AJAX for Java Web Applications

Session: 5

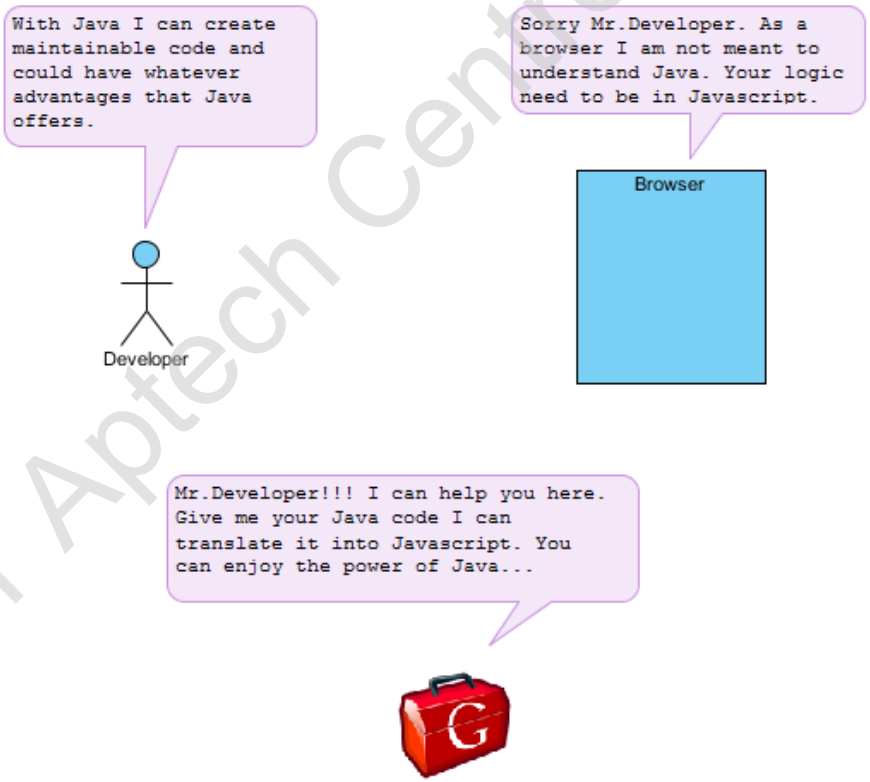
Google Web Toolkit

Objectives

- Describe Google Web Toolkit
- Work with Google Maps
- Describe Google AJAX search API

Google Web Toolkit 1-2

- Google Web Toolkit (GWT), pronounced as ‘gwit’, is an open source development kit provided by Google.
- It allows Java developers to develop rich applications without much expertise on client-side technologies such as JavaScript, CSS, and so on.
- In GWT, everything appears as Java objects.
- So ‘How can Java replace JavaScript?’. The answer is the GWT compiler as shown in the following figure:



Google Web Toolkit 2-2

- Anything that is written in Java, is translated by the GWT compiler into JavaScript/CSS/HTML as applicable.
- The application written in GWT is cross-browser compliant.
- GWT automatically generates JavaScript code suitable for each browser.
- GWT is completely free and licensed under the Apache License version 2.0.
- GWT is a framework for developing large scale, high performance, and easy-to-maintain Web applications.

Advantages and Disadvantages of GWT

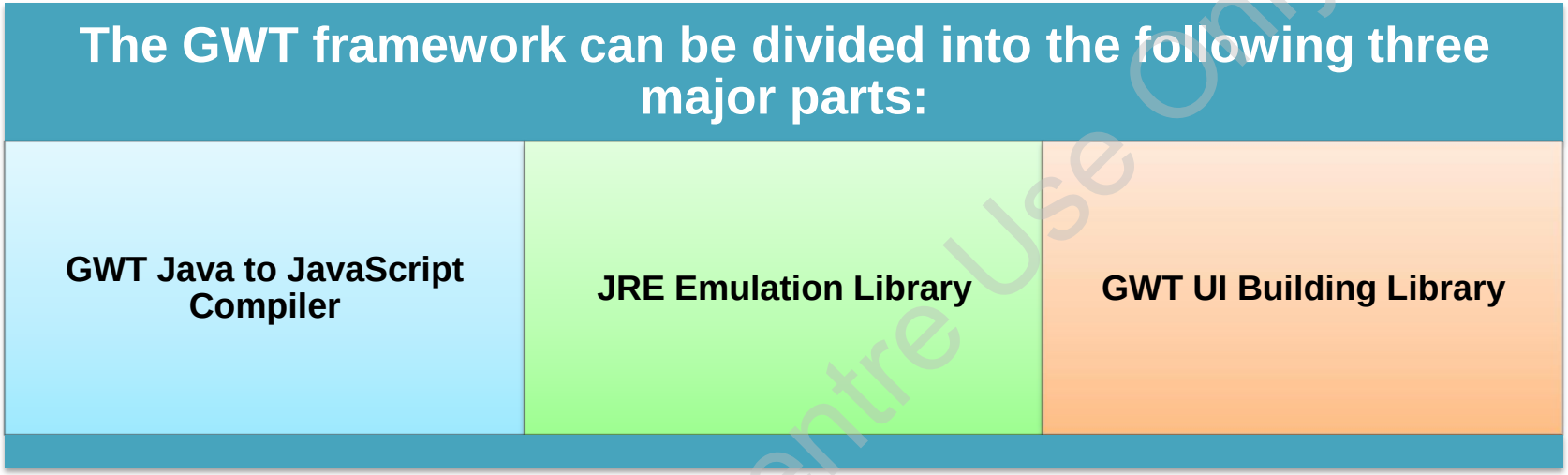
Advantages

- GWT is Java based hence, one can use Java IDEs such as Eclipse, NetBeans, and so on for developing a GWT application.
- GWT has a low learning curve for Java developers.
- GWT also provides debugging capability so that the client-side application can be debugged just as a Java application.
- GWT provides easy integration with JUnit and Maven.
- GWT generates browser-specific and optimized JavaScript code.
- GWT provides a Widgets library that helps to implement most of the functionality required in an application.
- GWT is extensible and allows creation of custom widgets to cater to application-specific requirements.

Disadvantages

- Web pages generated by GWT cannot be indexed by search engines because these applications are generated dynamically.
- If JavaScript is disabled in the browser then, user will only be able to view the basic page and nothing else.
- GWT is not suitable for Web designers who prefer using plain HTML with placeholders for inserting dynamic content at later point in time.

