



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT – IV-Rich Internet Applications – SCS1401

IV.Introduction to GWT

Overview - Why Gwt - GWT compiler - Cross browser compatibility. Basic GWT - GWT modules - Build and deploy simple GWT application - GWT Widget Library.

Overview

Google Web Toolkit (GWT) is a development toolkit to create Rich Internet Applications (RIA).

Features of GWT are

- GWT provides developers option to write client-side application in JAVA.
- GWT compiles the code written in JAVA to JavaScript code.
- Application written in GWT is cross-browser compliant. GWT automatically generates Java script code suitable for each browser.
- GWT is open source, completely free, and used by thousands of developers around the world. It is licensed under the Apache License version 2.0.
- GWT is a framework to build large-scale high-performance web application while keeping them as easy-to-maintain.
- Browser supports (IE, Firefox, Mozilla, Safari, and Opera)
- Browser history management
- Dynamic, reusable UI components
- Really simple RPC
- Real debugging
- Completely Open Source

Why GWT?

- All of our favourite Java development tools such as Eclipse, Junit, IntelliJ and J Profiler can be used for AJAX development.
- Static type checking in Java language boosts productivity while reducing errors.
- Common Java script errors are easily caught at compile time rather than by users at run time.
- Code prompting/completion is widely available.
- Java based OO designs are easier to communicate and understand, thus making AJAX code base more comprehensible with less documentation.
- GWT provides easy integration with Junit and Maven.
- Being Java based, GWT has a low learning curve for Java Developers.
- GWT generates optimized Java script code, produces browser's specific Javascript code by self.

- GWT provides Widgets library provides most of tasks required in an application.
- GWT is extensible and custom widget can be created to cater to application needs.
- On top of everything, GWT applications can run on all major browsers and smart phones including Android and iOS-based phones/tablets.

Disadvantages of GWT

Although GWT offers plenty of advantages, it suffers from the following disadvantages

- Not Indexable – Web pages generated by GWT would not be indexed by search engines because these applications are generated dynamically.
- Not Degradable – If your application user disables Javascript then user will just see the basic page and nothing more.
- Not Designer's Friendly – GWT is not suitable for web designers who prefer using plain HTML with placeholders for inserting dynamic content at later point in time

The GWT Components

The GWT framework can be divided into following three major parts –

- GWT Java to JavaScript compiler – This is the most important part of GWT which makes it a powerful tool for building RIAs. The GWT compiler is used to translate all the application code written in Java into JavaScript.
- JRE Emulation library – Google Web Toolkit includes a library that emulates a subset of the Java runtime library. The list includes java.lang, java.lang.annotation, java.math, java.io, java.sql, java.util and java.util.logging
- GWT UI building library – This part of GWT consists of many subparts which includes the actual UI components, RPC support, History management, and much more.

GWT also provides a GWT Hosted Web Browser which lets you run and execute your GWT applications in hosted mode, where your code runs as Java in the Java Virtual Machine without compiling to JavaScript.

Cross browser compatibility

GWT shields you from worrying too much about cross-browser incompatibilities. If you stick to built-in widgets and composites, your applications will work similarly on the most recent versions of Internet Explorer, Firefox, Chrome, and Safari. (Opera, too, most of the time.) HTML user interfaces are remarkably quirky, though, so make sure to test your applications thoroughly on every browser.

Whenever possible, GWT defers to browsers' native user interface elements. For example, GWT's Button widget is a true HTML <button> rather than a synthetic button-like

widget built, say, from a <div>. That means that GWT buttons render appropriately in different browsers and on different client operating systems. We like the native browser controls because they're fast, accessible, and most familiar to users.

Basic Widgets

Every user interface considers the following three main aspects –

UI elements – These are the core visual elements the user eventually sees and interacts with.

Layouts – They define how UI elements should be organized on the screen and provide a final look and feel to the GUI (Graphical User Interface).

Behavior – These are events which occur when the user interacts with UI elements.

GWT UI Elements

The GWT library provides classes in a well-defined class hierarchy to create complex web-based user interfaces. All classes in this component hierarchy has been derived from the UIObject base class shown in Figure 4.1.

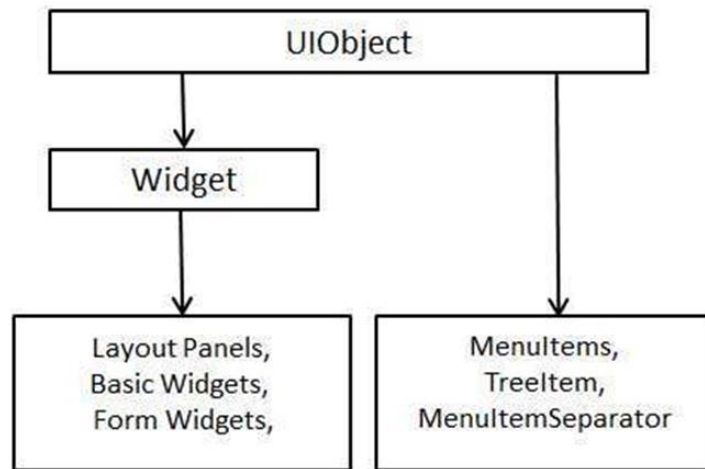


Figure 4.1 Hierarchical structure of GWT UI elements

Every Basic UI widget inherits properties from Widget class which in turn inherits properties from UIObject.

