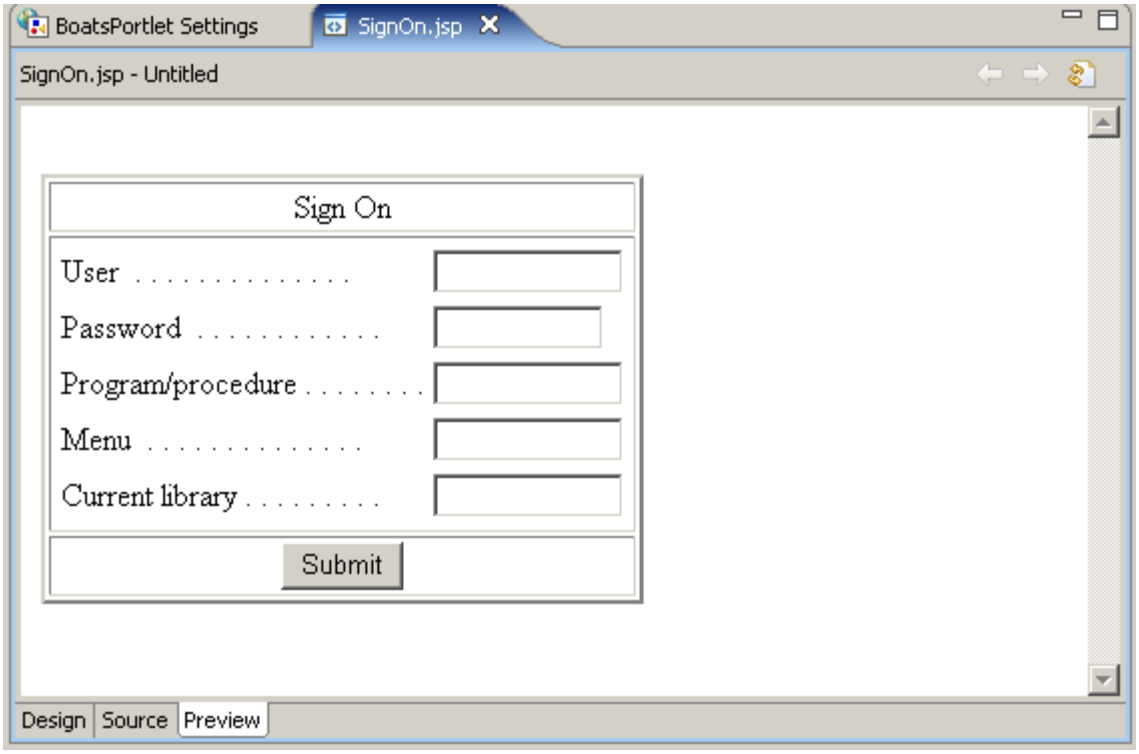
We create a new HATS project in which we specify the target server to be the **WebSphere Portal Server**.

In this project, we create simple customizations of IBM **iseries** server's **Sign On** screen and the **boats** application.

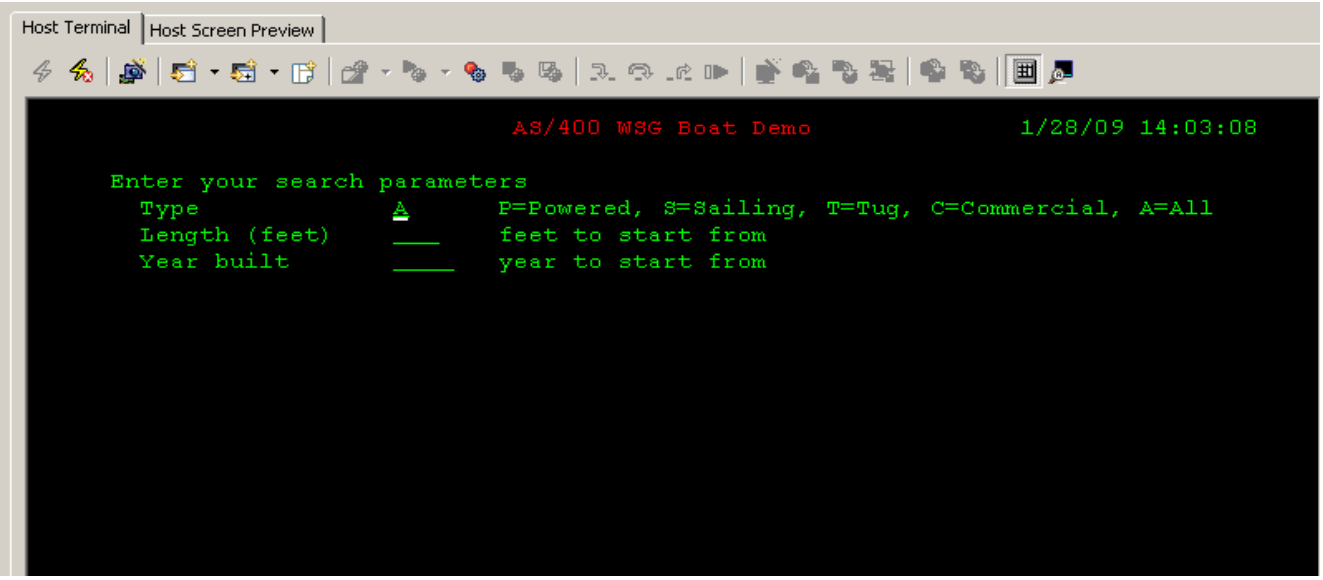
Firstly we create a simple customization for the **Sign On** screen that runs a transformation as an action:



The transformation of the above screen appears as below:

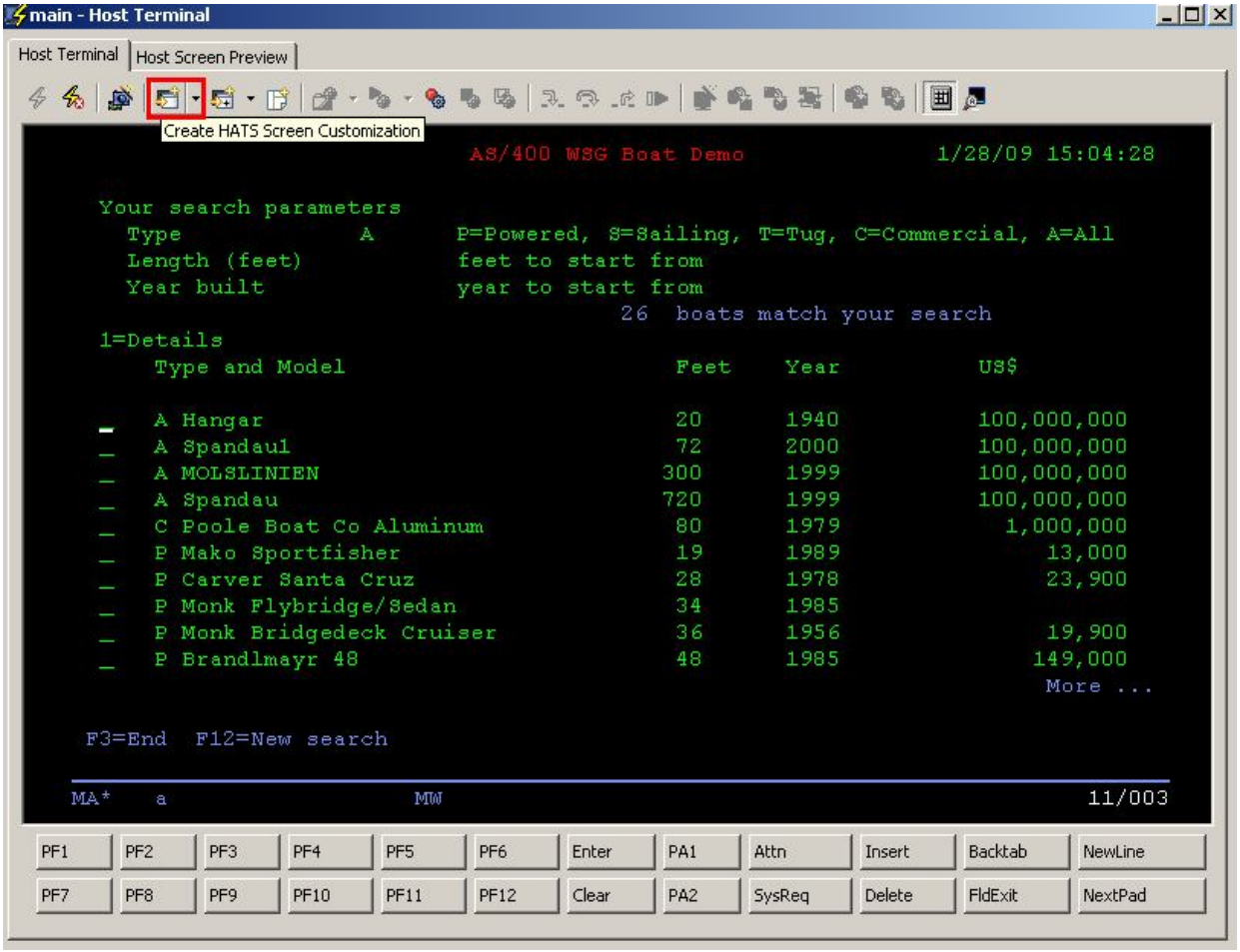


Now we create a customization for the **boats** search screen:



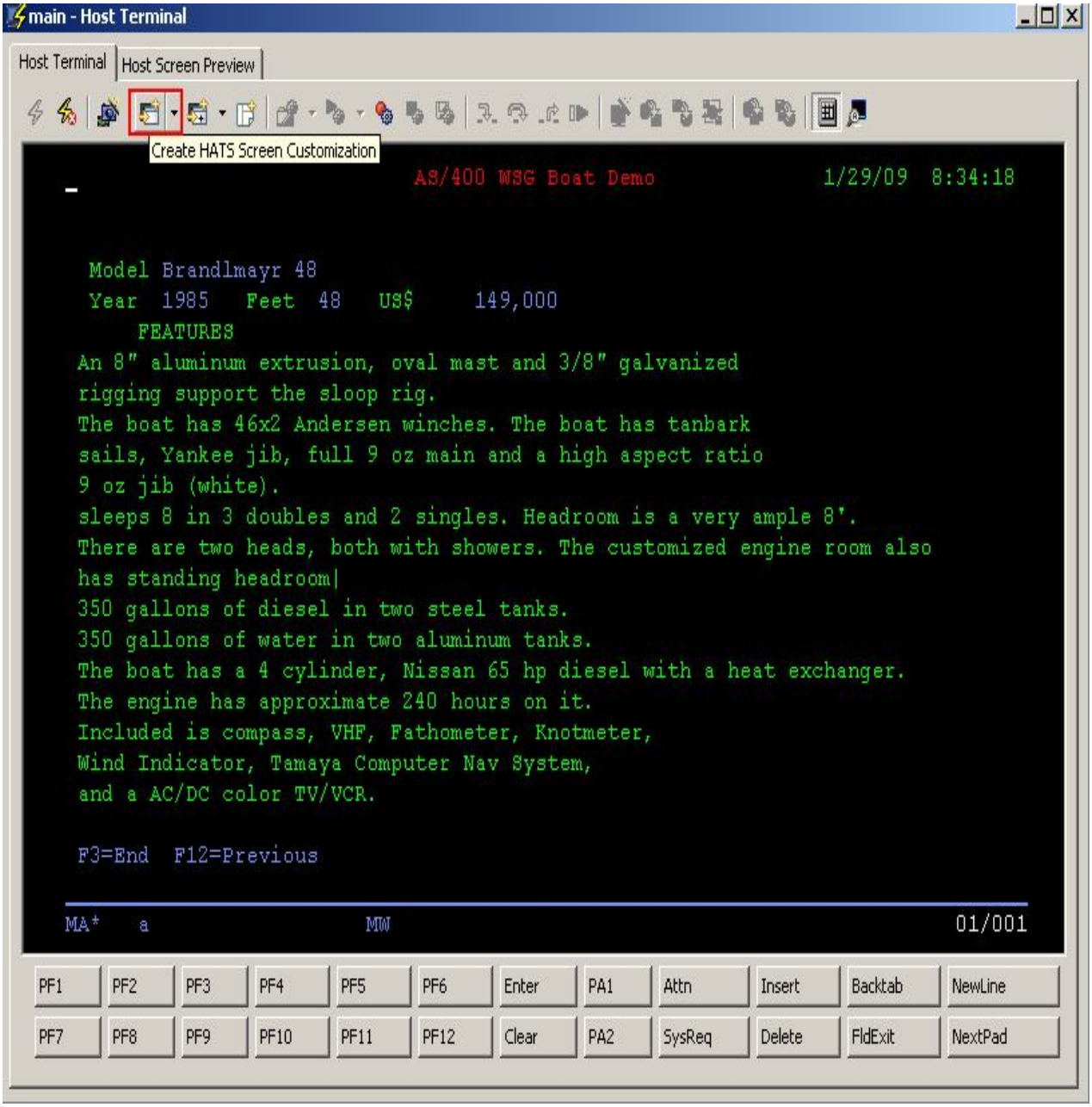
The above screen is transformed as shown below:

Similarly, we create a customization for the **boats** details screen:

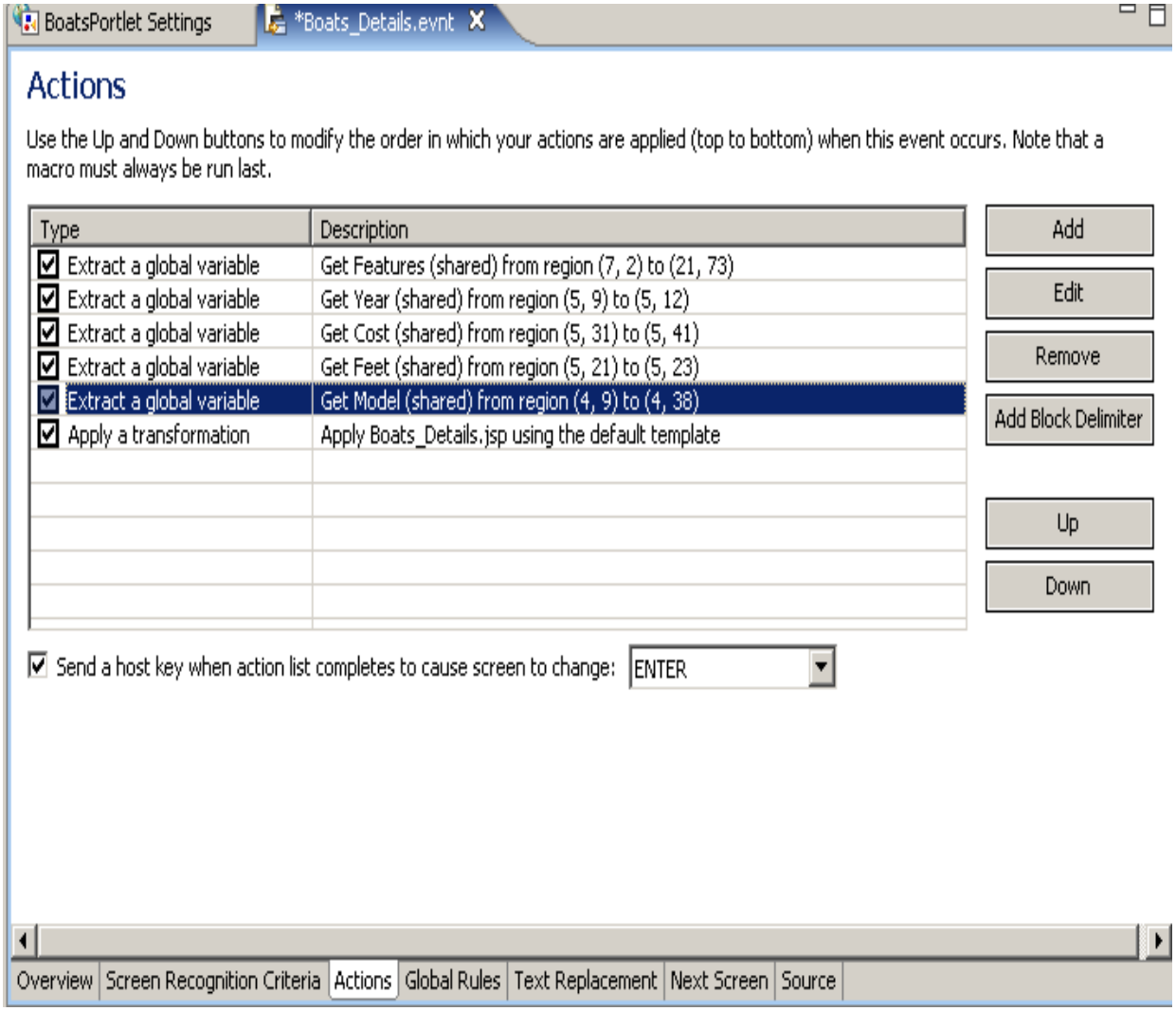


And the transformation appears as below:

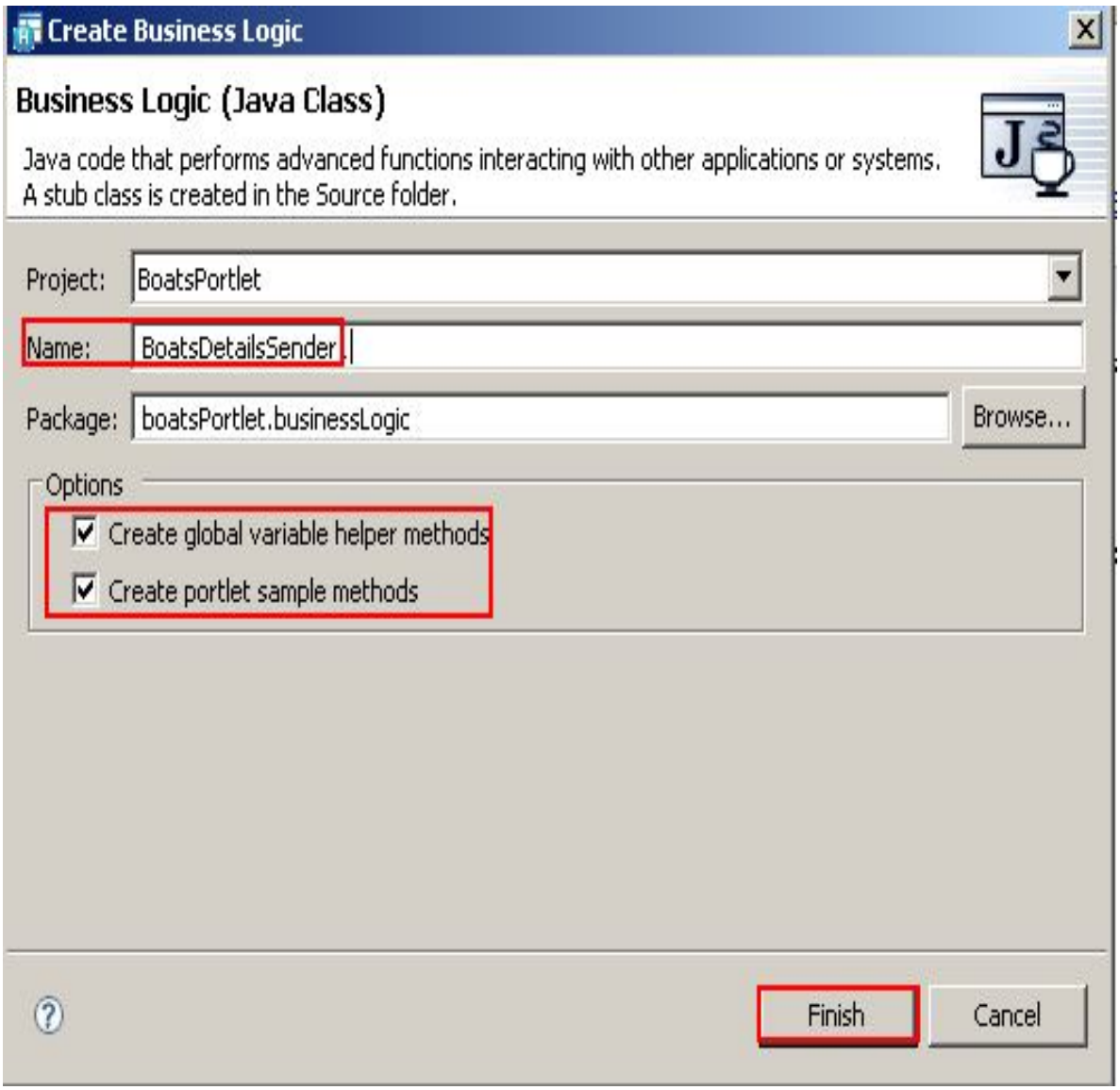
Similarly, a customization for the details screen of any particular boat is also created:



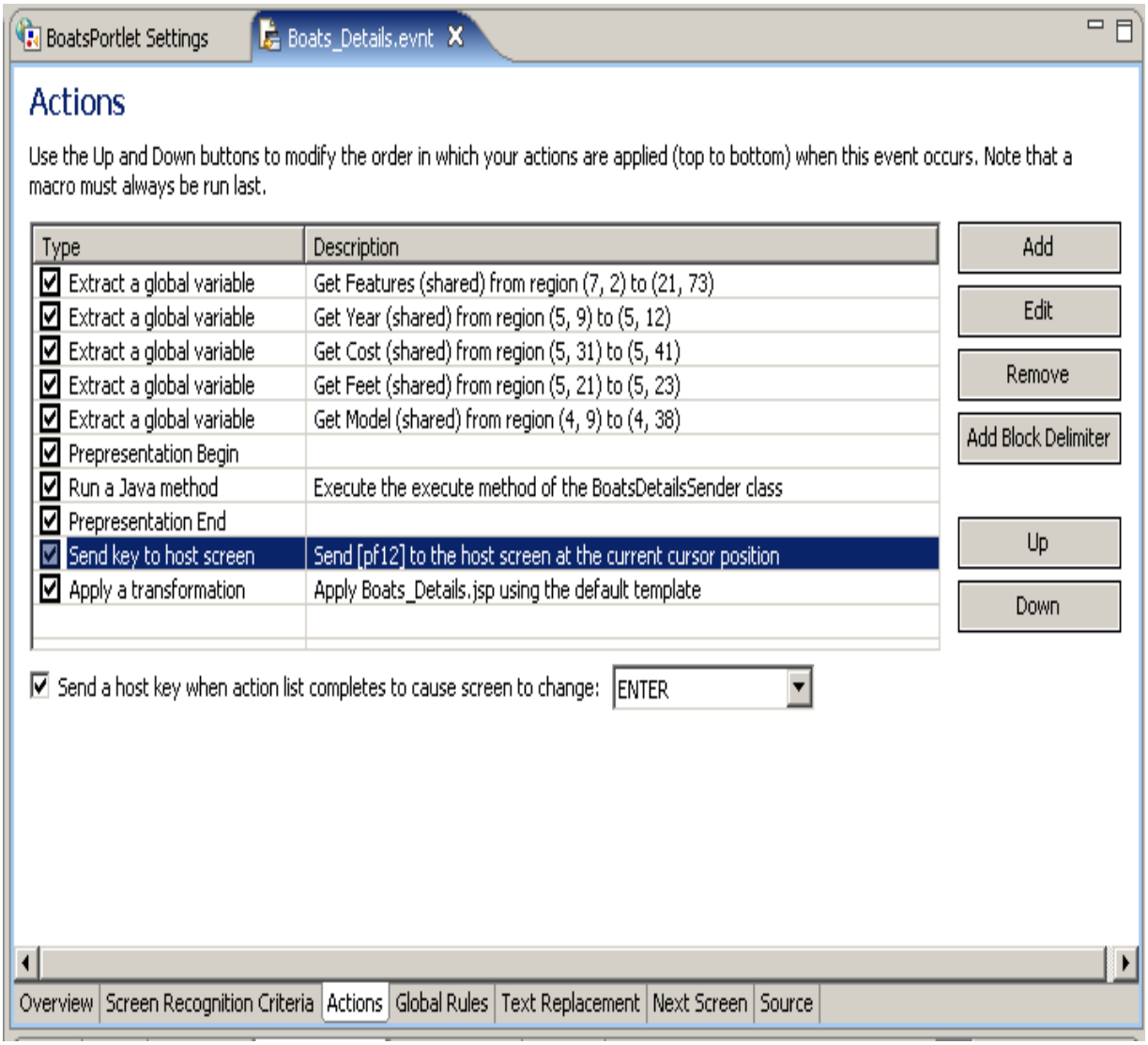
The following information is extracted from the above screen:



Now we create **BoatsDetailsSender** (**MessageSender**) business logic as follows:



Now we insert our business logic and other necessary actions into the last created customization as shown below:

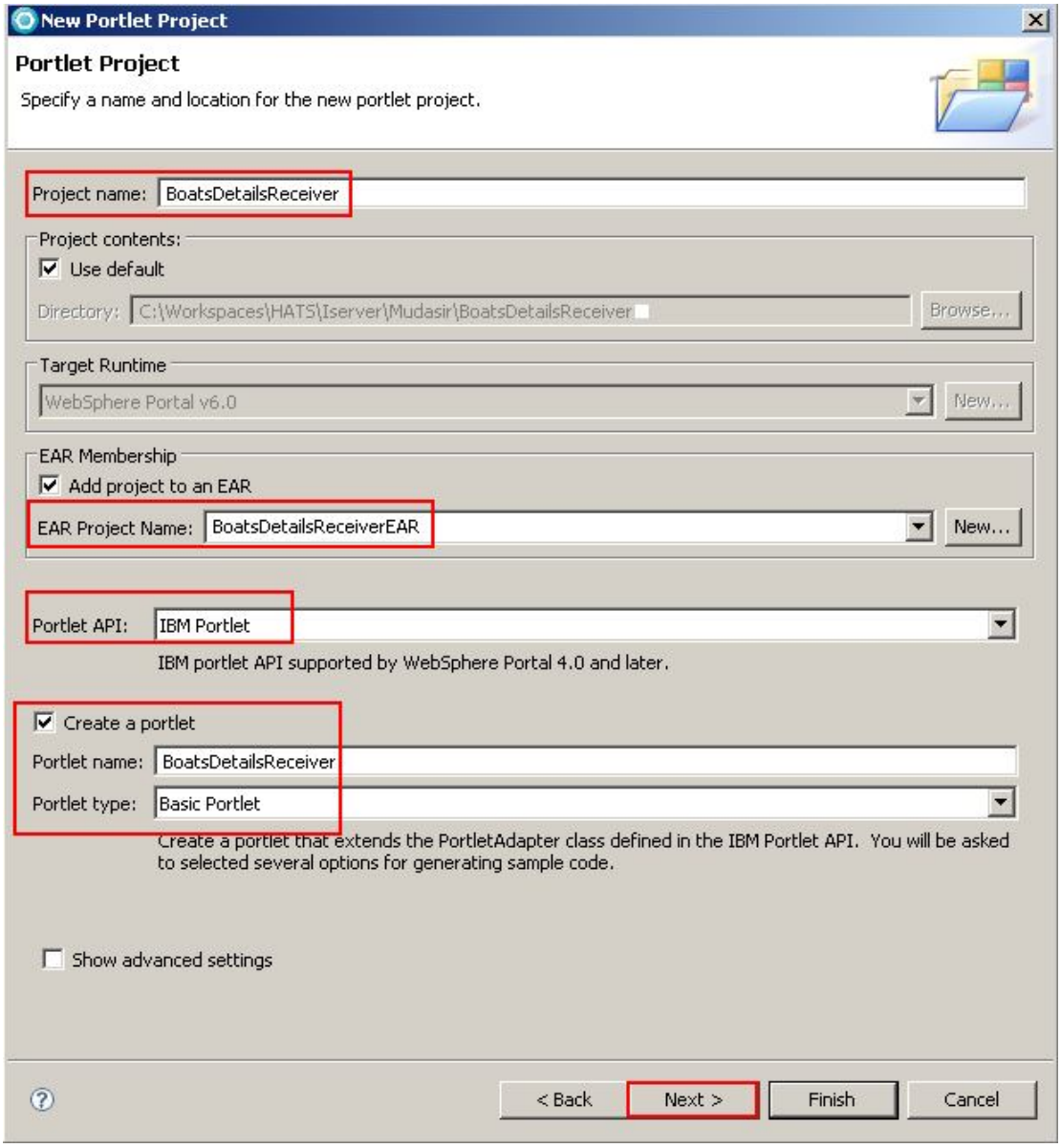


Next, we modify our business logic Java file to broadcast the Portlet message to the receiver Portlet application. This completes the development of **MessageSender** application and the WAR can now be generated which will be deployed later.