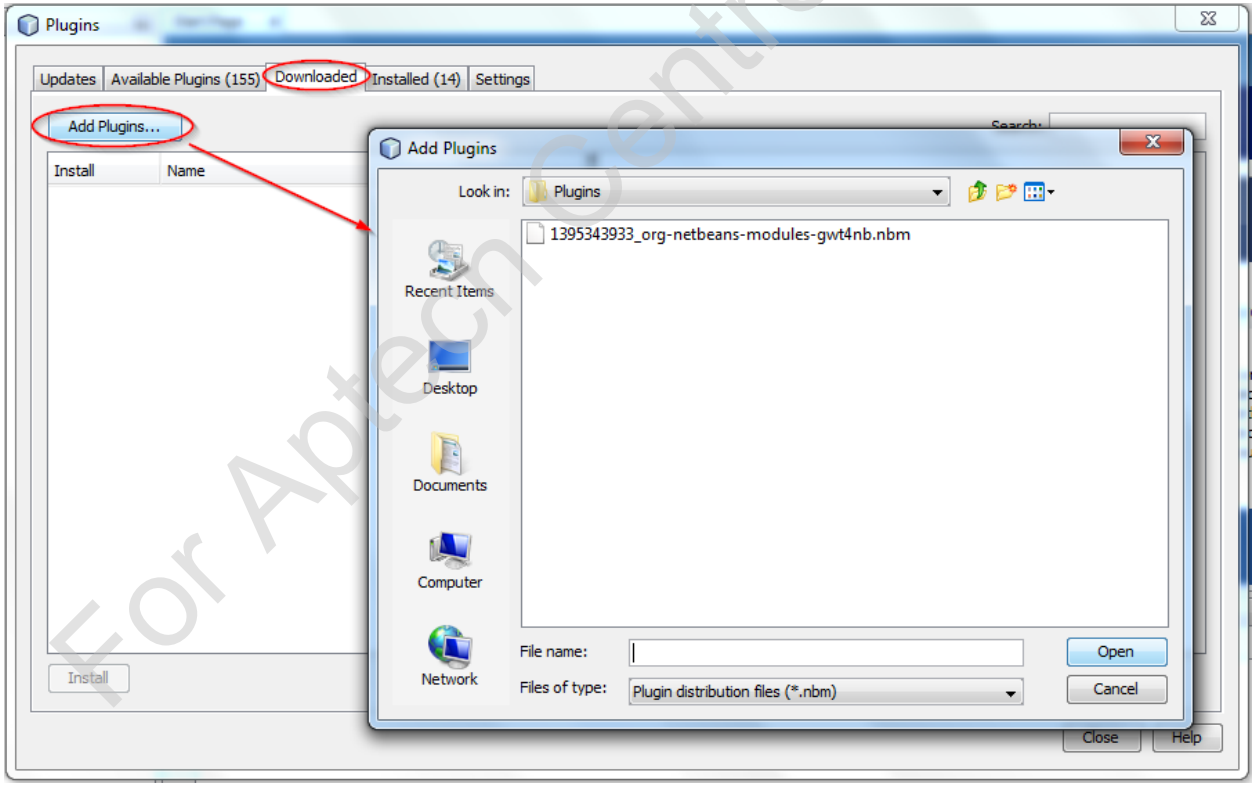
GWT Components



- Additionally, GWT provides a GWT Hosted Web Browser that allows executing the GWT applications in hosted mode.
- Here, the code runs as Java code in the JVM without compiling to JavaScript.
- A GWT application consists of the following four important parts:
  - Module descriptors
  - Public resources
  - Client-side code
  - Server-side code
- The last component is optional but first three components are mandatory.

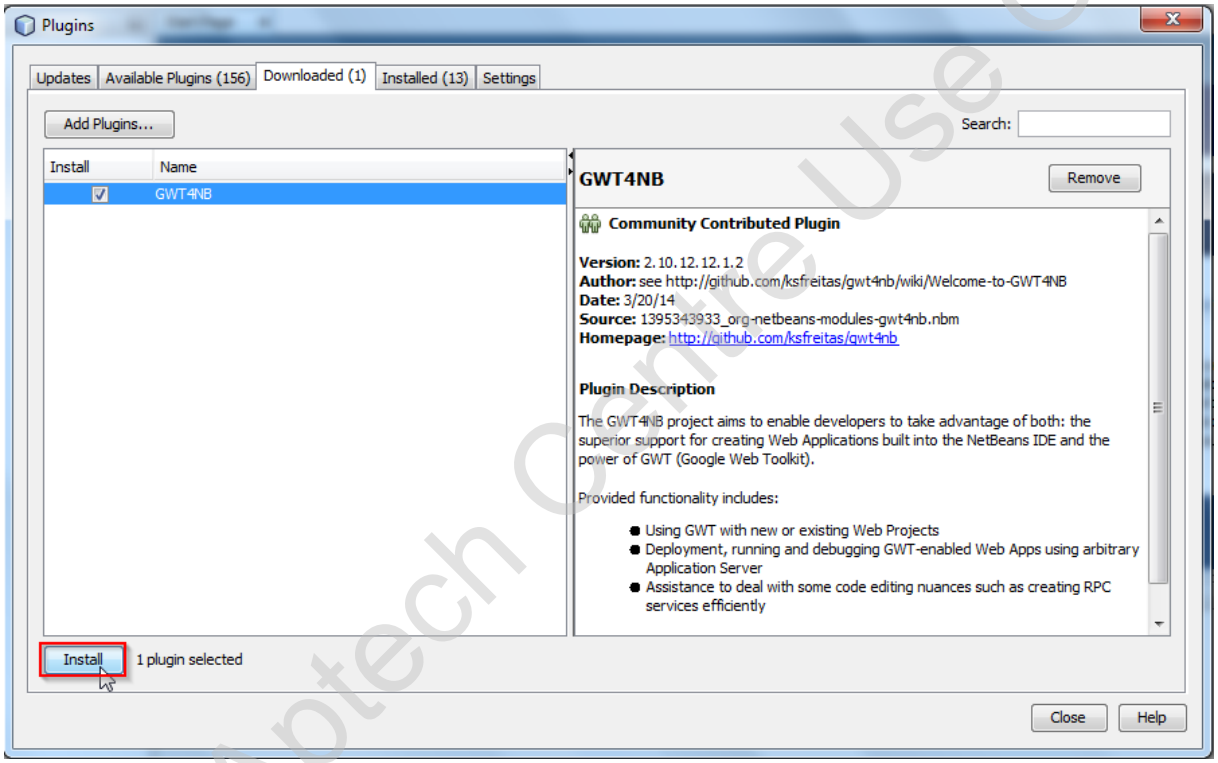
Steps to Install the Google Web Toolkit Plugin 1-2

1. • Download the latest plugin from http://plugins.netbeans.org/plugin/44509/gwt4nb.
2. • Go to Tools → Plugins.
3. • Click the Downloaded tab and then click Add Plugins.
4. • Select the plugin file that has just been downloaded, as shown in the following figure:



Steps to Install the Google Web Toolkit Plugin 2-2

5. • Once the plugin is added, click Install to install it as shown in the following figure:

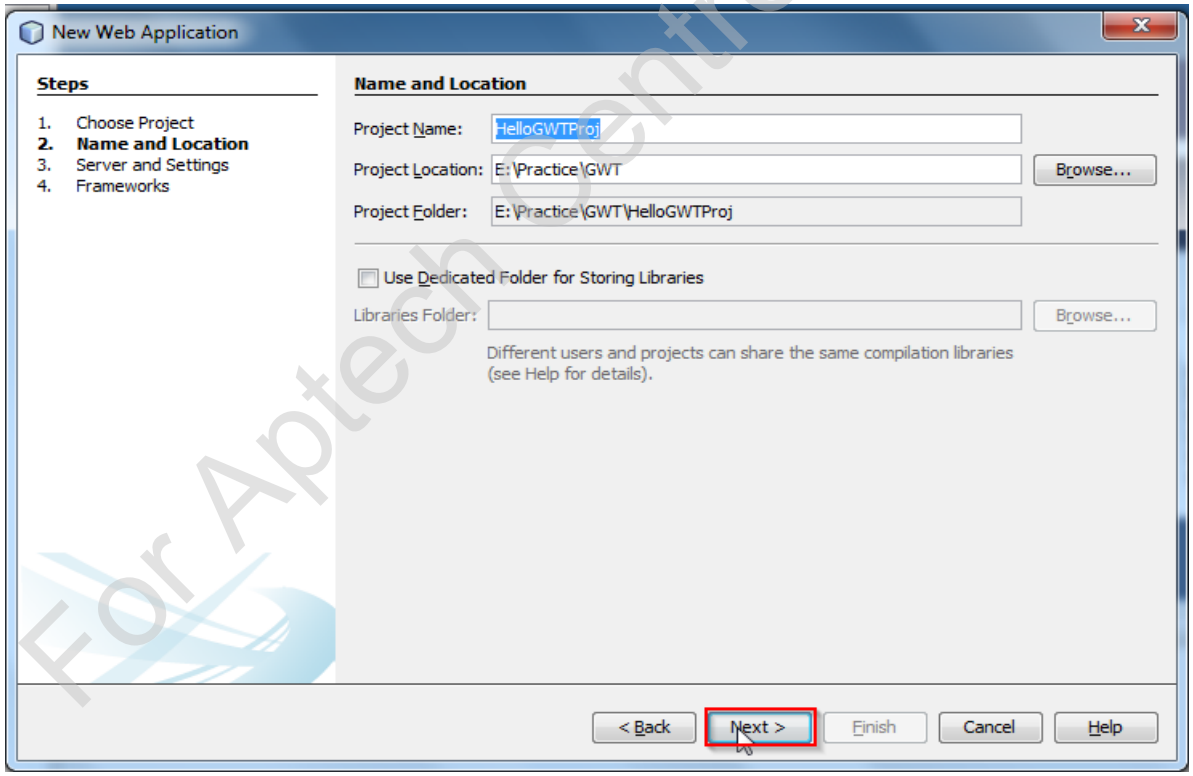


6. • Download and unzip the GWT SDK, gwt-2.7.0 from http://www.gwtproject.org/download.html?csw=1.