GWT Basic widgets are:

Label: The Label can contain only arbitrary text and it cannot be interpreted as HTML. This widget uses a <div> element, causing it to be displayed with block layout.

HTML: This widget can contain HTML text and displays the html content using a <div> element, causing it to be displayed with block layout.

Image: This widget displays an image at a given URL.

Anchor: This widget represents a simple <a> element.

GWT Form widgets:

The GWT form widgets are discussed in Figure 4.2 and Figure 4.3.

- Button -This widget represents a standard push button.
- PushButton - This widget represents a normal push button with custom styling.
- ToggleButton - This widget represents a stylish stateful button which allows the user to toggle between up and down states.
- CheckBox - This widget represents a standard check box widget. This class also serves as a base class for RadioButton.
- RadioButton - This widget represents a mutually-exclusive selection radio button widget.
- Button -This widget represents a standard push button.
- PushButton - This widget represents a normal push button with custom styling.
- ToggleButton - This widget represents a stylish stateful button which allows the user to toggle between up and down states.
- CheckBox - This widget represents a standard check box widget. This class also serves as a base class for RadioButton.
- RadioButton - This widget represents a mutually-exclusive selection radio button widget.
- Button -This widget represents a standard push button.
- PushButton - This widget represents a normal push button with custom styling.
- ToggleButton - This widget represents a stylish stateful button which allows the user to toggle between up and down states.
- CheckBox - This widget represents a standard check box widget. This class also serves as a base class for RadioButton.
- RadioButton - This widget represents a mutually-exclusive selection radio button widget.

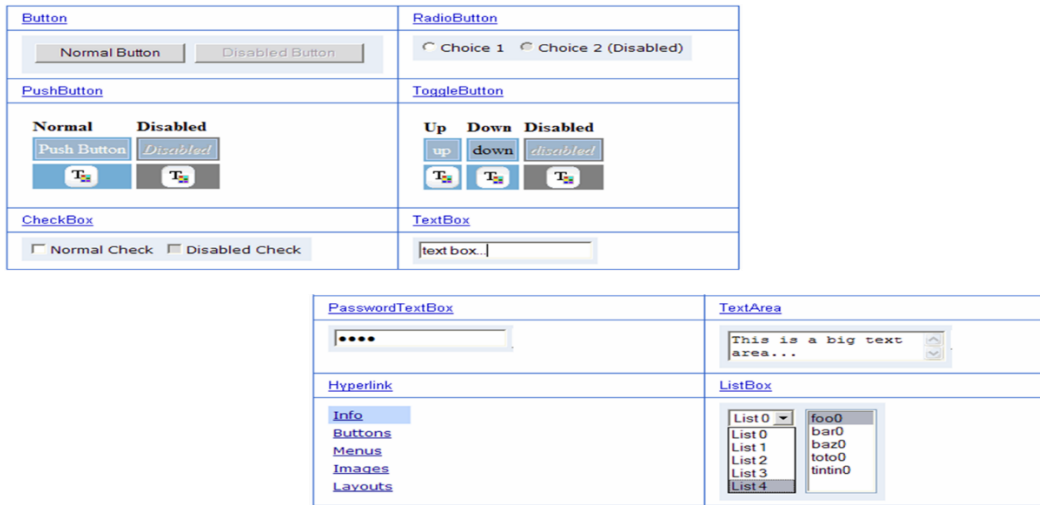


Figure 4.2 Form widgets

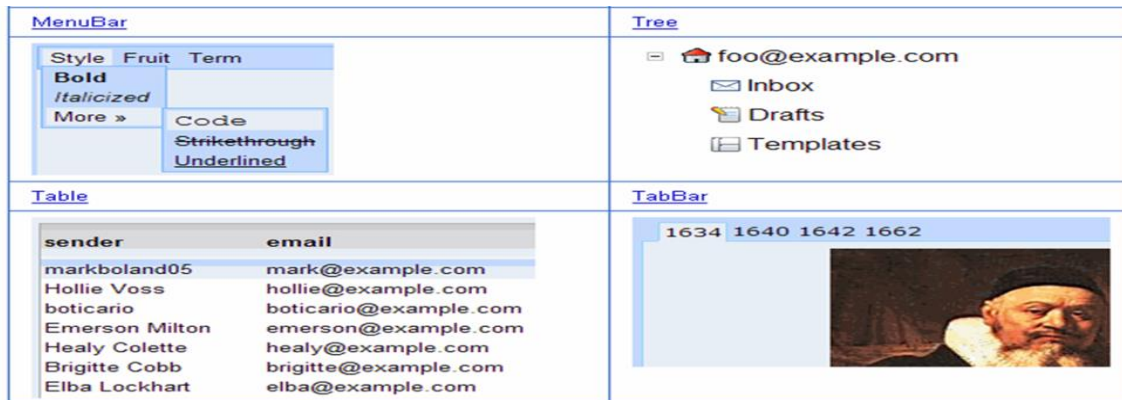


Figure 4.3 Form widgets

GWT Compiler

- The heart of GWT is a compiler that converts Java source into JavaScript,
- i.e transforming your working Java application into an equivalent JavaScript application.
- If your GWT application compiles and runs in hosted mode.
- GWT compiles your application into JavaScript output without complaint,
- Then your application will work the same way in a web browser as it did in hosted mode

Debugging and Deploying GWT Applications

GWT applications can be run in two modes:

1. Development mode (formerly Hosted mode):

- The application is run as Java bytecode within the (JVM).
- This mode is typically used for development, supporting hot swapping of code (Hot swapping and Hot plugging are terms used to describe the functions of replacing computer system components without shutting down the system.) and debugging.

