PureWeb® STK 4.0

Quick Start Guide: Java Client

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

Although reasonable effort is made to ensure that the information in this document is complete and accurate at the time of release, Calgary Scientific Inc. cannot assume responsibility for any existing errors. Changes and/or corrections to the information contained in this document may be incorporated in future versions.

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_JavaClient_QSG_07-2013_v1.000.00

Table of Contents

Preface

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

Intended Audience

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

- 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 Client*, focuses on the Java client. The guide provides instructions to install the PureWeb STK, and to build, run and troubleshoot a sample client.

You can choose from two Java sample clients:

- Scribble, which works with the C#, C++, and Java sample services, allows users to draw lines on a canvas, change line color, and erase lines.
- Asteroids, which works with the Java sample service, 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.

Before working with the sample Java client, you will need to build the sample service; instructions for doing so can be found in the relevant *Quick Start Guide* for the service programming language you plan to work with.

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 Swing client API reference material

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 clients.

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; your can download to the 2010 version from the following location: http://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.

To build and run the sample Java Swing clients, you will need:

- Java JDK 1.5 or greater (you will already have JDK 1.7 installed for the PureWeb server)
- Apache Ant 1.8.1
- A Java IDE, such as Eclipse Juno 4.2 (recommended)

Although not strictly required to work with the sample clients, a Java IDE provides the 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 the 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 the `ANT_HOME` variable. 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 that string.
3. Select the option **Edit the System Environment Variables** from the search results. This will display the System Properties dialog box, with the Advanced tab selected.
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 path to `JAVA_HOME`, if missing:
   `%JAVA_HOME%\bin;`
4. In the **Variable value** string, add the path to `ANT_HOME`, if missing:
   `%ANT_HOME%\bin;`
5. Click the **OK** button to save your changes.
6. Click **OK** to close the Environment Variables dialog box.
7. 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.

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.

Note: You must build the service first, as described in the *Quick Start Guide* for the service, before following the procedure below.

If you have already built the Java sample service, you do not need to follow this procedure, as the Java Swing client will have already been built along with the service.

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 default server and port for the PureWeb server is `localhost:8080`. Replace as appropriate if this is not correct for your implementation of the server.

Launching the Applications

Launching a Java client 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 client.

1. If the PureWeb server is not running, launch it by double-clicking the **Start PureWeb** desktop icon.
2. Open a browser.
3. Navigate to `http://localhost:8080/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 the **Reload Plugins** button, under the **cluster** 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 either of the following commands:
   - For Scribble: `RunScribbleClient.bat`
   - For Asteroids: `RunAsteroidsClient.bat`

Logging In

1. Select the **Server | Connect** menu option to display the Connect to Server dialog.
2. Type one of the following in the **Server URL** field, based on which service and client applications you are running:
   - Java service and Java Scribble client
     `http://localhost:8080/pureweb/app?name=ScribbleAppJava`
   - C++ service and Java Scribble client:
     `http://localhost:8080/pureweb/app?name=ScribbleAppCpp`
   - C# service and Java Scribble client
     `http://localhost:8080/pureweb/app?name=ScribbleApp`
   - Java service and Java Asteroids client
     `http://localhost:8080/pureweb/app?name=AsteroidsApp`
3. If using Asteroids, enter your name in the **Player name** dialog, or leave blank to play as Anonymous.
4. Click the **Connect** button to open the Authorization dialog.
5. Enter `admin` in both the **Username** and **Password** fields.
6. 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

Scribble is a simple canvas that users can use to draw.

When you first launch the application, it displays a blank canvas. 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 Diagnostics Panel section below for more information.
- Disconnect using the menu (**Server | Disconnect**) or by closing the client.