We now move on to create the **MessageReceiver** portlet.

CREATING THE *MessageReceiver* PORTLET:

We create a new portlet project:



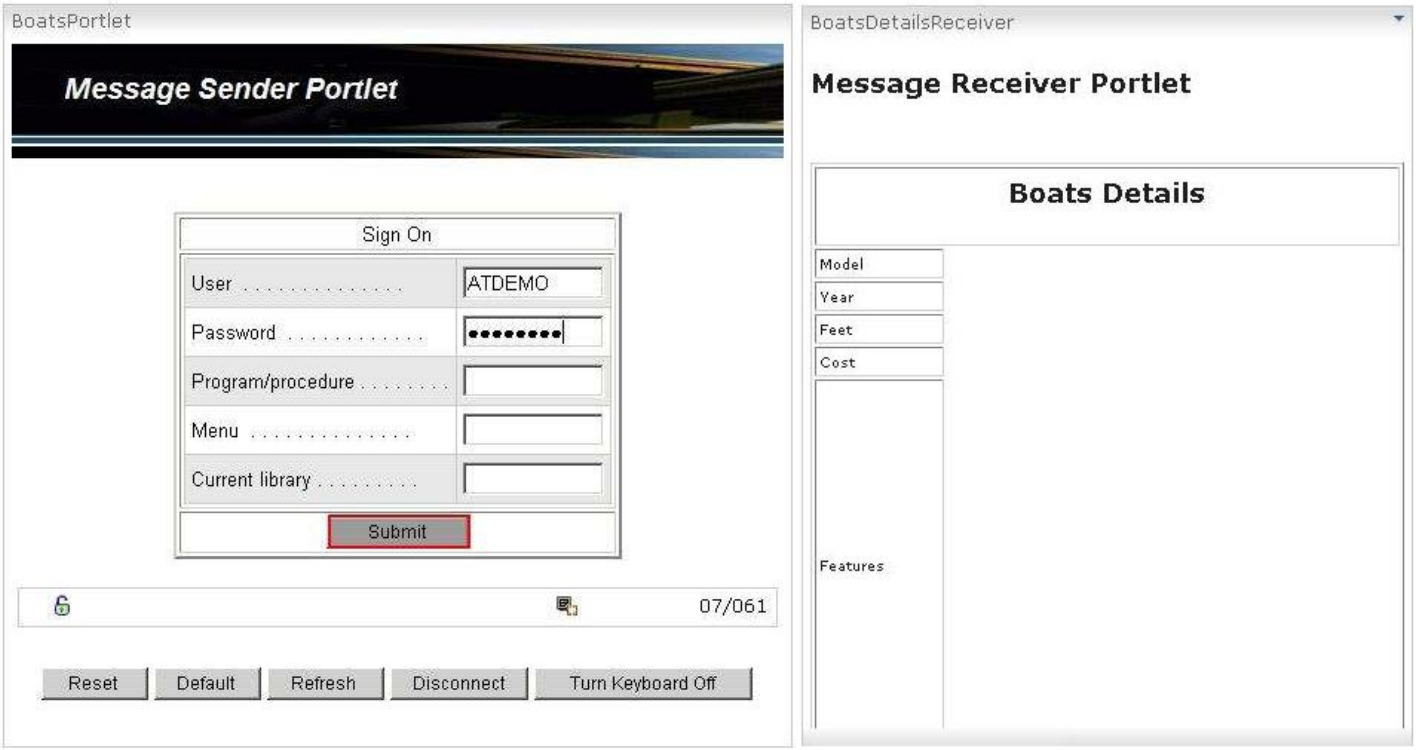
We now modify the **BoatsDetailsReceiverPortlet.java** class to include the message receiving functionality.

Next, we modify the portlet JSP to receive the sent message from the **MessageSender** portlet and to display the details.