- This contains the core libraries, compiler, and development server that you need to write Web applications.

Steps to Create a Google Web Toolkit Project 1-3

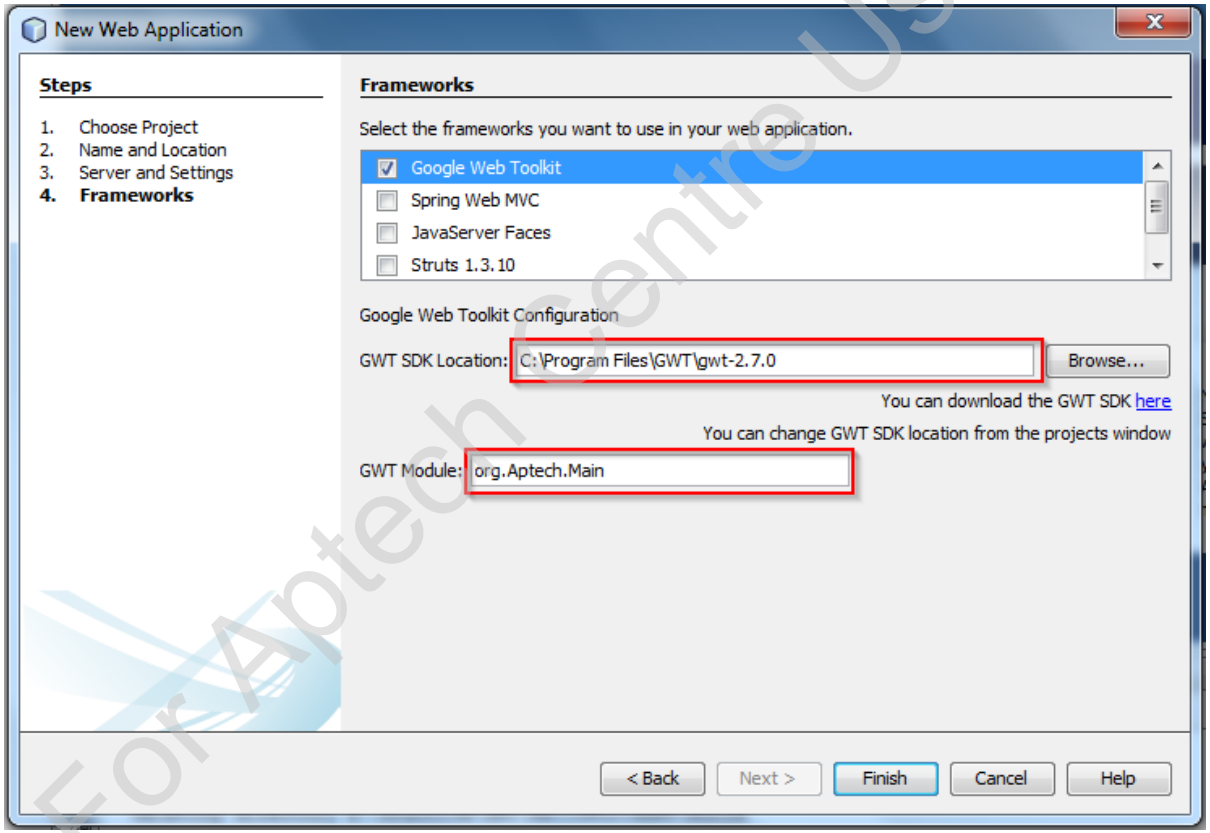
1. • Create a Java Web Application project by clicking File → New Project.
2. • Select Java Web → Web Application, and then click Next.
3. • Specify the Project Name as shown in the following figure and click Next.



Steps to Create a Google Web Toolkit Project 2-3

4. • Select the application server GlassFish Server 4.1.

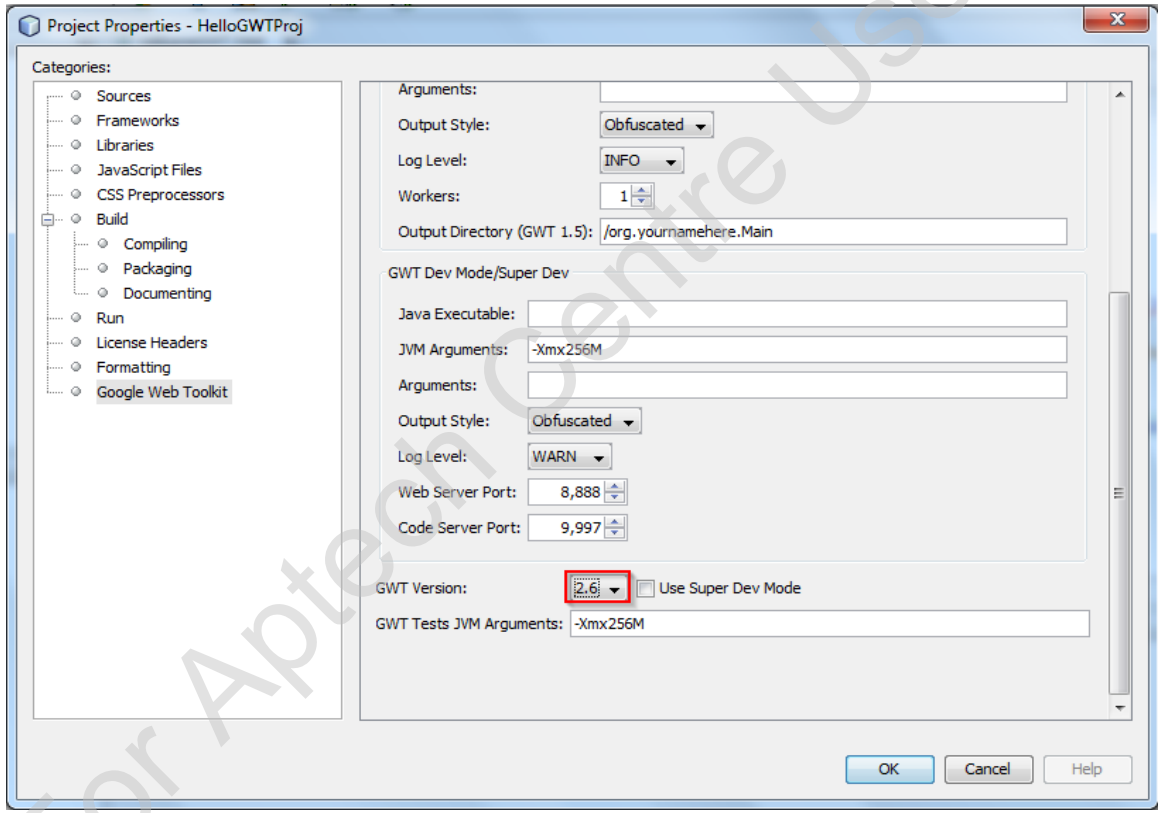
5. • Select the latest GWT version that has been installed from the Frameworks list, set the location of GWT SDK, and specify the GWT Module name as shown in the following figure. Click Finish.



Steps to Create a Google Web Toolkit Project 3-3

6. • Right-click the project and open the project properties.

7. • Change the GWT version 2.7 to version 2.6 as shown in the following figure:



8. • Click OK.