Notice 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

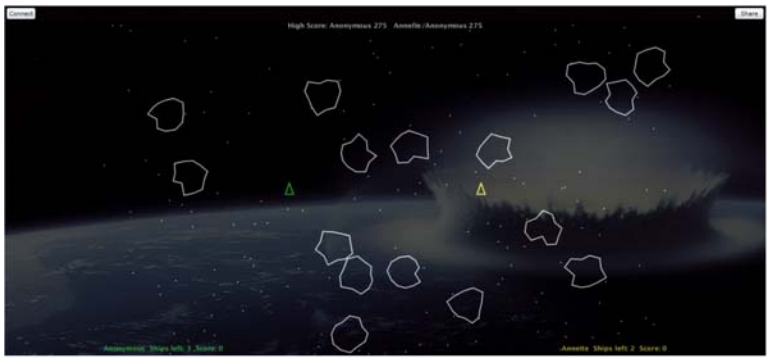
The Diagnostics Panel debugging tool is described in detail in the Debugging chapter of the *PureWeb STK Developer Guide*, however below is a brief overview of the functionality it offers.

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

- The **Options** tab is used to define options that impact image bandwidth and quality.
- The **Trace** tab is used to examine the messages to and from the server. The controls allow you to set the behavior of the trace view, including the buffer length, scrolling, and clearing of the buffer.
- 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.



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 an 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:

- By default 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:

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

Second user:

1. Launch the client by running 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`.

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 clients; 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*.

A graphic illustration the client-side classes discussed in this chapter can be found on page 19 (Scribble) and page 20 (Asteroids).

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

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

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

`...\SDK\Samples\Java\src\client\pureweb\samples\asteroidsapp`

Scribble Client Code Pointers

The `ScribbleClient.java` file contains the `ScribbleClient` class, which implements the scribble client application.

`ScribbleClient` extends `SwingApp`, which ensures the application is initialized to use the Swing event dispatcher. It uses an instance of the `View` class to display and manipulate scribbles. The `View` class provides the functionality required to send keyboard and mouse events to the service application, it also handles the rendering of image updates from the service application. `ScribbleClient` does not have to do anything special to capitalize on this functionality other than use an instance of `View`.

From a PureWeb enablement perspective there are some actions `ScribbleClient` implements to connect to the service application, and to transmit color changes through application state.

run Method

The `run` method of the `ScribbleClient` class adds connected and stalled-changed handlers to the PureWeb framework instance. It includes a value change handler used to listen to color changes made by collaborators.

ConnectedChangedHandler

This inner class implements the `connected` handler. The PureWeb framework notifies the handler when a connection to the PureWeb server has been successfully established using the **Server | Connect** menu option.

`ConnectedChangedHandler` sets the enabled/disabled state of various UI elements (the Share button for example).

StalledChangedHandler

This inner class implements the `stalled-changed` handler. The PureWeb framework calls this handler whenever it detects that the connection to the PureWeb server may have been lost. The handler displays a message dialog informing the user of the possible lost connection.

EraseListener

This event listener listens for buttons clicks on the **Erase All** button. When the button is clicked, it sends a `Clear` command to the service application. The service responds by clearing previously drawn scribbles and sending an updated blank image back to the client.

ColorChangedListener

This event listener listens for focus lost events on the **color** text field and action events. It also track typing changes in the color text field and automatically set the color when a full color name has been entered.

When the **color** text field loses focus or the **Enter** key is pressed, the `ScribbleColor` application state property is set to the current color. If the current color is different from the previous color, PureWeb generates an application state difference and transmits the difference to the service application. The service merges the difference into its application state, causing a color change on the server-side, which is propagated back to the client in the form of an updated image.

ShareListener

This event listener listens for buttons clicks on the **Share** button. When the button is clicked, an `AppShare` service request is created and queued for execution by the PureWeb server. When this service request has completed, the `EventHandler<ServiceRequestCompletedEventArgs>` instance supplied at construction time is notified with the result. The handler creates a new `DisplayShareUrl` instance to display the share URL (or an error message if execution failed for some reason) on the UI thread. The share URL can be sent to other users to connect to the Scribble service application of the current user.

