calgary scientific

PureWeb® STK 4.0

Quick Start Guide: Java Service

About Calgary Scientific

Calgary Scientific Inc. is dedicated to providing advanced visualization, web enablement, and mobility enhancement solutions to industries looking for secure access and use of their data or graphics intensive applications, while using their existing systems. Visit www.calgaryscientific.com for more information.

Notice

Your Responsibility for Your System's Security

You are responsible for the security of your system. Product administration to prevent unauthorized use is your responsibility. Your system administrator should read all documents provided with this product to fully understand the features available that reduce your risk of incurring charges for unlicensed use of Calgary Scientific products.

Trademarks

Released by

Calgary Scientific Inc. www.calgaryscientific.com.

Document Version: PW4.0_JavaService_QSG_07-2013_v1.000.00

calgary scientific

Table of Contents

Preface

Welcome to the *PureWeb STK 4.0 Quick Start Guide: Java Service*, a document that provides instructions to install the PureWeb STK, and to build, run and troubleshoot sample PureWeb Java service applications.

Intended Audience

This document is primarily intended for software developers who plan to install and use the STK to PureWeb enable a Java service. It has been written with the assumption that you have a working knowledge of:

- Java
- Eclipse

Making Comments on This Document

If you especially like or dislike anything about this document, feel free to e-mail your comments to techpubs@calgaryscientific.com.

Contacting Calgary Scientific Support

Use one of the methods in the table below to contact Calgary Scientific support.

Web Site	E-Mail
support.getpureweb.com	support@getpureweb.com

Chapter

1 Introduction

PureWeb® is a platform that enables the rapid transformation of enterprise software into cloud-ready, web and mobile applications.

The PureWeb STK (software transformation kit) is a set of tools that provides developers everything they need to implement the PureWeb solution in their own applications, in particular service and client APIs, documentation and API reference material, and sample applications in each supported programming language.

The sample applications illustrate key concepts of PureWeb enablement; they can be used by developers as a starting point to create their own PureWeb-enabled applications.

The PureWeb *Quick Start Guide* series is intended to get developers up and running as quickly as possible, using these sample applications as models.

As its name indicates, the current document, *PureWeb STK Quick Start Guide: Java Service*, focuses on the Java sample service. The guide provides instructions to install the PureWeb STK, and to build, run and troubleshoot a sample service.

Because the PureWeb enablement of a service is usually done in tandem with the creation of a client, the sample Java service comes with two built-in sample Java Swing clients, whose functionality is also described in this document:

- Scribble, which allows users to draw lines on a canvas, change line color, and erase lines.
- Asteroids, a PureWeb-enabled implementation of the classic video game with a twist -- it supports both single-player and two-player modes; each player independently controls his or her own ship.

If you prefer to work with a sample client other than Java Swing, refer to the *Quick Start Guide* for the client development platform of your choice.

Recommended Reading

This *Quick Start Guide* will get you up and running quickly, but it does not provide in-depth information; you should use it in conjunction with the other guides of the PureWeb STK documentation suite, in particular:

- *Installation Guide*
- *Developer's Guide*
- *Server Administration Guide*
- Java service API reference

To access these references in your Windows operating system, press the `Windows` key on your keyboard to display the **Start** menu (**Start** screen in Windows 8), then type "pureweb" as the search term.

Chapter

2 Deployment Procedures

In this chapter, you will find prerequisite information as well as instructions to install the PureWeb STK and build the sample Java services.

Because a service application is normally developed in tandem with a client application, the instructions below will also build Java Swing sample clients. If you prefer to work with the sample in a different client development platform, refer to the *Quick Start Guide* document for that client environment.

For more detailed instructions, such as how to handle a side-by-side installation of the STK, please refer to the complete *PureWeb STK Installation Guide*.

Prerequisites

To install the PureWeb STK, you will need:

- Java Development Kit (JDK) 1.7
- C++ Runtime, either of: 2008 SP1 ATL x86/x64 or 2010 SP1 x86/x64; the 2010 version is available from the following location: www.microsoft.com/en-us/download/details.aspx?id=5555
- A Windows 64-bit operating system, either of: Windows 7, Windows 8, Windows Server 2008 R2, or Windows Server 2012

Note: Although only 64-bit platforms are supported for application development, you can target the applications developed using the PureWeb STK to deploy on either a 32-bit or a 64-bit platform.