GWT Basics 1-2

- To work with GWT, first it is required to understand the following building blocks of GWT and GWT terminologies:

EntryPoint

- The EntryPoint interface belongs to the com.google.gwt.core.client package.
- It must be implemented by the class that represents the entry point of a module.
- Any GWT application must have atleast one class that implements com.google.gwt.core.client.EntryPoint.
- A class that implements the EntryPoint interface to a GWT application is similar to the main() method of a Java program.
- The only difference is that there can be multiple classes that implement the EntryPoint.
- The logic starts with the onModuleLoad() method of the class that implements the EntryPoint interface.
- Ensure that the EntryPoint class contains a default constructor.

Module

- A module is an entity that contains one or more entry points.
- A GWT application can be considered as a collection of one or more modules.

Module Descriptor

- For every module, there is an .xml file called Module Descriptor.
- For example, for a module with a name, ModuleName, the Module Descriptor will be in the format of ModuleName.gwt.xml.

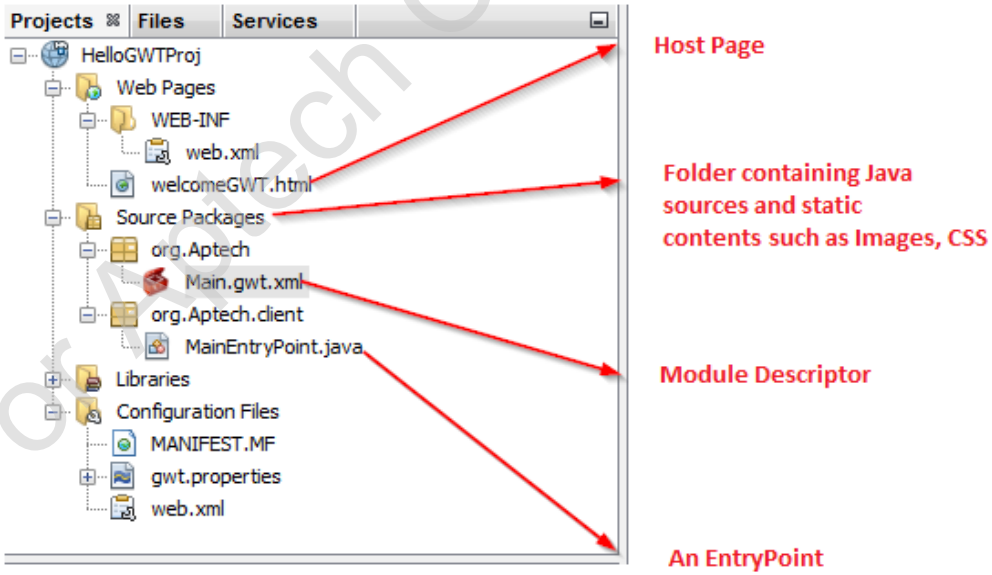
GWT Basics 2-2

Host Page

- The logic from the EntryPoint classes is converted into JavaScript files.
- Hence, there must be some HTML or other server pages that include the scripts and hook-up the GWT functionalities into the application.
- These pages are referred as Host Pages as shown in the given figure.

Bootstrap file

- This refers to the JavaScript file that has been generated as an equivalent for the GWT module.
- This is where all the functionality written in GWT is stored.
- This will be located with the relative path as follows:
  - `<script type="text/JavaScript" src="org.Aptech.Main/org.Aptech.Main.nocache.js"></script>`



CSS Styling APIs

- Some important APIs that are helpful in Web programming using GWT are as follows:

public void setStyleName(java.lang.String style)

public void addStyleName(java.lang.String style)

public void removeStyleName(java.lang.String style)

public java.lang.String getStyleName()

public void setStylePrimaryName(java.lang.String style)

GWT Widgets

- GWT Widgets are UI components which can be used as per requirement on the Web page.
- In GWT, a simple Button, Text box, a Date Picker, or a Rich Text Area, everything is a Widget. Widgets are added to the panels which handles user interaction.
- List of few widgets that are offered by GWT by default are is follows:
  - com.google.gwt.user.client.ui.Button
  - com.google.gwt.user.client.ui.TextBox
  - com.google.gwt.user.client.ui.Tree
  - com.google.gwt.user.client.ui.RichTextArea
  - com.google.gwt.user.client.ui.PushButton
  - com.google.gwt.user.client.ui.RadioButton
  - com.google.gwt.user.client.ui.CheckBox
  - com.google.gwt.user.datepicker.client.DatePicker
  - com.google.gwt.user.client.ui.PasswordTextBox
  - com.google.gwt.user.client.ui.TextArea
  - com.google.gwt.user.client.ui.ToggleButton
- To use the Widgets, perform the following steps:

1. Create an instance of the Widgets and change the characteristics of the widget as required.
2. Add the widgets to the panel(s), so that they can be positioned in the proper location of the document.

GWT Event Handling

- GWT is basically event driven. That is, the action mechanism will be based on the event that occurs on the client-side.
- GWT Events are similar to the events in other Java UI frameworks such as AWT or Swing. A UI component informs the other instances through events.
- Following are the building blocks of event handling:

The UI component in which the event occurs. For example, Button.

The Implementation class, whose instance needs to be notified by the component when some event occurs. This can be any class that handles the Event.

A handler interface, which must be implemented by the Class that will handle the event. For example, ClickHandler.

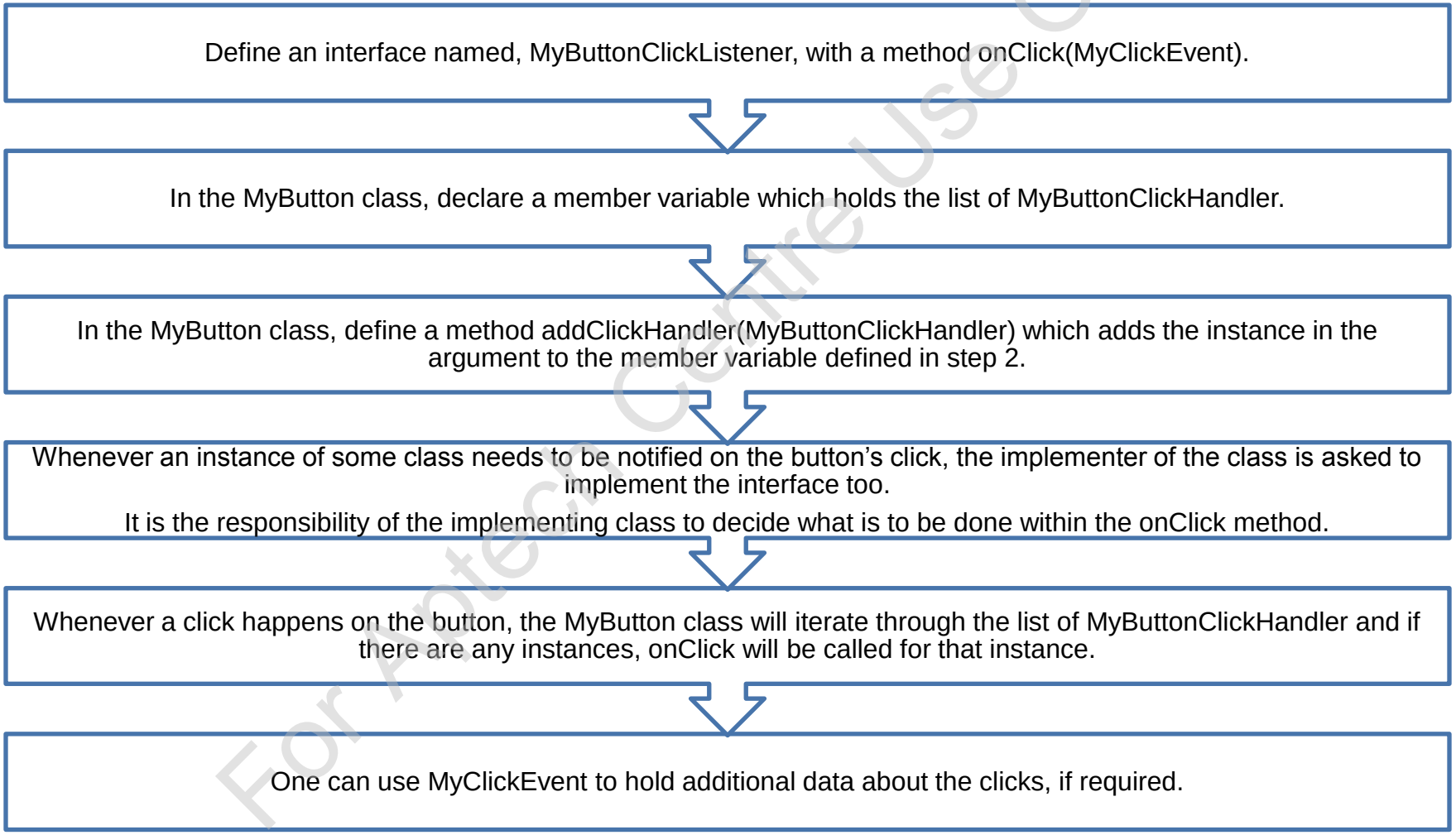
The event instance, which will be passed by the component. The required details can be retrieved from that instance. For example, ClickEvent.