DisconnectListener

This event listener listens for selection of the **Server | Disconnect** menu option. When this menu option is selected, `DisconnectListener` calls the `disconnect` method to disconnect from the Scribble service. The application state is cleared to prevent any residual state information from causing issues in the event a new connection is established.

Asteroids Client Code Pointers

The files described in this section can be found in the PureWeb directory:
`...\SDK\Samples\Java\src\client\pureweb\samples\asteroidsapp`

This `AsteroidsClient.java` file contains the `AsteroidsClient` class, which implements the Asteroids client application. `AsteroidsClient` extends `SwingApp`, which ensures the application is initialized to use the Swing event dispatcher.

`AsteroidsClient` uses an instance of the `View` class to display the Asteroids game. In addition, the `View` class provides the functionality required to send keyboard events to the service application; `AsteroidsClient` does not have to do anything special to capitalize on this functionality other than use an instance of View.

ConnectedChangedHandler

This inner class implements the `connected` handler. The PureWeb framework notifies the handler when a connection to the server has been successfully established using the **Server | Connect** menu option.

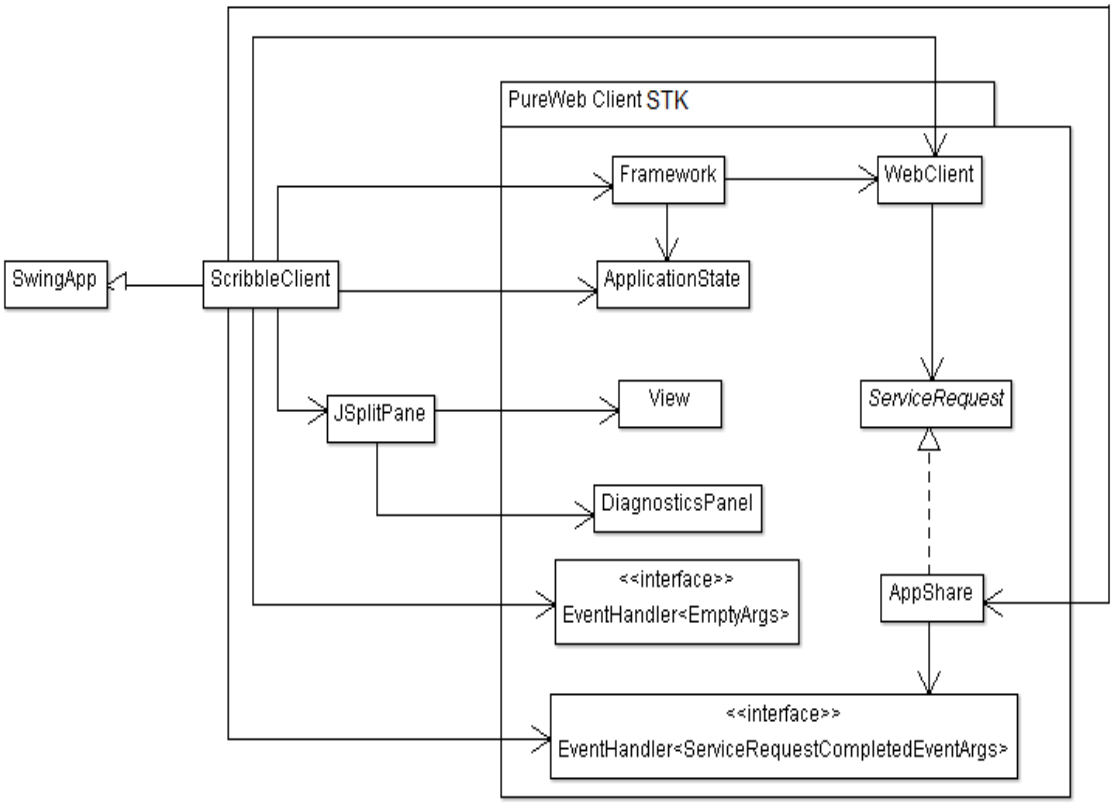
Note that the player's name is sent to the service as a parameter of the `ConnectSession` command. An initial resize is also sent to the service by calling the resize method of `asteroidsView`. The resize method informs the Asteroids service of the size of the client view so that any necessary scaling can be performed before updated images are sent.