2. Production mode (formerly Web mode):

- The application is run as pure JavaScript and HTML, compiled from the Java source.
- This mode is typically used for deployment.

Google Web Toolkit Architecture

GWT has three major components:

1. Java-to-Javascript compiler

- Translates the Java programming language to the JavaScript programming language.

2. GWT Development Mode

- Allows the developers to run and execute GWT applications in development mode.

3. Two Java class libraries

(i) JRE Emulation library.

(ii) GWT web UI class library.

JRE Emulation library : Google Web Toolkit includes a library that adds a subset of the Java runtime library.

The list includes *java.lang*, *java.lang.annotation*, *java.math*, *java.io*, *java.sql*, *java.util* and *java.util.logging*

GWT UI building library : This part of GWT consists of many subparts which includes the actual UI components, RPC support, History management, and much more shown in Figure 4.4.

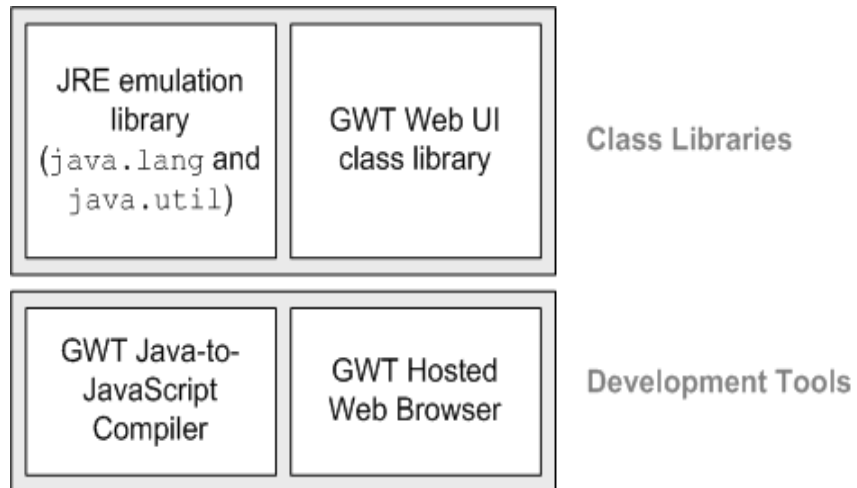


Figure 4.4 GWT UI building library

Google Web Toolkit Features

- Browser supports (IE, Firefox, Mozilla, Safari, and Opera)
- Browser history management
- Dynamic, reusable UI components
- Really simple RPC
- Real debugging
- Completely Open Source

System Requirement

The GWT system requirement is shown in Figure 4.5.

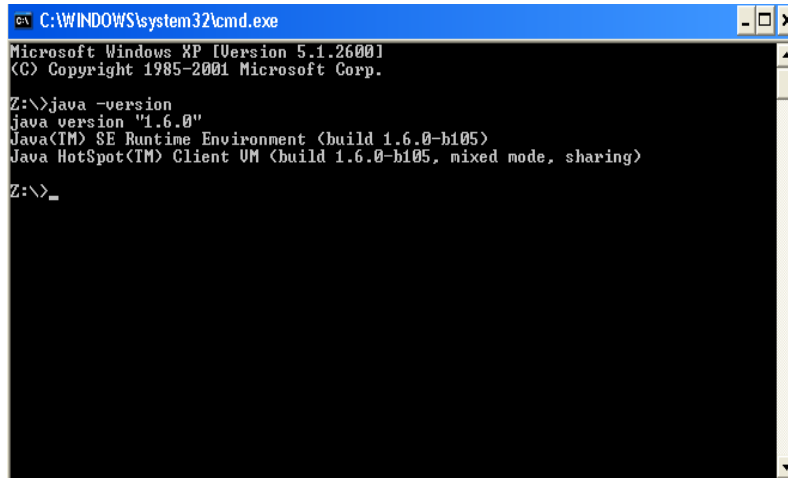
GWT requires JDK 1.6 or higher so the very first requirement is to have JDK installed in your machine.

JDK	1.6 or above.
Memory	no minimum requirement.
Disk Space	no minimum requirement.
Operating System	no minimum requirement.

Figure 4.5 System Requirement

Follow the given steps to setup your environment to start with GWT application development.

- Step 1 -Verify Java installation on your machine
- Now open console and execute the following java command shown in Figure 4.6.