Observer/Listener Design Pattern 1-2

- Listener design pattern is a solution for any scenario in which an instance needs to inform one or many instances on some events.
- For example, when a button is clicked, some dialog box needs to be displayed and there exists an instance that validates something.
- Consider that there are three different entities performing all these tasks as follows:
  - The button instance, on which the click happens
  - The instance which will show the dialog box
  - The instance which does the validation

Observer/Listener Design Pattern 2-2

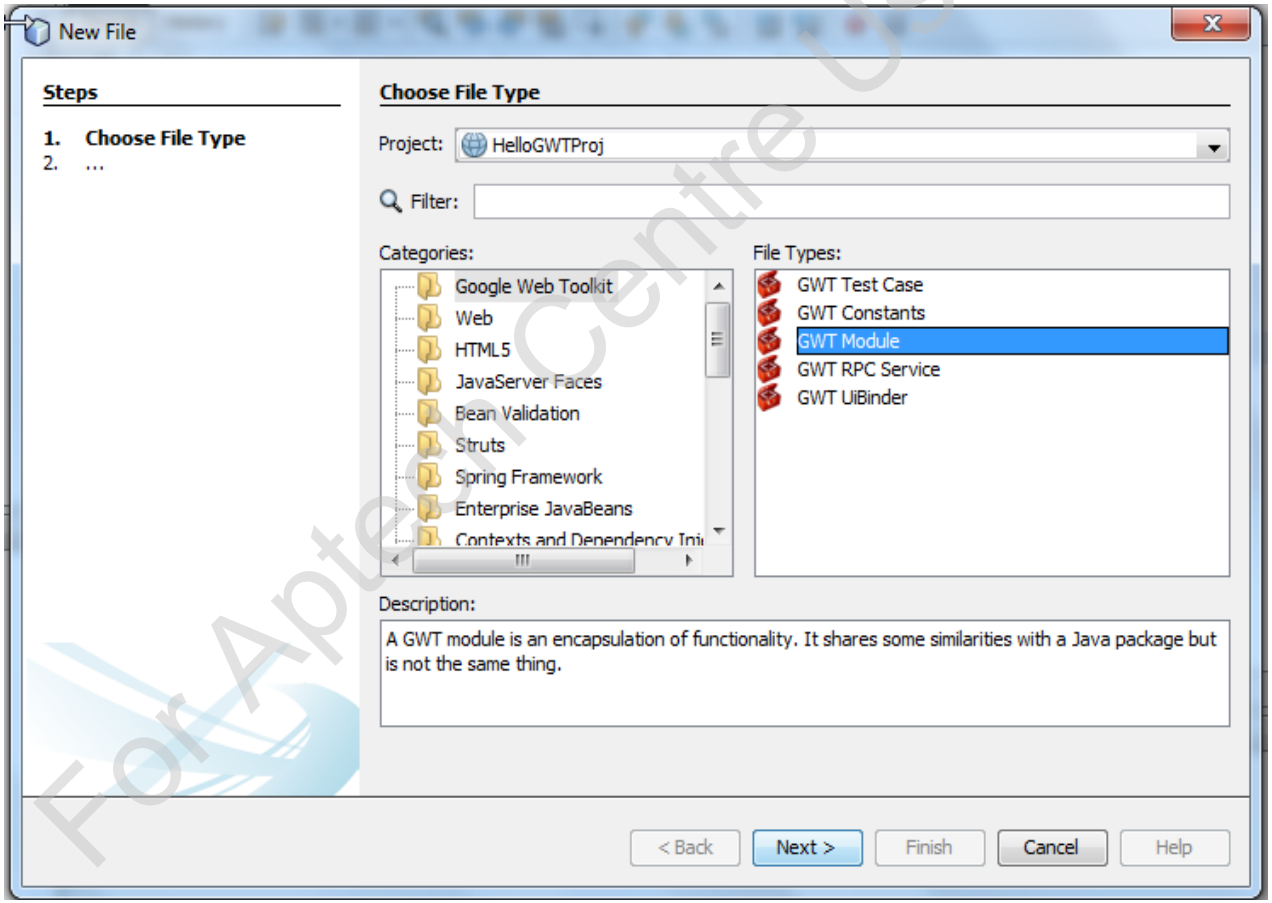
- The steps to apply Observer/Listener design pattern are as follows:



Steps for Creating a Module 1-2

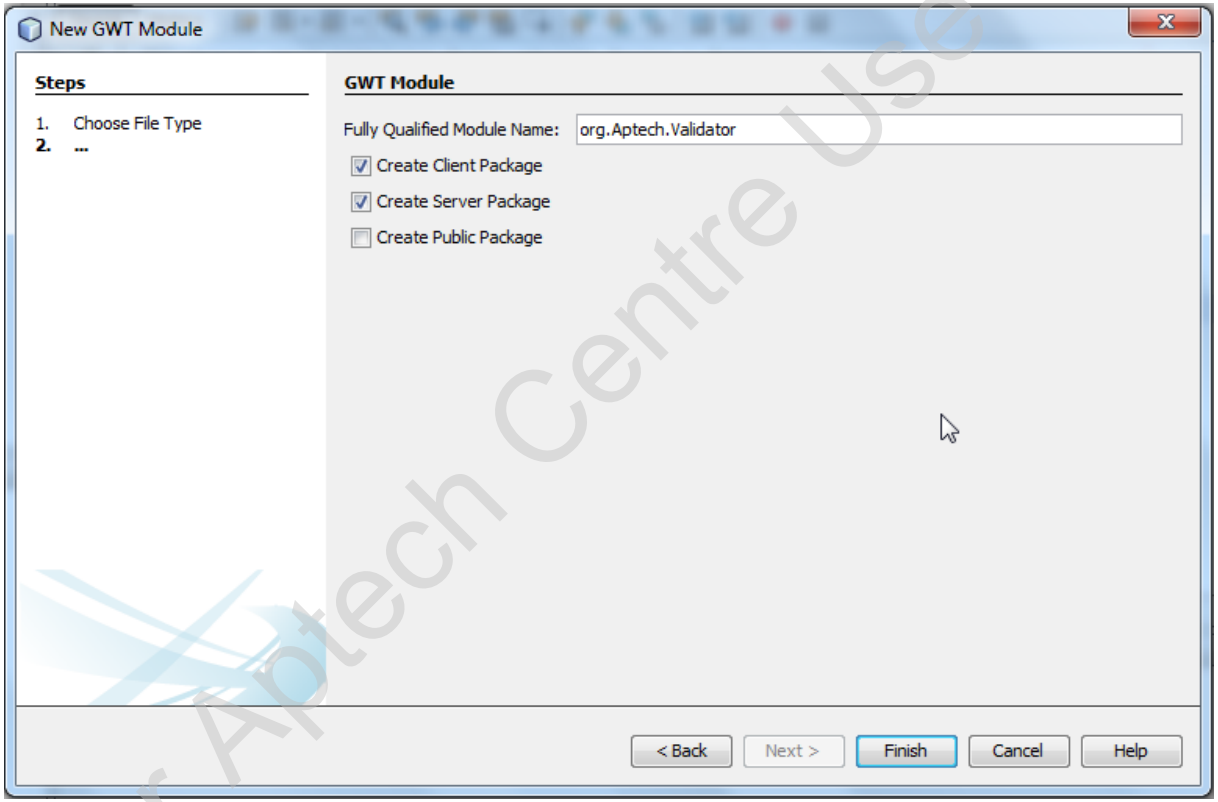
1. • Right-click the project and select New → Other.