In addition, to build and run the sample Java services, you will need:

- Apache Ant 1.8.1
- A Java IDE, such as Eclipse Juno 4.2 (recommended)

Although not strictly required to work with the sample application, a Java IDE provides the obvious benefits of speeding development and facilitate debugging.

An IDE is necessary to debug the sample application interactively, which is the method used in this document. Any other IDE may be used, but details will differ according to whichever IDE is chosen.

Setting Environment Variables

The PureWeb server requires the `JAVA_HOME` environment variable to be pointing to the correct instance of the JDK. If building with Ant, you will also need to set `ANT_HOME`. In both cases, the `Path` environment variable must also be set.

On a Windows operating system, to access the Environment Variables dialog:

1. Press the `Windows` key on your keyboard to display the **Start** menu (or **Start** screen in Windows 8).
2. Type "environment variables" to search for that string.
3. Select the option **Edit the System Environment Variables** from the search results. This will display the System Properties dialog box's Advanced tab.
4. Click the **Environment Variables** button.

JAVA_HOME

Check that you have a `JAVA_HOME` entry under System variables:

- If the entry is there and the path points to version 1.7 of the JDK, skip this section and check your `ANT_HOME` variable, as described in next section.
- If the entry is there, but the path points to the wrong version of the JDK: click the **Edit** button, change the path in the **Variable value** field and click **OK**.
- If the entry is missing:
  a. Click the **New** button.
  b. Type `JAVA_HOME` in the **Variable name** field.
  c. Enter the path to JDK 1.7 in the **Variable value** field (for example, the default for update 6 is `C:\Program Files\Java\jdk1.7.0_6`).
  d. Click the **OK** button to save your changes.

ANT_HOME

Check that you have a `ANT_HOME` entry under System variables:

- If the entry is there and the path points to your instance of Ant 1.8.1, skip this section and check your `Path` variable, as described in next section.
- If the entry is there, but the path points to the wrong version of Ant: click the **Edit** button, change the path in the **Variable value** field and click **OK**.
- If the entry is missing:
  a. Click the **New** button.
  b. Type `ANT_HOME` in the **Variable name** field.
  c. Enter the path to Ant 1.8.1 in the **Variable value** field (default is `C:\Program Files (x86)\apache-ant-1.8.1`).
  d. Click the **OK** button to save your changes.

Path

If your `Path` system variable is missing either the path to the JDK or to Ant:

1. In the list of system variables, select the `Path` variable.
2. Click the **Edit...** button.
3. In the **Variable value** string, add the following if they are missing:
   - `JAVA_HOME: %JAVA_HOME%\bin;`
   - `ANT_HOME: %ANT_HOME%\bin;`
4. Click the **OK** button to save your changes.
5. Click **OK** to close the Environment Variables dialog box.
6. Click **OK** to close the System Properties dialog box.

Note: After changing environment variables, if you had a console window or Eclipse open, you will need to restart it for these changes to take effect.

You will also need to stop and restart the PureWeb server.

Installing PureWeb

If you already have PureWeb installed, uninstall it first.

To install the PureWeb STK in a Windows operating system:

1. Double-click on its executable file, `PureWeb_Win_Setup-[version].exe`, and follow the instructions in the wizard. The installation directory path cannot contain any spaces.
2. Install your PureWeb server license, a .lic file which you will have received from Calgary Scientific separately from the STK:
   a. Navigate to `[Installed_directory]\Server\conf\`.
   b. Place the .lic license file into that folder.
   c. Double-click on the **Start PureWeb** desktop icon to start the server.
3. Check that the following environment variables are set correctly:
   - `PUREWEB_HOME:`
     `[Installed_directory]\Server`
   - `PUREWEB_LIBS:`
     `[Installed_directory]\SDK\Redistributable\Libs`

Building the Sample Applications

The procedure below builds both the Scribble and the Asteroids applications.

1. Open a console (command prompt) window.
2. Change the directory to: `[Installed_directory]\SDK\Samples\Java`
3. Type `ant` to run the `build.xml` script .This generates two .jar files and two .bat files in the `\dist` directory:
   - `CSI.PureWeb.Client.Samples.jar`
   - `CSI.PureWeb.Server.Samples.jar`
   - `RunScribbleClient.bat`
   - `RunAsteroidsClient.bat`

Chapter

3 Sample Applications Overview

This chapter contains instructions on how to run the Scribble and Asteroids sample applications, and describes the functionality available in each.

Note: The procedures describe the Java Swing clients. If you plan to work with a different sample client, refer to the *Quick Start Guide* for that client development environment.