Now we can generate the WAR file of our **MessageReceiver** portlet.

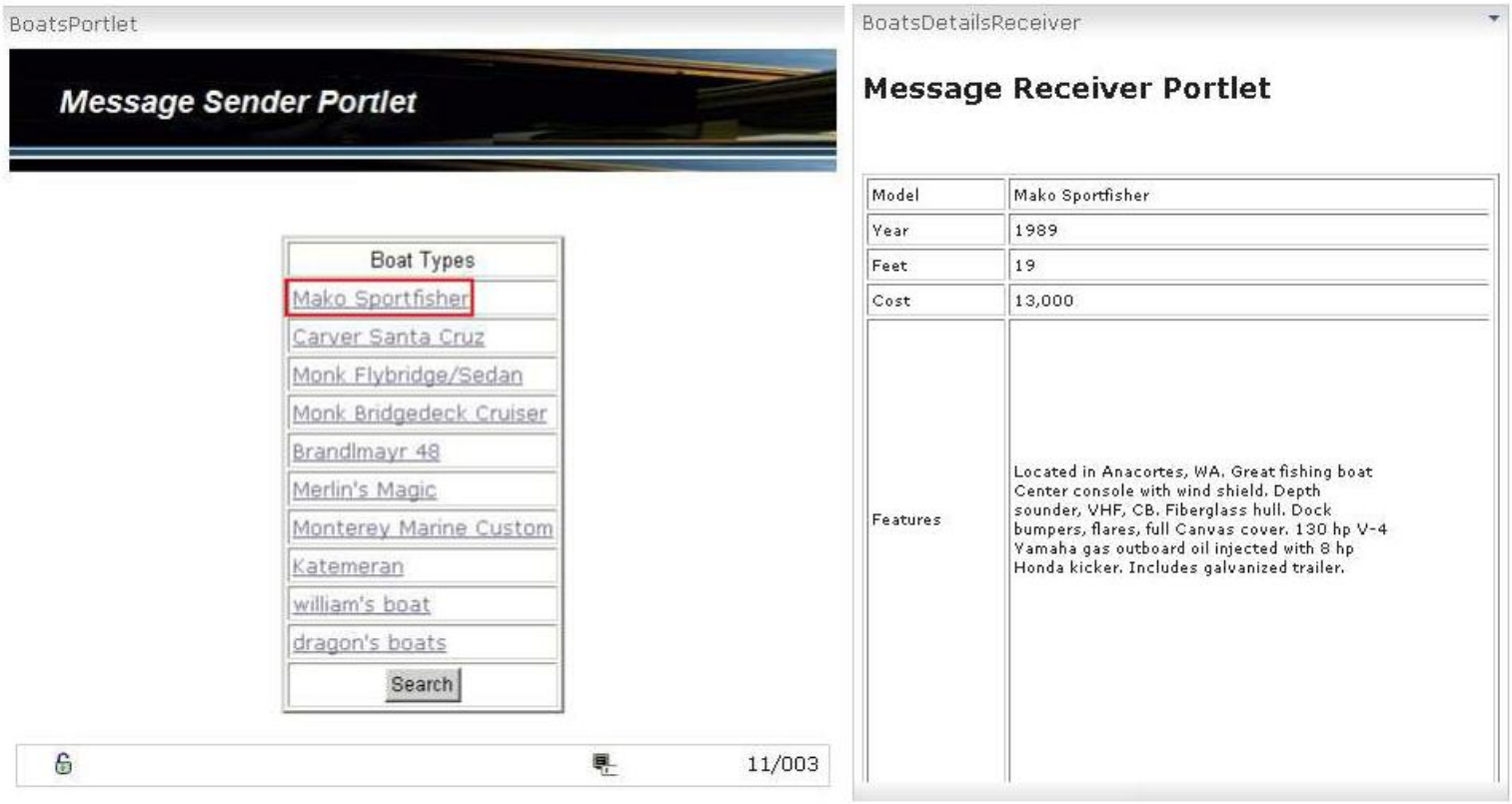
DEPLOYMENT AND TESTING ON THE WEBSPHERE PORTAL SERVER.

The WAR files of our **MessageSender** and **MessageReceiver** portlets can now be deployed on the WebSphere Portal Server. The screen below shows the login screen on the **MessageSender** portlet running on the Portal Server and its corresponding **MessageReceiver** screen. Since there is no data on the **Sign On** screen that can be displayed on the **MessageReceiver** portlet, it displays the JSP without any details as shown below



:

After signing on and navigating to the boat search results and clicking on one of the results in the **MessageSender** (left) portlet, the details of our selected item are displayed in the **MessageReceiver** (right) portlet JSP as shown below:



IBM Global Services
Route 100
Somers, NY 10589
U.S.A.
Produced in the United States of America
08-10