2. • In the dialog box that appears, select the GWT module, as shown in the following figure and click Next.



Steps for Creating a Module 2-2

3. • In the next screen that appears, specify the Module name and click Finish as shown in the following figure:

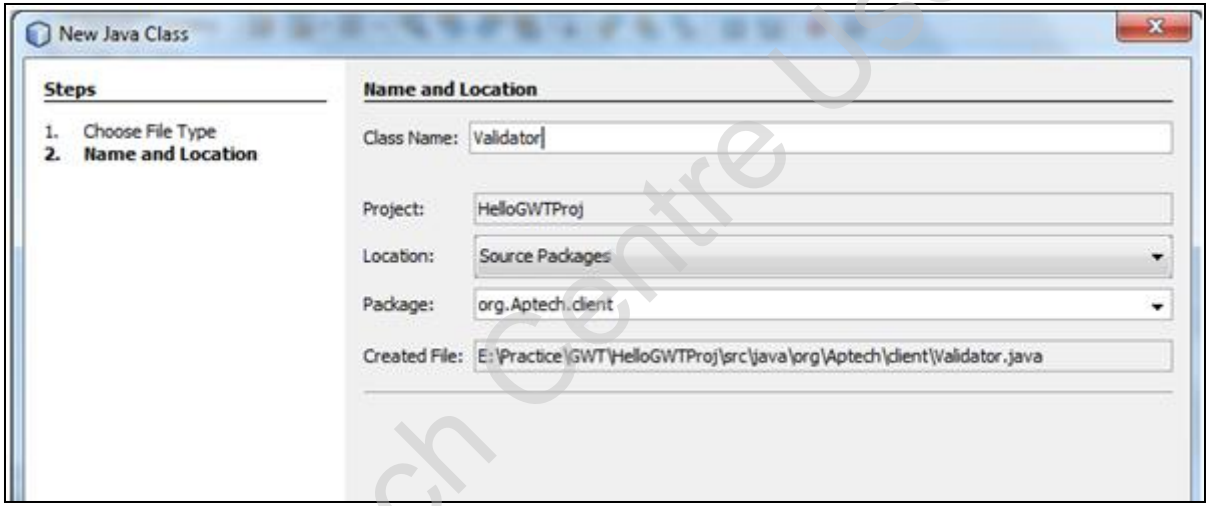


- The module descriptor gets created in the source package.

Steps for Creating an Entry Point

1. • Right-click the client package and click New → Java Class.

2. • Specify the Class Name as shown in the following figure and click Finish.



3. • Implement the EntryPoint interface in the Validator class and override the onModuleLoad() method as demonstrated in the following Code Snippet:

```
public class Validator implements EntryPoint{
    @Override
    public void onModuleLoad()    {
        // TODO: The implementation starts here.
    }
}
```

Steps to Add Entry Point to a Module

- Add the entry for the EntryPoint class in the module descriptor as demonstrated in the following Code Snippet:

```
<!-- Specify the app entry point class.  -->
 <entry-point class="org.Aptech.client.Validator"/>
```

- Note, in this case, the entry point class is org.Aptech.client.Validator.
- **Steps to Hook-up a Module in a Host Page**
  - To get the logic written in the Module to work, a bootstrap file must be included in the page in which the module needs to be added.
  - The bootstrap file is present in the following location: <module_name>/<module_name>.nocache.js
  - For example, if the GWT module org.Aptech.client.Validator functionality needs to be added in some HTML file, a line must be added to the HTML page as shown in the following Code Snippet:

```
<script type="text/JavaScript"
src="org.Aptech.validator/org.Aptech.Validator.nocache.js">
</script>
```

GWT RPC

- GWT Remote Procedure Calls (RPC) is a mechanism with which GWT allows to call the server method from the client code, asynchronously.
- From a developer perspective,
  - A servlet is created (by extending RemoteServiceServlet). This servlet will contain the method that will be invoked through RPC.
  - A client-side interface will be created (by extending RemoteService) which will be responsible for invoking the method in server.
  - The Java POJOs will be created for carrying data in this communication.
- The serialization of the POJO instances in this communication will be handled by GWT.
- The sequence of events is as follows:
  - The server-side code remains as it is.
  - The client-side code will be translated into equivalent JavaScript. This JavaScript will be responsible for making calls to the server-side servlet. This call is an AJAX call.
  - GWT takes the responsibility of making appropriate translations in this communication.

Widget Example 1-3

- Following Code Snippet demonstrates an example of creating and adding a simple Toggle button to a Web page:
- **HelloWidget.html** – Present in Web Pages folder of the Project.

```
...
<script type="text/JavaScript"
src="org.Aptech.Widget/org.Aptech.Widget.nocache.js"></script>
    </head>
...
```

- Following Code Snippet demonstrates the module descriptor file Widget.gwt.xml:
- **<Source Package>/org.Aptech/Widget.gwt.xml**

```
...
<module>
...
<-- Specify the app entry point class.  -->
 <entry-point class="org.Aptech.client.MyWidgetDemo"/>
</module>
...
```

Widget Example 2-3

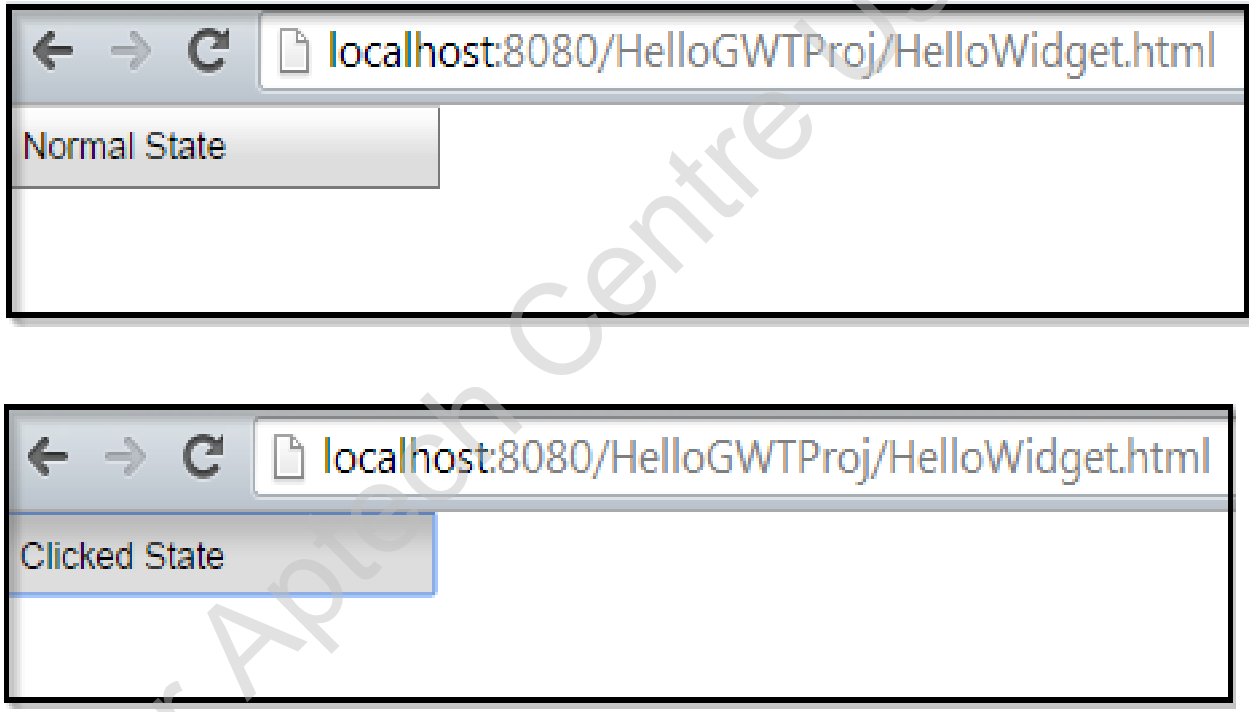
- Following Code Snippet demonstrates the entry point class for the application:
- Entry Point Class – **MyWidgetDemo.java**

```
package org.Aptech.client;
import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.user.client.ui.RootPanel;
import com.google.gwt.user.client.ui.ToggleButton;

public class MyWidgetDemo implements EntryPoint{
    @Override
    public void onModuleLoad()    {
        // Create Instance for your Widget
      ToggleButton aToggleButton = new ToggleButton("Normal State",
"Clicked State");
    // Apply required style as per your wish
        aToggleButton.setPixelSize(150, 20);
        aToggleButton.setTitle("Click to Toggle");
      // Add it to the panel of your wish
      RootPanel.get().add(aToggleButton);
}
}
```