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

Z:\>java -version
java version "1.6.0"
Java(TM) SE Runtime Environment (build 1.6.0-b105)
Java HotSpot(TM) Client VM (build 1.6.0-b105, mixed mode, sharing)
Z:\>_
```

Figure 4.6 Command

Step 2 -Setup Java Development Kit (JDK):

- If you do not have Java installed then you can install the Java Software Development Kit (SDK) from Oracle's Java site: <https://www.oracle.com/java/technologies/javase-downloads.html>
- set PATH and JAVA_HOME environment variables to refer to the directory that contains java and javac, typically java_install_dir/bin and java_install_dir respectively.
- Set the JAVA_HOME environment variable to point to the base directory location where Java is installed on your machine. For example

Step 3 -Setup Eclipse IDE

- To install Eclipse IDE, download the latest Eclipse binaries from <http://www.eclipse.org/downloads/>.
- Once you downloaded the installation, unpack the binary distribution into a convenient location.

For example in C:\eclipse on windows, or

- /usr/local/eclipse on Linux/Unix and finally set PATH variable appropriately.
- Eclipse can be started by executing the following commands on windows machine, or you can simply double click on eclipse.exe
- %C:\eclipse\eclipse.exe

Eclipse can be started by executing the following commands on Unix machine:

- \$/usr/local/eclipse/eclipse
- After a successful startup, if everything is fine then it should display following result in Figure : 4.7.

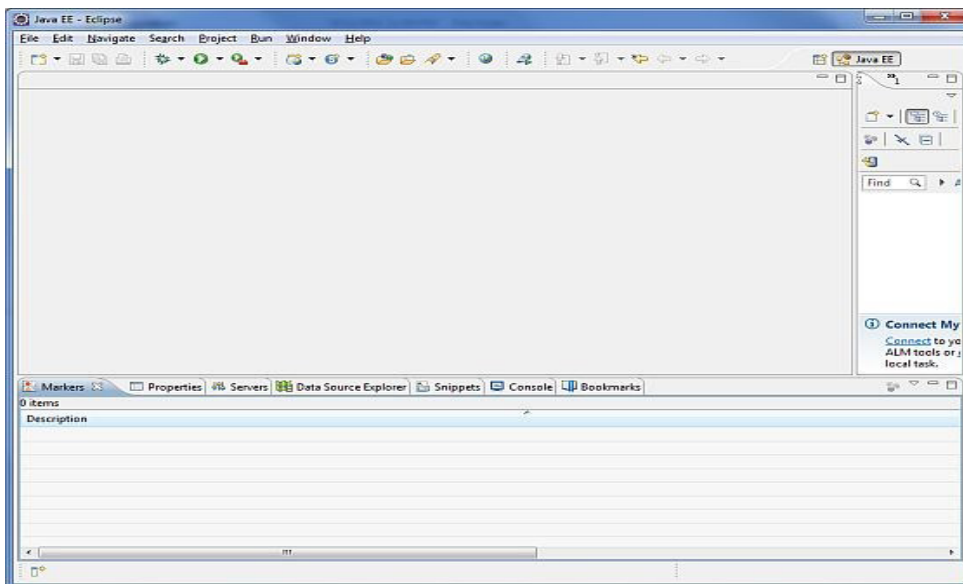


Figure 4.7 Successful Startup

Step 4: Install GWT SDK & Plugin for Eclipse

- **Follow the instructions given**
 - Start Eclipse, then select Help > Install New Software...
 - In the dialog that appears, enter the update site URL into the Work with text box: <http://dl.google.com/eclipse/plugin/4.2>
- After a successful setup for the GWT plugin, if everything is fine then it should display following screen in Figure 4.8 with google icon marked with red rectangle:

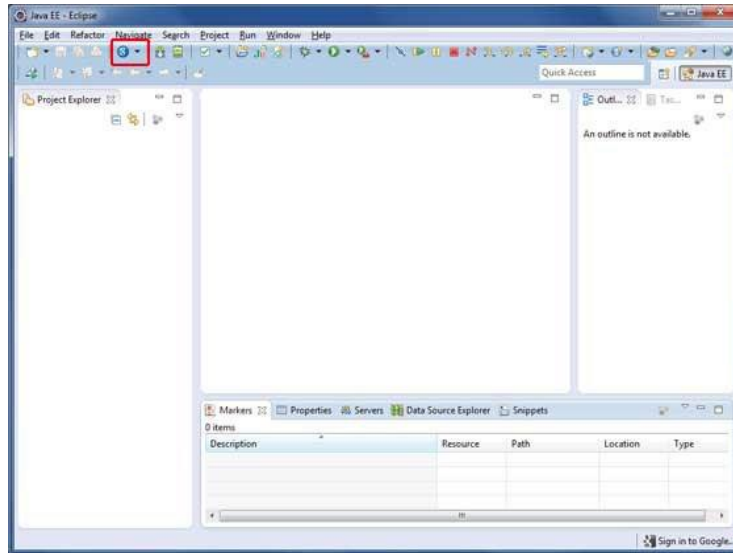


Figure 4.8 Successful Setup

Step 5: Setup Apache Tomcat:

- You can download the latest version of Tomcat from <http://tomcat.apache.org/>.
- Once you downloaded the installation, unpack the binary distribution into a convenient location.

For example in C:\apache-tomcat-6.0.33 on windows,

- /usr/local/apache-tomcat-6.0.33 on Linux/Unix
- and set CATALINA_HOME environment variable pointing to the installation locations.
- Tomcat can be started by executing the following commands on windows machine, or you can simply double click on **startup.bat**
 - %CATALINA_HOME%\bin\startup.bat

or

- C:\apache-tomcat-6.0.33\bin\startup.bat
- Tomcat can be started by executing the following commands on Unix machine:
- \$CATALINA_HOME/bin/startup.sh
- or /usr/local/apache-tomcat-6.0.33/bin/startup.sh
- **The .sh file extension is a shell script.**

After a successful startup, the default web applications included with Tomcat will be available by visiting **http://localhost:8080/**. If everything is fine then it should display following result in Figure 4.9

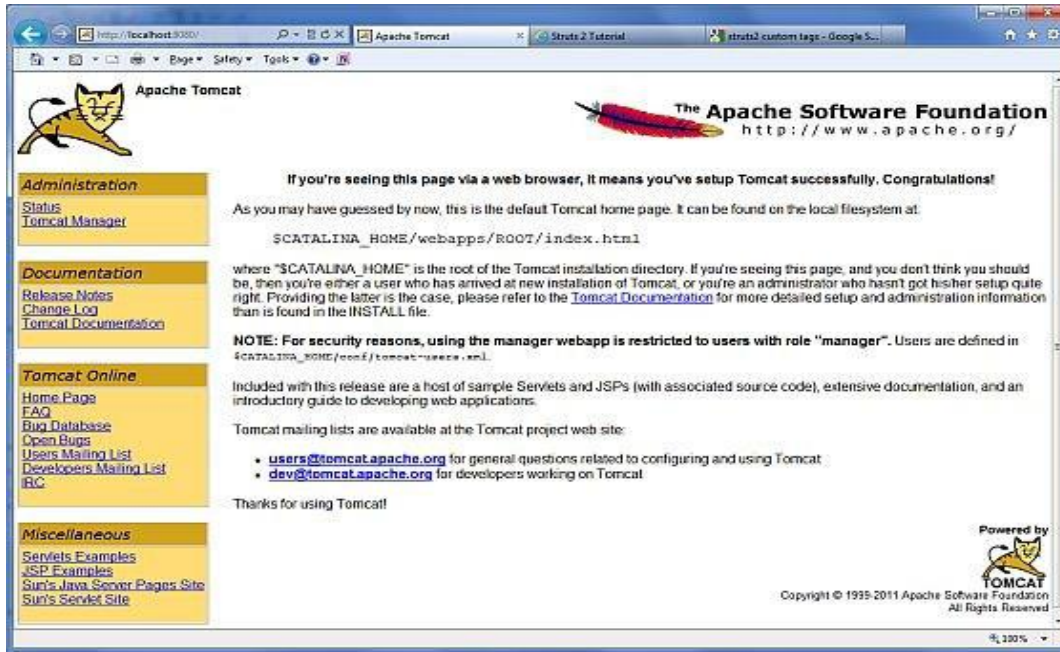


Figure 4.9 Successful Startup

- Tomcat can be stopped by executing the following commands on windows machine:
 - %CATALINA_HOME%\bin\shutdown
 - or
 - C:\apache-tomcat-5.5.29\bin\shutdown
- Tomcat can be stopped by executing the following commands on Unix machine:
 - \$CATALINA_HOME/bin/shutdown.sh
 - or
 - /usr/local/apache-tomcat-5.5.29/bin/shutdown.sh

Applications

A GWT application consists of following four important parts out of which last part is optional but first three parts are mandatory.

1. Module descriptors
2. Public resources
3. Client-side code

4. Server-side code

1. Module Descriptors

- A module descriptor is the configuration file in the form of XML which is used to configure a GWT application.
- A module descriptor file extension is *.gwt.xml,
- where * is the name of the application and this file should reside in the project's root.
- Following will be a default module descriptor HelloWorld.gwt.xml for a HelloWorld application is shown in Figure 4.10.