Launching the Applications

Launching the Java service application is a three-step process, which involves reloading the plugins in the PureWeb server, running the .bat file, and logging in.

Reloading the Plugins

This step is only necessary the first time you build the service.

1. Launch the PureWeb server using the **Start PureWeb** desktop icon.
2. Open a browser.
3. Navigate to `http://localhost:8080/pureweb/config/plugins` (if applicable, replace `localhost:8080` with the correct server and port for your configuration of the PureWeb server).
4. Enter `admin` in both the **Name** and **Password** fields and click the **Sign In** button to open the Configuration page.
5. Click **Reload Plugins**, under the **PureWeb Server Configuration** heading.

Running the .bat File

1. Open a console (command prompt) window and change the directory to:
   `[Installed_directory]\SDK\Samples\Java\dist`
2. Type the run command applicable to your application:
   `RunScribbleClient.bat` or `RunAsteroidsClient.bat`

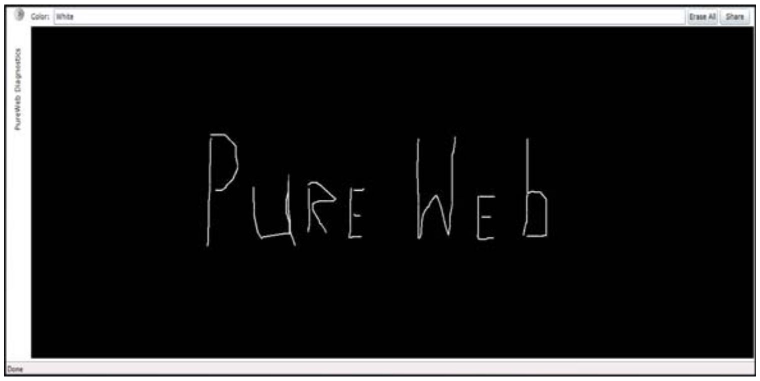
Logging In

1. If the server is not running, double-click on the **Start PureWeb** desktop icon.
2. Select the **Server | Connect** menu option to display the Connect to Server dialog.
3. Type the following in the **Server URL** field (if applicable, replace `localhost:8080` with the correct server and port for your configuration of the PureWeb server):
   - Scribble:
     `http://localhost:8080/pureweb/app?name=ScribbleAppJava`
   - Asteroids:
     `http://localhost:8080/pureweb/app?name=AsteroidsApp`
4. If using Asteroids, enter your name in the **Player name** field, or leave blank to play as Anonymous.
5. Click the **Connect** button to open the Authorization dialog.
6. Enter `admin` in both the **Username** and **Password** fields.
7. Click the **OK** button to start the application.

Exercising the Functionality

This section briefly describes the functionality available in each of the two sample applications, Scribble and Asteroids.

Scribble



The Scribble sample application is a simple canvas that users can use to draw lines. When you first launch it, the canvas is blank. You can draw lines on the canvas by left-clicking the mouse and dragging.

Scribble also offers the following functionality:

- Change colors by entering the name of a new color in the **Color** text field.
- Clear the canvas by clicking on the **Erase All** button.
- Share the application with other users by clicking the **Share** button (see Sharing the Application on page 14).
- Display the Diagnostics Panel by clicking on the small right arrow below the Server menu; see the Diagnostics Panel section below for more information.
- Disconnect using the application’s menu (**Server | Disconnect**) or by closing the client.

Notice also how the service and client windows are in sync:

- When you resize the client browser window, the service window resizes.
- When you close the client window, the service window also closes.

Diagnostics Panel

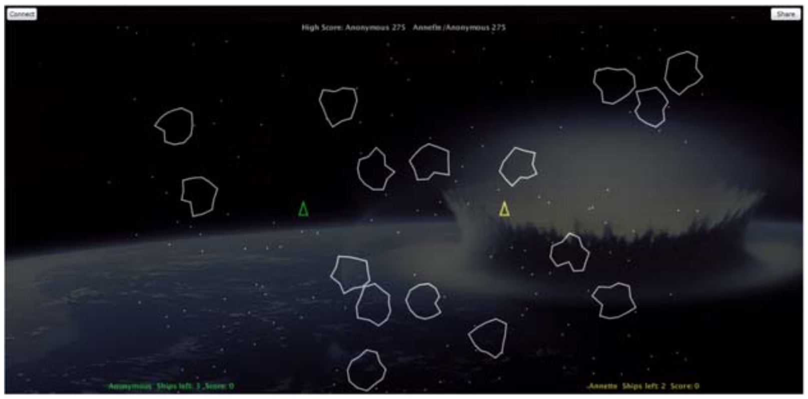
The Diagnostics Panel debugging tool is described in detail in the Debugging chapter of the *PureWeb® STK Developer Guide*, however below is a brief overview of the functionality it offers.