Widget Example 3-3

- The code creates a ToggleButton with values of strings to be toggled every time the button is clicked.
- Following figure shows the output of the code:



Event Handling Example 1-4

In this example, a text box will be used to accept the color from the user.

Once the user enters it, the keyboard event will be handled and the color of the body of the current document will be set based on the value entered by the user.

Following Code Snippet demonstrates the creation of the host page:

```
...
<script type="text/JavaScript"
src="org.Aptech.EventDemo/org.Aptech.EventDemo.nocache.js"></script>
    </head>
    <body>
        <div>
            <p>Enter a valid color :</p>
            <input type="text" id="ipColor" name="ipColor"/>
        </div>
    </body>
...
```

Event Handling Example 2-4

Following Code Snippet demonstrates the module descriptor:

- **EventDemo.gwt.xml**

```
...
<module>
...
<!-- Specify the app entry point class. -->
   <entry-point class="org.Aptech.client.EventHandle"/>
</module>
...
```

Following Code Snippet demonstrates the entry point class for handling the event:

```
org.Aptech.client.EventHandle
package org.Aptech.client;
import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.event.dom.client.KeyUpEvent;
import com.google.gwt.event.dom.client.KeyUpHandler;
import com.google.gwt.user.client.DOM;
import com.google.gwt.user.client.ui.RootPanel;
import com.google.gwt.user.client.ui.TextBox;
```

Event Handling Example 3-4

```
public class EventHandle implements EntryPoint{
    private final static String itsTextId = "ipColor";
    @Override
    public void onModuleLoad(){
        // Get the text box as a Java instance
        final TextBox aTextBox =
TextBox.wrap(DOM.getElementById(itsTextId));

        // Add the KeyUpHandler to the text box
        aTextBox.addKeyUpHandler(
// Anonymous implementation of KeyUpHandler which handles the key event
        new KeyUpHandler(){
            @Override
            public void onKeyUp(KeyUpEvent event){
// Set the background color of the <body> element based on the value
// from the text box

RootPanel.getBodyElement().setAttribute("style","background-
color:"+aTextBox.getValue());
            }
        }
    );
}
}
```

Event Handling Example 4-4

Following figure shows the output when user enters the color as Orchid:

Following figure shows the output when hex color code is specified:

Following figure shows the output when color is specified in RGB format:

RPC Example 1-5

Following Code Snippet demonstrates the creation of the RPCMessage class:

```
...
public class RPCMessage implements Serializable{
    String itsMessage;
    private static final long serialVersionUID = 1L;
    public RPCMessage (){

    }
        public RPCMessage (String theMessage){
        itsMessage = theMessage;
    }
    public String getServerMessage(){
        return itsMessage;
    }
}
```

Following Code Snippet demonstrates the creation of the DemoRPCService class:

```
...
@RemoteServiceRelativePath("rpcdemo")
public interface DemoRPCService extends RemoteService {
    RPCMessage getServerMessage();
}
...
```

- The RemoteService interface declares the getServiceMessage() method to return the message from the server.

RPC Example 2-5

Following Code Snippet demonstrates the creation of the DemoRPCServiceImpl class:

```
...
public class DemoRPCServiceImpl extends RemoteServiceServlet implements
DemoRPCService{

    @Override
    public RPCMessage getServerMessage(){
      RPCMessage aMessage = new RPCMessage("The current date is: "+ new Date());
      return aMessage;
    }
}
```

- The message to be returned from the server is set in the RPCMessage object.

Following Code Snippet demonstrates the creation of the DemoRPCServiceAsync class:

```
package org.Aptech.client.rpc;
import com.google.gwt.user.client.rpc.AsyncCallback;

public interface DemoRPCServiceAsync{
  void getServerMessage(AsyncCallback<RPCMessage> theCallback);
}
```

RPC Example 3-5

Following Code Snippet demonstrates the creation of the RPCMessageCallBack class:

```
...
public class RPCMessageCallBack implements AsyncCallback<RPCMessage>{
    @Override
    public void onFailure(Throwable caught) {
        Window.alert("Error while communicating the server.");
    }
    @Override
    public void onSuccess(RPCMessage result){
        Window.alert("Message from server"+result.getServerMessage());
    }
}
```

Following Code Snippet demonstrates the creation of the RPCEntryPoint class:

```
...
public class RPCEntryPoint implements
EntryPoint{
    private DemoRPCServiceAsync
itsMessageService =
GWT.create(DemoRPCService.class);

    @Override
    public void onModuleLoad()    {
        Button aButton = new
Button("Test RPC!!!");

HorizontalPanel aVPanel = new
HorizontalPanel();
aVPanel.add(aButton);
RootPanel.get().add(aVPanel);
aButton.addClickHandler(new
ClickHandler(){
@Override
public void onClick(ClickEvent event){

itsMessageService.getServerMessage(new
RPCMessageCallBack());
            }
        });
    }
}
```

RPC Example 4-5

Following Code Snippet demonstrates the module descriptor file:

- **RPC.gwt.xml**

```
...
<module>
...
<!-- Specify the app entry point class.                         -->
     <entry-point class="org.Aptech.client.RPCEntryPoint"/>
</module>
```

Following Code Snippet demonstrates the RPCTest.html file:

```
...
<script type="text/JavaScript"
src="org.Aptech.RPC/org.Aptech.RPC.nocache.js"></script>
    </head>
    <body>
        <div></div>
    </body>
...
```

RPC Example 5-5

Following Code Snippet demonstrates the deployment descriptor:

- **web.xml**

```
...
<servlet-mapping>
     <servlet-name>DemoRPCServiceImpl</servlet-name>
     <url-pattern>/org.Aptech.RPC/rpcdemo</url-pattern>
</servlet-mapping>
<servlet>
  <servlet-name>DemoRPCServiceImpl</servlet-name>
  <servlet-class> org.Aptech.client.rpc.server.DemoRPCServiceImpl</servlet-
class>
</servlet>
...
```

Following figure shows the output of the code when the call is successfully executed:

Google Maps

Google Maps is used off and on to find the location on the map. It allows to view the earth's satellite image and is commonly used for navigation.

Google provides the API that allows a developer to use the map in an application.

All one has to do is to include the Google's JavaScript to the page. The library has some pre-defined objects.

The user needs to specify to those objects which place to show and how to present the map as well as where to show the map in the page.

The following library that is included in the page performs the tasks for the developer: http://maps.googleapis.com/maps/api/js

Map Options

- The Map Options need to be set to specify the location and display details for the map.