```
<?xml version="1.0" encoding="utf-8"?>
<modulerefname-to='helloworld'>
<!-- inherit the core web toolkit stuff. -->
<inheritsname='com.google.gwt.user.user' />

<!-- inherit the default gwt style sheet. -->
<inheritsname='com.google.gwt.user.theme.clean.Clean' />

<!-- specify the app entry point class. -->
<entry-pointclass='com.tutorialspoint.client.HelloWorld' />

<!-- specify the paths for translatable code -->
<sourcepath='...' />
<sourcepath='...' />

<!-- specify the paths for static files like html, css etc. -->
<publicpath='...' />
<publicpath='...' />

<!-- specify the paths for external javascript files -->
<scriptsrc="js-url" />
<scriptsrc="js-url" />

<!-- specify the paths for external style sheet files -->
>
<stylesheetsrc="css-url" />
<stylesheetsrc="css-url" />
</module>
```

Figure 4.10 Default Module Descriptor

1. <module rename-to="helloworld">
1. This provides name of the application.
- 2.<inherits name="logical-module-name" />
- This adds other gwt module in java applications.
- 3.<entry-point class="classname" />
- This specifies the name of class which will start loading the GWT Application .
4. <source path="path" />
- This specifies the names of source folders which GWT compiler will search for source compilation.

5. <public path="path" />

- The public path is the place to store resources such as CSS or images .

6. <script src="js-url" />

- Automatically inserts the external JavaScript file located at the location specified by src.

7. <stylesheet src="css-url" />

Automatically inserts the external CSS file located at the location specified by src.

2. Public resources

- These all files referenced by your GWT module ,such as HTML, CSS or Images.
- The location of these resources can be configured using <public path="path" /> .
- When you compile your application into JavaScript, all the files that can be found on your public path are copied to the module's output directory .