The panel is composed of three tabs: Options, App State, and Bandwidth.

- The **Options** tab is used to define options that impact image bandwidth and quality.
- The **App State** tab displays an XML representation of the application’s state.
- The **Bandwidth** tab is used to measure latency and bandwidth.

Asteroids

The Asteroids sample application is a PureWeb-enabled implementation of the classic video game with a twist -- it supports both single-player and two-player modes; each player independently controls his or her own ship.


Connect
Share
High Score: Anonymous 275 Annette/Anonymous 275
Anonymous Ships left: 3 Score: 0
Annette Ships left: 2 Score: 0

Single-Player Mode

- Press any key to start the game.
- Use the up/down arrow keys to move forward/reverse.
- Use the left/right arrow keys to change your heading.
- Press the space bar to fire torpedoes.
- Activate shields by pressing the `S` key. Each player gets 5 seconds of shield time per game.
- Collide with an asteroid with your shields up to destroy the asteroid and collect the points.
- Monitor the number of frames per second in the FPS field.
- disconnect using the application’s menu (**Server | Disconnect**) or by closing the client. When you close the client window, the service window also closes.

Two-Player Mode

Instructions on how to share the application with another user are provided in section Sharing the Application, below.

Once the application is shared, additional two-player mode functionality includes:

- Collide with an unshielded opponent with your shields up to destroy their ship and collect all of their points.
- Collide with a shielded opponent with your shields up to create an elastic collision.

Sharing the Application

Both the Scribble and Asteroids sample applications contains functionality to illustrate collaboration. This section describes how one user can share the application with a second user so that they are both logged in the same session.

A few important points to note before getting started:

- The sample client connects using the hostname of the local host. If you are trying to share the application from another computer, you must modify the URL, `http://localhost:8080/pureweb/...`, by replacing `localhost` with the server’s IP address or DNS name.
- Cookie sharing between browser tabs may cause issues with the functionality of the sample application. We recommend that you use two separate browser instances if you want to open two sessions of the same application. Chrome users can also use an incognito window.

First user:

- Click the **Share** button located in the upper-right corner of the application.
- Copy the URL displayed in the popup window and close the window.
- Provide the URL you copied to the person you want to share the session with.

Second user:

If using a Java client installed locally:

1. Launch the client using the .bat file.
2. Select the **Server | Connect** menu option.
3. In the **Server URL** field, enter the URL shared by the first user.
4. In the **Password** field, enter `Scientific`.

If using a Silverlight, Flex or HTML5 client (assuming it has been built):

1. Open a browser.
2. Navigate to the URL shared by the first user to display the **PureWeb Collaboration Login** page.
3. Select a preferred client from the drop down box. All possible browser-based clients are listed, but only the clients that have actually been built will work.
4. Enter `Scientific` as the password and click the **Sign In** button.

Chapter

4 Exploring the Code

One of the main purposes of including sample applications with the STK is for developers who are new to PureWeb to use these samples as a starting point for coding and to experiment with them.

Therefore, don't be shy about editing the code and trying new things. That's the best way to learn, and you'll see how quickly you'll become proficient with PureWeb.

This chapter gives some pointers about the code in the sample Java services; however, it assumes that you are familiar PureWeb concepts such as views, state, commands and event handlers, which are described in the *Developer's Guide*.

For a graphic illustrating the service-side classes discussed in this chapter, see page 20.

For an overview of the client-side code, please refer to the *Quick Start Guide* available for each supported client programming language.

Scribble Service Code Pointers

The files described in this section can be found in the PureWeb directory:

```
...\SDK\Samples\Java\src\server\pureweb\samples\scribbleapp
```

ScribbleApp.java File

This file contains the `main` method, used to creates an instance of the `StateManager` class, which is responsible for managing application state, responding to input events and commands sent from the client, generating responses such as updated images to send back to the client application. The `StateManager` accomplishes this by using a collection of plugins such as `XmlStateManager`, `CommandHandler`, and `ViewManager`.

The `main` method also creates an instance of `StateManagerServer`, which implements input and output threads that receive input events and commands from the client via the PureWeb server. The input events and commands are dispatched to the `StateManager` for processing. Responses are gathered from the `StateManager` and returned to the client via the PureWeb server.