Following table lists few basic Map Options:

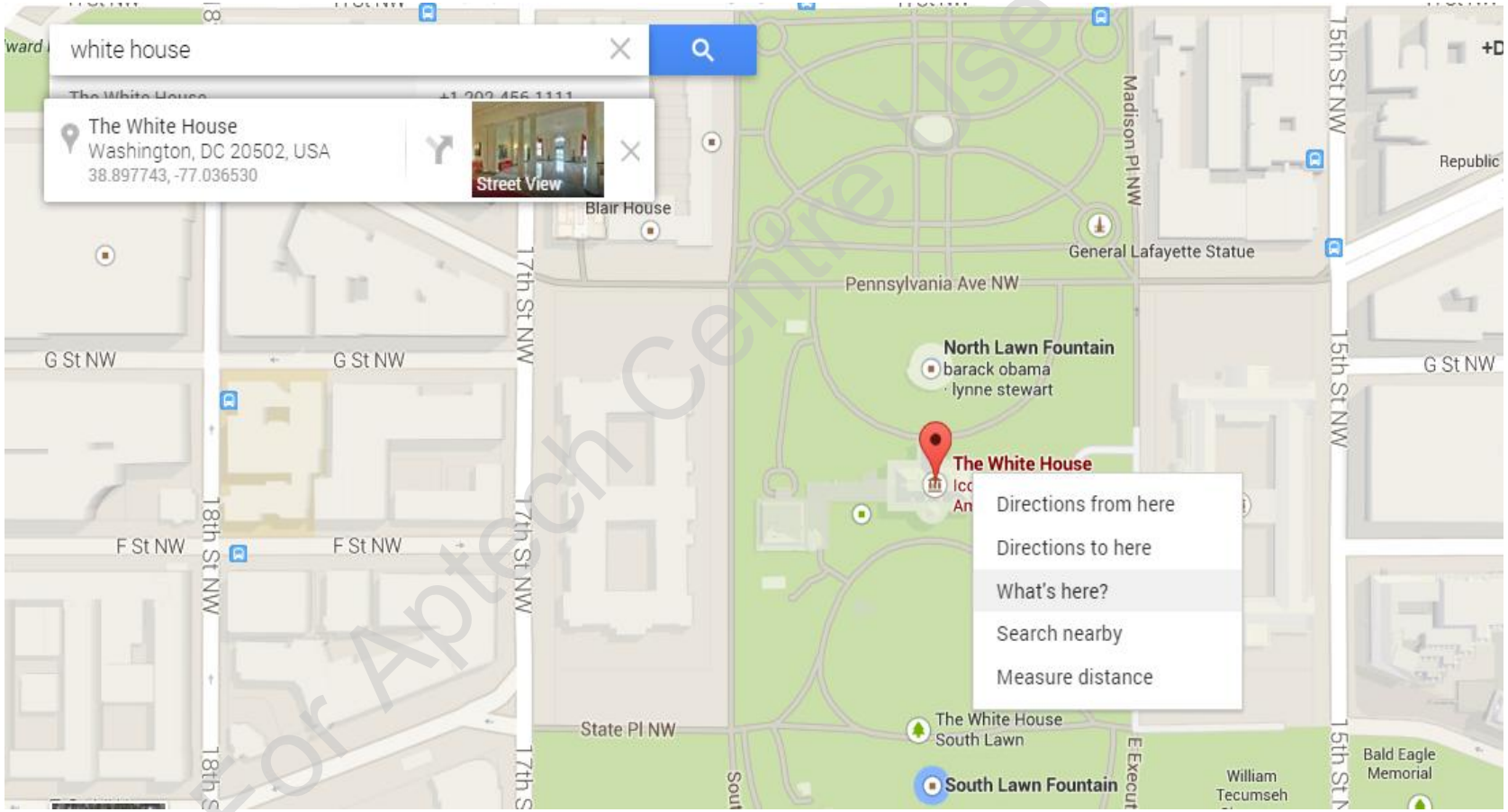
Option Key	Type	Description
Center	LatLng	LatLng, that is, latitude and longitude, is a data type that is included in the API. This option specifies which point should be the centre of the Map.
keyboardShortcuts	Boolean	Indicates if the Keyboard shortcut needs to be enabled or not.
Zoom	Number	The initial default zoom level when the map is loaded.
zoomControl	Boolean	Indicates if the zoom control is needed or not.
zoomControlOptions	zoomControlOptions	Indicates how control options for zoom should be.
Maptypeid	MapTypeId	Specifies how the default Map type should be. The map can be Hybrid, Road map, Satellite, and Terrain.

Longitude and Latitudes 1-2

- Longitude and Latitudes are the numbers which denotes a particular point on the earth.
- They are similar to the X and Y co-ordinates that denote a point on a 2-D graph.
- The latitudes and longitudes are imaginary, but standard.
- That is, Latitude and Longitude of a particular point remains the same always.
- To get the Latitude and Longitude of a particular point,
  - Launch Google Maps (https://maps.google.com).
  - Search for the place of interest.
  - Right–click the location and click 'What's here?'.

Longitude and Latitudes 2-2

- The latitude and longitude will appear in the top–left corner of the page as shown in the following figure:



Map Object 1-2

- The instance of google.maps.Map represents a Map.
- So whenever a map is needed, the instance of google.maps.Map must be created as follows:
  - `new google.maps.Map(document.getElementById("divToContainMap"), aVariableWithMapOptions);`

Following Code Snippet demonstrates the use of a map in a div of dimension 400X400 with the centre of the map as White House:

- **WhiteHouse.html**

```
...
<script src="http://maps.googleapis.com/maps/api/js"></script>
<script>
function init(){
  // Give the Latitude and longitude of White House and the initial zoom level
  var aMapPosition = {
    center:new google.maps.LatLng(38.897743, -77.036530),
    zoom:18
  };
// The map will be displayed in the div with id googleMap
  new google.maps.Map(document.getElementById("googleMap"),aMapPosition);
}
```

Map Object 2-2

```
// The call to show the map will happen on window load.
google.maps.event.addDomListener(window, 'load', init);
</script>
</head>
<body>
   <div id="googleMap" style="width:400px;height:400px;"></div>
</body>
...
```

Following figure shows the output pointing to the location on the map based on the latitude and longitude specified in the code:

Google AJAX Search API 1-2

The Google AJAX Search API allows including Google Search in Web pages with JavaScript.

That is, one can embed a simple, dynamic search box, and display search results in Web pages or use the results in innovative and programmatic ways.

The Google AJAX Search API allows developers to integrate Web Search, News Search, and Blog Search into their Web site.

The developers can add Local Search results to the Web site, or integrate them with their Google Maps API mashup.

It also allows adding YouTube Videos and Google Image Search results to a Web site or blog.

- This API is now deprecated and the functionality of this API is now covered by the Google Custom Search API.

Google AJAX Search API 2-2

- Developers can now exploit the full power of ideas to dynamically generate Custom Search Engines.
- One can host the CSE specification on the Web site and include the url for this specification in the CSE search request.
- Google retrieves the CSE specification from the Website when a user searches in the CSE.

There are two editions of Google Custom Search as follows:

Custom Search Engine (basic edition)

- With Google Custom Search, developers can add a search box to the homepage to help people find required content on their Website.

Google Site Search (business edition)

- Google Site Search includes the same search technology that is applied in Google.com in the developer's Website, to display relevant results with lightning speed.

Summary

- The GWT compiler translates Java code into JavaScript/CSS/HTML as applicable.
- Any GWT application must have atleast one class that implements com.google.gwt.core.client.EntryPoint.
- A module is an entity that contains one or more Entry Points. A GWT application can be said as a collection of one or more modules.
- There must be some HTML or other server pages that include the scripts and hook-up those GWT functionalities into the application.
- Web pages generated by GWT cannot be indexed by search engines because these applications are generated dynamically.
- GWT Events are similar to the events in other Java UI frameworks such as AWT or Swing.