```
<html>
<head>
<title>Hello World</title>
<link rel="stylesheet"href="HelloWorld.css"/>
<script language="javascript" src="helloworld/helloworld.nocache.js">
</script> </head>
<body>
<h1>Hello World</h1>
<p>Welcome to first GWT application</p>
</body> </html>
```

Following is the sample style sheet which we have included in our host page

```
body {
text-align: center;
font-family: verdana, sans-serif; }
h1 {
font-size:2em;
```

```
font-weight: bold;
color:red;
margin:40px0px70px;
text-align: center; }
```

3. Client-side code

- The location of these resources can be configured using `<source path="path" />`.
- This is the actual Java code implementing the business logic of the application and that the GWT compiler translates into JavaScript .
- When a module is loaded, every entry point class is detected and its
 - `EntryPoint.onModuleLoad()` method gets called.
- A sample HelloWorld Entry Point class will be as follows

```
Public class HelloWorld implements EntryPoint
{
    public void onModuleLoad()
    {
        Window.alert("Hello, World!");
    }
}
```

4. Server-side code

- This is the server side part of your application and its very much optional.
- If you are not doing any backend processing with-in your application then you do not need this part

Create Application

Let's start with a simple HelloWorld application is shown in Figure 4.11(a),Figure 4.11(b), Figure 4.11(c).

Step 1 -Create Project

- The first step is to create a simple Web Application Project using Eclipse IDE.
- Launch project wizard using the option

- **Google Icon > New Web Application Project....**

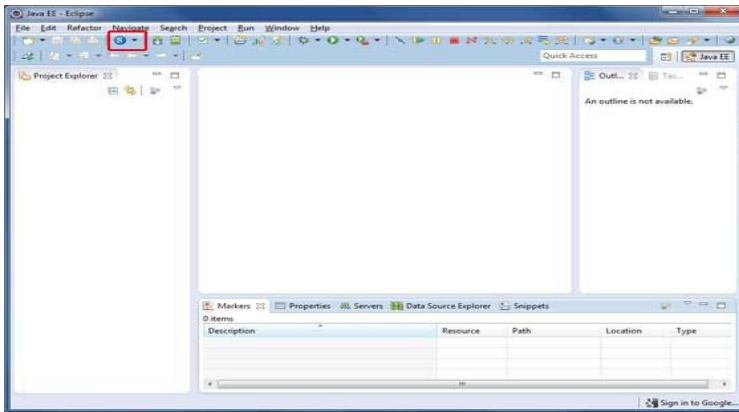


Figure 4.11(a) Create Application-step1

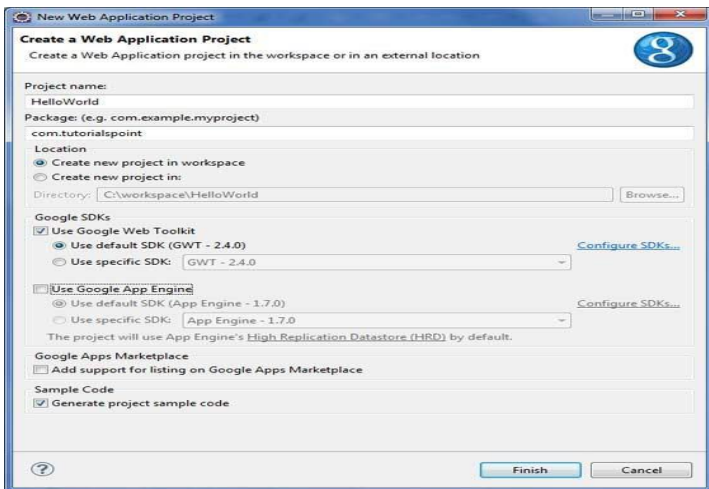


Figure 4.11(b) Create Application-step2

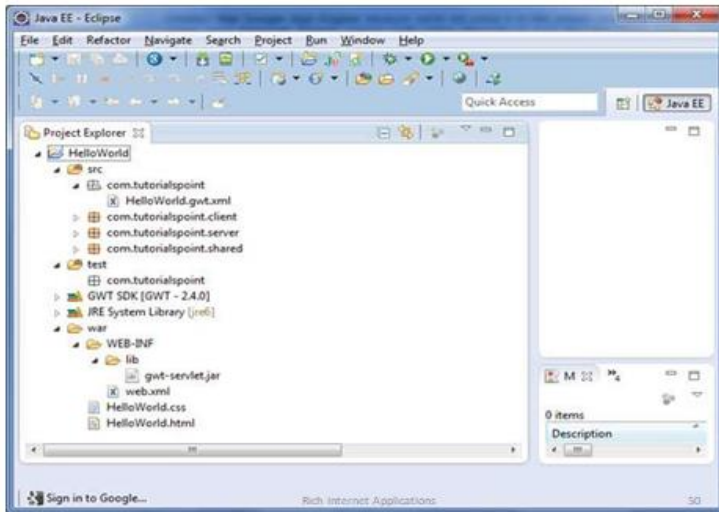


Figure 4.11(c) Create Application-Step 3

- **Description of all important folders**

SRC:

- ✓ Source code (java classes) files.
- ✓ Client folder containing the client-side specific java classes responsible for client UI display.
- ✓ Server folder containing the server-side java classes responsible for server side processing.
- ✓ Shared folder containing the java model class to transfer data from server to client and vice versa.
- ✓ HelloWorld.gwt.xml, a module descriptor file required for GWT compiler to compile the HelloWorld project.

TEST:

- Test code (java classes) source files.
- Client folder containing the java classes responsible to test gwt client side code.
- This is the most important part, it represents the actual deployable web application.
- WEB-INF containing compiled classes, gwt libraries, servlet libraries, HelloWorld.css, project style sheet.
- HelloWorld.html, hosts HTML which will invoke GWT UI Application.

Step 2 -Modify Module Descriptor: HelloWorld.gwt.xml

GWT plugin will create a default module descriptor file - *src/HelloWorld.gwt.xml* which is given below :