Finally, the `main` method also starts the service application.

ScribblePanel.java File

The `ScribblePanel` class implements the business logic of the Scribble service application. It contains a number of methods and inner classes that are noteworthy from a PureWeb enablement perspective.

ScribblePanel.addNotify

This method is called when the panel is added to its parent component and performs a number of important actions including:

- Registering the `ScribblePanel` instance with the view manager.
- Adding an application state property change listener to listen for color changes.
- Adding a command handler to listen for erase commands from the client.
- Initializing the `ScribbleColor` application state property to the color WHITE.

OnScribbleColorChanged

This inner class implements the `EventHandler<ValueChangedEventArgs>` interface to respond to changes to the `ScribbleColor` application state property.

Whenever the client application changes `ScribbleColor`, a state difference is generated and transmitted to the service application. The `StateManager` merges the difference into the application state on the server-side, and notifies any registered event handlers of the change.

OnExecuteClear

This inner class implements the PureWeb `CommandHandler` interface to respond to `Clear` commands sent by the client application.

mouseHandler

Notice the calls to `setInteracting` and `remoteRender` in the `mousePressed`, `mouseReleased`, and `mouseDragged` methods.

setInteracting

PureWeb allows developers to tell the service to send lower quality images when the user is interacting with a view (thereby conserving bandwidth and reducing latency), and to restore original quality afterwards. This is done by setting the value for `SetInteracting` in the view's mouse and keyboard event handlers to either `true` (send reduced quality images during interaction) or `false` (restore image encoding to full quality). `SetInteracting` is a method on `ViewManager`.

The actual quality of the images during interaction can be set using the Diagnostics Panel, which is described in more detail in the *Developer's Guide*.

The `setInteracting` method of the `RemotedPanel` class calls the `setViewInteracting` method of the `remoteRenderer` class.

remoteRender

The call to this method notifies the PureWeb server that the scribble image has changed and that an updated image should be sent back to the client application at the next available opportunity.

Asteroids Service Code Pointers

The files described in this section can be found in the PureWeb directory:

```
...\SDK\Samples\Java\src\server\pureweb\samples\asteroids
```

AsteroidsApp.java File

This file contains the `main` method, which works the same way as the same class in the Scribble application:

- Creates an instance of the `StateManager` class, which is responsible for managing application state, responding to input events and commands sent from the client, generating responses such as updated images to send back to the client application; the `StateManager` accomplishes this by using a collection of plugins such as `XmlStateManager`, `CommandHandler`, and `ViewManager`.
- Creates an instance of `StateManagerServer`, which implements an IO thread that receives the input events and commands from the client via the PureWeb server; the input events and commands are dispatched to the `StateManager` for processing, and responses are gathered from the `StateManager` and returned to the client via the PureWeb server.
- Starts the server.

AsteroidsPanel.java File

The `AsteroidsPanel` class implements the game logic of the Asteroids service. It contains a number of methods and inner classes that are noteworthy from a PureWeb enablement perspective.

AsteroidsPanel.addNotify

This method is called when the panel is added to its parent component and performs a number of important actions including:

- Registering the `AsteroidsPanel` instance with the view manager.
- Enabling client scaling of the images sent back from the server. This means that the client-side view can be resized and the images will be scaled accordingly.
- Setting the full and interactive image qualities to 30 in order to reduce bandwidth consumption.

- Setting the initial game level to 0. The level state value is incremented each time all asteroids are destroyed, and the game progresses to a more difficult level.

ConnectSession and DisconnectSession

When a player connects to the Asteroids service, a `ConnectSession` command is sent. This allows PureWeb to establish a unique session ID identifying the player.

PureWeb notifies registered session connected event handlers whenever a `ConnectSession` command has been received and processed. The session connected handler is implemented by the `SessionConnected` inner class, which in turn implements the `EventHandler<SessionEventArgs>` interface. The handler responds to the event by binding the session ID to the first available ship, and performing various initialization actions. If `ConnectSession` binds to the second ship, the game is reconfigured to enter two-player mode.

Similarly, when a player disconnects from the Asteroids service, a `DisconnectSession` command is sent. PureWeb notifies registered session disconnected event handlers, which respond to the event by unbinding the ship corresponding to the session ID.