ShareListener

This event listener listens for buttons clicks on the **Share** button. When the button is clicked, an `AppShare` service request is created and queued for execution by the PureWeb server. When this service request has completed, the `EventHandler<ServiceRequestCompletedEventArgs>` instance supplied at construction time is notified with the result. The handler creates a new `DisplayShareUrl` instance to display the share URL (or an error message if execution failed for some reason) on the UI thread. The share URL can be sent to the second player. If the second player connects using the share URL, Asteroids will automatically go into two-player mode.

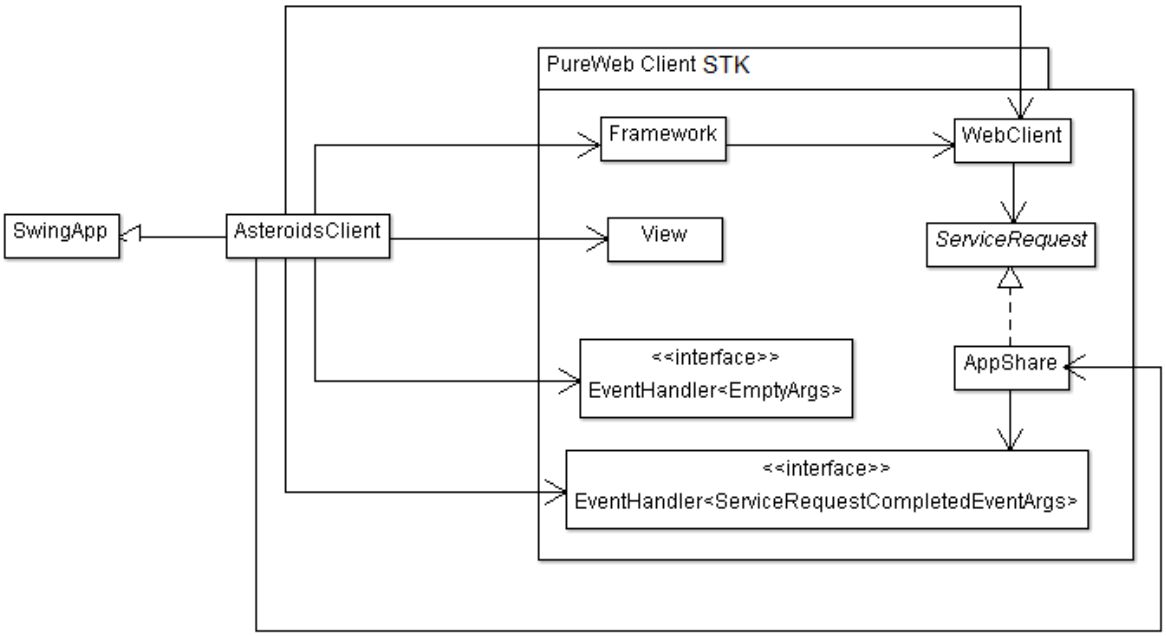
Scribble Client-Side Classes

The diagram below shows the classes comprising the Scribble client.



Asteroids Client-Side Classes

The diagram below shows the classes comprising the Asteroids client. This diagram is very similar to the Scribble one, but doesn't use the `JSplitPane` or the `DiagnosticsPanel`.



Chapter

5 Debugging

The console window running the sample java client application provides extensive debugging information. 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. This is a two-step process:

- Creating a projet in Eclipse
- Setting breakpoints

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 service development.

Creating a Project in Eclipse

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.

Click the **No** button if the Setting Build Paths dialog appears.

Setting Breakpoints

1. Run Eclipse.
2. Click on the .java file in Package Explorer to open the file (`ScribbleClient.java` or `AsteroidsClient.java`).
3. Press F11 to launch sample client in the debugger.
4. Set breakpoints as required in the .java file.