```
<?xml version="1.0" ?>
```

```

<modulerefname-to='helloworld'>
<!-- Inherit the core Web Toolkit stuff. -->
<inheritsname='com.google.gwt.user.User'/>
<!-- Inherit the default GWT style sheet. You can change -->
<!-- the theme of your GWT application by uncommenting -->
<!-- any one of the following lines. -->
<inheritsname='com.google.gwt.user.theme.clean.Clean'/>
<!--<inherits name='com.google.gwt.user.theme.chrome.Chrome'/> -->
<!--<inherits name='com.google.gwt.user.theme.dark.Dark'/> -->
<!-- Other module inherits -->
<!-- Specify the app entry point class. -->
<entry-pointclass='com.client.HelloWorld'/>
<!-- Specify the paths for translatable code -->
<sourcepath='client'/>
<sourcepath='shared'/>
</module>

```

Step 3 -Modify Style Sheet: HelloWorld.css

- GWT plugin will create a default Style Sheet file *war/HelloWorld.css*.
- *Let us modify this file to keep our example at simplest level of understanding:*

```

body { text-align: center;
font-family: verdana, sans-serif; }
h1 { font-size:2em;
font-weight: bold;
color:red;
margin:40px0px70px;
text-align: center; }

```

Step 4 -Modify Host File: HelloWorld.html

GWT plugin will create a default HTML host file *war/HelloWorld.html*.

Let us modify this file to keep our example at simplest level of understanding:

```
<html> <head> <title>Hello World</title>
<linkrel="stylesheet"href="HelloWorld.css"/>
<scriptlanguage="javascript"src="helloworld/helloworld.nocache.js">
</script></head> <body>
<h1>Hello World</h1>
<p>Welcome to first GWT application</p>
</body> </html>
```

Step 5 -Modify Entry Point: HelloWorld.java

- GWT plugin will create a default Java file *src/com.HelloWorld.java*, which keeps an entry point for the application.
- *Let us modify this file to display "Hello,World!":*