InputEventReceiver

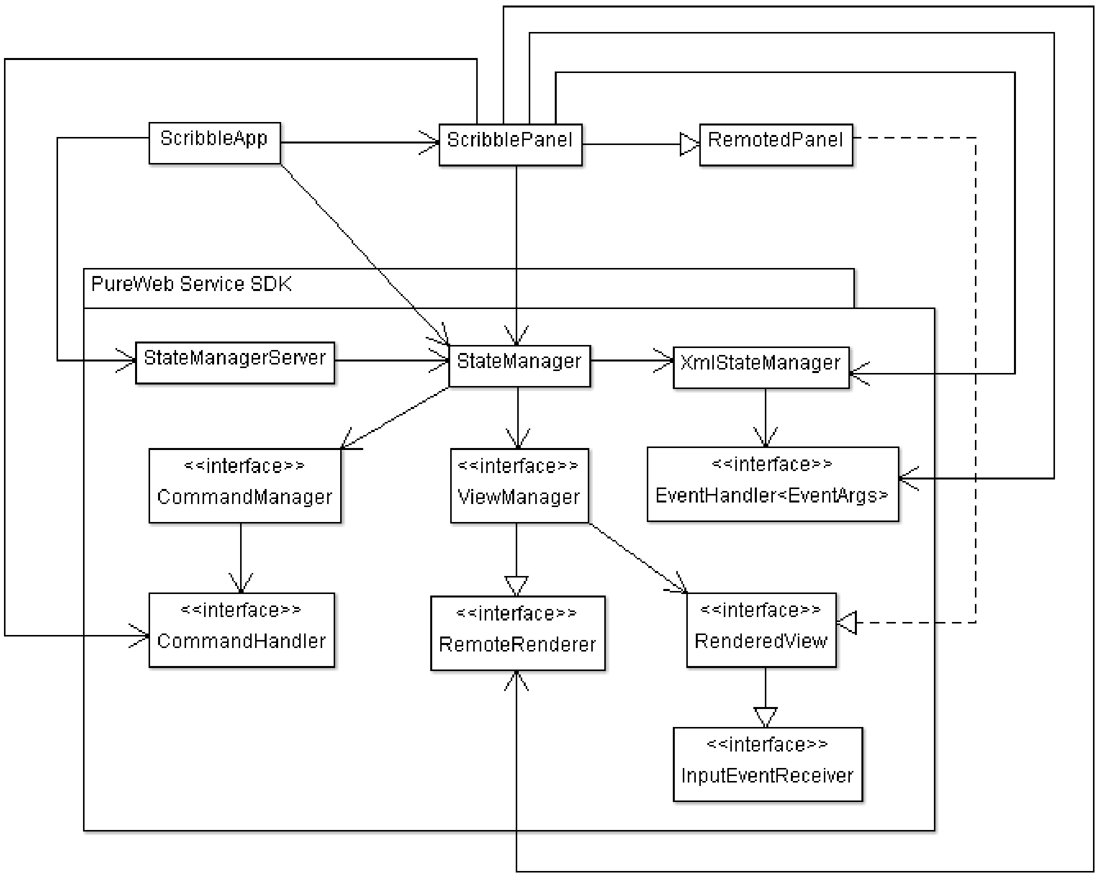
The Asteroids game is keyboard-driven, and there needs to be some mechanism to respond to key input events.

`AsteroidsPanel` implements the `InputEventReceiver` interface which includes a `postKeyEvent` method to respond to key input events. Each key input event is delivered as an instance of `PureWebKeyboardEventArgs`. The key event instance contains all the required information about the event, including which key was pressed or released, along with the session ID of the player that generated the event. The latter is clearly important to provide independent control of the ships in two-player mode.

Service-Side Classes

The diagram below shows the classes comprising the Scribble service, and the key associations with fundamental classes and interfaces in the PureWeb service STK.

For Asteroids, simply replace `ScribbleApp` and `ScribblePanel` with `AsteroidsApp` and `AsteroidsPanel`.


ScribbleApp
ScribblePanel
RemotedPanel
PureWeb Service SDK
StateManagerServer
StateManager
XmlStateManager
<<interface>> CommandManager
<<interface>> ViewManager
<<interface>> EventHandler<EventArgs>
<<interface>> CommandHandler
<<interface>> RemoteRenderer
<<interface>> RenderedView
<<interface>> InputEventReceiver

Chapter

5 Debugging

In the sample Java service application, logging information is written to the PureWeb server console. While this is often helpful in diagnosing a problem, it is sometimes necessary to debug the application interactively to track down the root cause.

Using the sample application as models, this chapter describes how you would debug a PureWeb Java service interactively using Eclipse. It contains the following procedures:

- Creating a project
- Enabling remote debugging
- Performing remote debugging

Other sources of debugging information include the Diagnostics Panel section of the Debugging chapter in the *Developer's Guide*, as well as the Debugging chapter in each of the Quick Start Guides for client development.

Creating a Project

1. Run Eclipse and select the **File | New | JavaProject** menu option.
2. Enter a Project name (`ScribbleApp` for Scribble, `AsteroidsApp` for Asteroids).
3. Uncheck the **Use default location** checkbox.
4. Click on the Browse button and navigate to `[Installed_directory]\SDK\Samples\Java`
5. Click the **Next** button.
6. Change the default output folder to avoid conflicts with the Ant build output folder.
   - For Scribble: `ScribbleApp/bin`
   - For Asteroids: `AsteroidsApp/bin`
7. Click on the **Libraries** tab.
8. Select `CSI.PureWeb.Client.Samples.jar` and `CSI.PureWeb.Server.Samples.jar`.
9. Click on the **Remove** button to remove these two .jar files.
10. Click on the **Add External Jars** button.

11. Navigate to `[Installed_directory]\SDK\Redistributable\Libs\Java`.
12. Select the following .jar files then click the **Open** button:
    - `apache-mime4j-0.6.jar`
    - `commons-codec-1.3.jar`
    - `CSI.PureWeb.Client.jar`
    - `CSI.PureWeb.StateManager.jar`
    - `httpclient-4.0.1.jar`
    - `httpcore-4.0.1.jar`
    - `httpmime-4.0.1.jar`
    - `jcl-over-slf4j-1.5.11.jar`
    - `jdom.jar`
    - `log4j-1.2.15.jar`
    - `slf4j-api-1.5.10.jar`
    - `slf4j-log4j12-1.5.10.jar`
13. Click the **Finish** button to create the project.
14. Click the **No** button if the Setting Build Paths dialog appears.

Enabling Remote Debugging

To enable remote debugging, you edit the deployment descriptor file. There are two methods for editing this file, either through a text editor, or using the configuration page on the PureWeb server.

Both methods involve editing the code shown below.

```
<bean class="pureweb.process.SocketProcessFactory">
  <property name="useFullDuplex" value="true"/>
  <property name="connectionBarrier"
  ref="applicationConnectionBarrier"/>
  <property name="directory" value="@plugin.bin.dir@"/>
  <property name="applicationRegistry" ref="applicationRegistry"/>
  <property name="licenseManager" ref="licenseManager"/>
  <property name="application" value="ScribbleAppJava"/>
  <property name="description" value="ScribbleApp Java"/>
  <property name="executable" value="java"/>
  <property name="options">
    <map>
      <entry key="-cp" value="@plugin.classpath@"/>
    </map>
  </property>
//code continued on next page
```

```
15    <property name="arguments">
16      <list>
17        <!--
18        <value>-Xdebug</value>
19        <value>-Xrunjdwp:transport=dt_socket,address=5005,
          server=y,suspend=y,quiet=y</value>
20        -->
21        <value>-Xms128M</value>
22        <value>-Xmx256M</value>
23        <value>pureweb.samples.scribbleapp.ScribbleApp</value>
24      </list>
25    </property>
26  </bean>
```

Note: The bean's application property value (lines 7, 8 and 23) depends on which sample application you are working with, either `ScribbleAppJava`, or `AsteroidsApp`.

Editing the Deployment Descriptor in a Text Editor

1. Stop the PureWeb server, if necessary, by double-clicking on the **Stop PureWeb** desktop icon.
2. Open the following file in a text editor:
   `[Installed_directory]\SDK\Samples\Java\conf\plugin.xml`
3. Scroll to the bean shown in the sample code at the beginning of this section and uncomment the commented lines of code (lines 17 to 20).
4. Save the plugin.xml file.
5. Rebuild the sample application as follows:
   - **d.** Open a console (command prompt) window.
   - **e.** Change directory to:
     `[Installed_directory]\SDK\Samples\Java`
   - **f.** Type `ant` to run the `build.xml` script.
6. Re-launch the PureWeb server by double-clicking the **Start PureWeb** desktop icon.

Editing the Deployment Descriptor in the Configuration Page

1. Launch the PureWeb server by double-clicking the **Start PureWeb** desktop icon, if not already launched.
2. Open a web browser.
3. Navigate to `http://[server]:[port]/pureweb/config/plugins`.
4. Enter `admin` in both the **Name** and **Password** fields.

5. Click the **Sign In** button to open the Configuration page.
6. Click on the `JavaSamples-plugin.xml` link.
7. Scroll to the bean shown in the sample code at the beginning of this section and uncomment the commented lines of code (lines 17 to 20).
8. Click the **Save** link to return to the Configuration page
9. Click on the **Reload Plugins** link.

Performing Remote Debugging

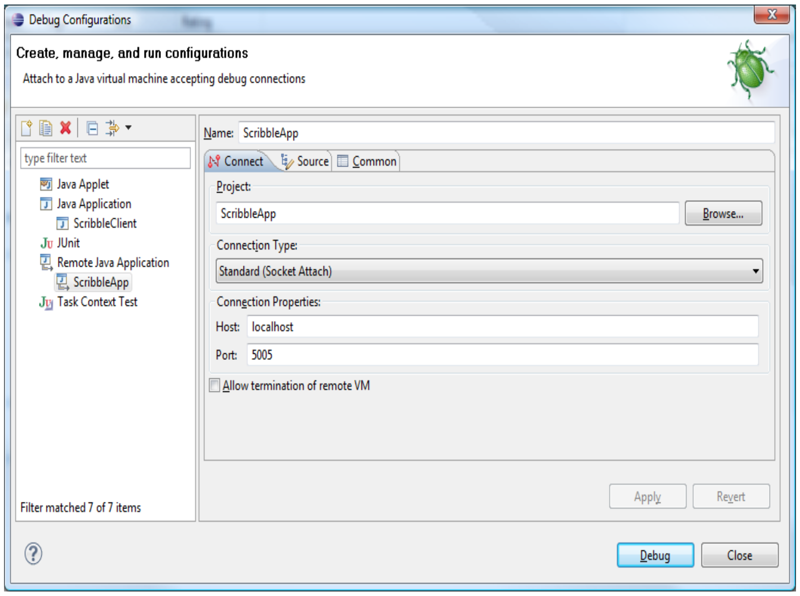
Once you have edited the `plugins.xml` file as indicated above using either a text editor or the server configuration page, you can begin the actual degugging.

1. Launch the PureWeb server, if not already launched, by double-clicking the **Start PureWeb** desktop icon.
2. Open the .java file from the Package Explorer by double-clicking on the filename.
   - For Scribble: `ScribbleClient.java`
   - For Asteroids: `AsteroidsClient.java`
3. Press Ctrl+F11 or select the **Run | Run** menu option to run sample client.
4. Connect to the PureWeb server as follows:
   a. Open a console (command prompt) window.
   b. Change the directory to:
   `[Installed_directory]\SDK\Samples\Java\dist`
   c. Type one of the following commands, depending on which sample application you want to build:
      - For Scribble: `RunScribbleClient.bat`
      - For Asteroids: `RunAsteroidsClient.bat`
   d. Select the **Server | Connect** menu option to display the Connect to Server dialog.
   e. Click the **Connect** button to open the Authorization dialog.
   f. Enter `admin` in both the **Username** and **Password** fields.
   g. Click the **OK** button to start the application.

Note: When you click **Connect** in the Connect to Server dialog, the client will appear to freeze with the Connect to Server dialog still visible. This is expected behavior, and is due to the remote debugging options that specify that the service start in suspended mode.

5. Once again, open the .java file from the Package Explorer by double-clicking on its filename.
   - For Scribble: `ScribbleApp.java`
   - For Asteroids: `AsteroidsApp.java`
6. Select the **Run | Debug Configurations** menu option.

7. Right-click on **Remote Java Application** item in the tree control on the left of the Debug Configurations dialog and select **New**.
8. Enter `5005` in the **Port** text box.



Debug Configurations

Create, manage, and run configurations

Attach to a Java virtual machine accepting debug connections

type filter text

- Java Applet
- Java Application
  - ScribbleClient
- JUnit
- Remote Java Application
  - ScribbleApp
- Task Context Test

Filter matched 7 of 7 items

Name: ScribbleApp

Connect | Source | Common

Project: ScribbleApp [Browse...]

Connection Type: Standard (Socket Attach)

Connection Properties:
Host: localhost
Port: 5005

Allow termination of remote VM

9. Click on the **Apply** button.
10. Click on the **Debug** button. The Connect to Server dialog will be dismissed and the application will start running.
11. Set breakpoints as required to remotely debug the sample service application.
    - For Scribble: `ScribbleApp.java` or `ScribblePanel.java`
    - For Asteroids: `AsteroidsApp.java` or `AsteroidsPanel.java`