```
package com.client;
import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.user.client.Window;
Public class HelloWorld implements EntryPoint{
Public void onModuleLoad(){
Window.alert("Hello, World!");
} }

```

Step 6 -Compile Application

- Once you are ready with all the changes done, its time to compile the project.
- Use the option Google Icon > GWT Compile Project...
- to launch GWT Compile dialogue box as shown below in Figure 4.12:

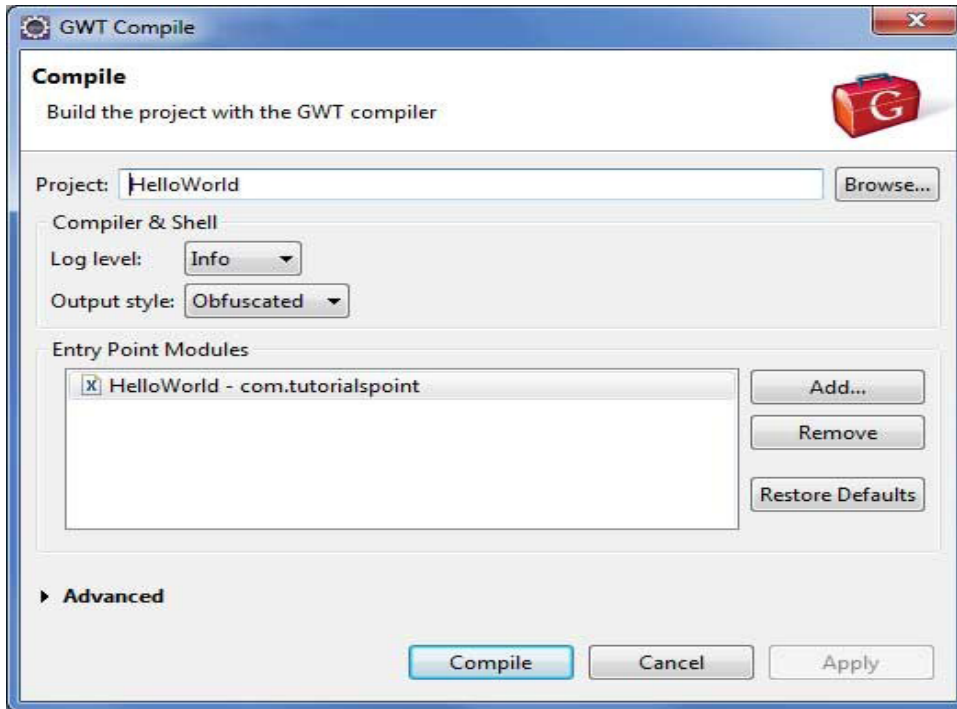


Figure 4.12 GWT Compile Dialogue Box

- Click Compile button. If everything goes fine, you will see following output in Eclipse console
- Compiling module com.HelloWorld
- Compiling 6 permutations
- Compiling permutation 0...
- Compiling permutation 1...
- Compiling permutation 2...
- Compiling permutation 3...
- Compiling permutation 4...
- Compiling permutation 5...
- Compile of permutations succeeded
- Linking into C:\workspace\HelloWorld\war\helloworld
- Link succeeded

Step 7 -Run Application

- Now click on Run application menu and select Hello World application to run the application in Figure 4.13.



Figure 4.13 Run Application

If everything is fine, you must see GWT Development Mode active in Eclipse containing a URL as shown below in Figure 4.14.

Double click the URL to open the GWT application.

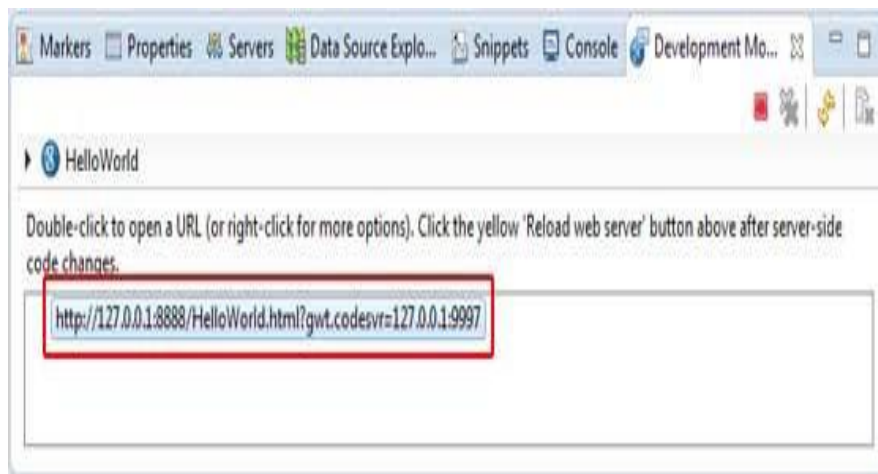


Figure 4.14 GWT Development Mode

Deploy Application

- Create WAR(Web application Archive) File
- Now our applicaion is working fine and we are ready to export it as a war file. Follow the following steps:

- Go into your project's war directory
C:\workspace\HelloWorld\war
- Select all the files & folders available inside war directory.
- Zip all the selected files & folders in a file called HelloWorld.zip.
- Rename HelloWorld.zip to HelloWorld.war.

Deploy WAR file

- Stop the tomcat server.
- Copy the HelloWorld.war file to tomcat installation directory > webapps folder.
- Start the tomcat server.
- Look inside webapps directory, there should be a folder helloworld got created.
- Now HelloWorld.war is successfully deployed in Tomcat Webserver root.

Run Application –

- Enter a url in web browser: <http://localhost:8080/HelloWorld> to launch the application in figure 4.15.
- Server name (localhost) and port (8080) may vary as per your tomcat configuration.

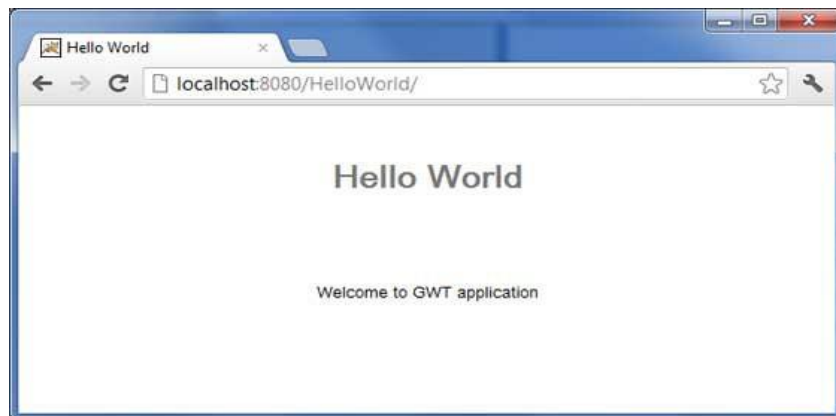


Figure 4.15 Execution of Hello Application

GWT Widgets

- Button
- PushButton
- RadioButton
- CheckBox

- DatePicker
- ToggleButton
- TextBox
- PasswordTextBox
- TextArea
- Hyperlink
- ListBox
- CellList
- MenuBar
- Tree
- CellTree
- SuggestBox
- RichTextArea
- FlexTable
- Grid
- CellTable
- CellBrowser
- TabBar D
- DialogBox

TextBox Widget Example

src/com.text/HelloWorld.gwt.xml.

```
<?xml version="1.0" >  
<module rename-to='helloworld'>  
<inherits name='com.google.gwt.user.User'/>  
<inherits name='com.google.gwt.user.theme.clean.Clean'/>  
<entry-point class='com.tutorialspoint.client.HelloWorld'/>  
<source path='client'/>  
<source path='shared'/>
```

```
</module>
```

war/HelloWorld.css

```
body{  
text-align: center;  
font-family: verdana, sans-serif; }  
h1{  
font-size:2em;  
font-weight: bold;  
color:red;  
margin:40px0px70px;  
text-align: center; }  
.gwt-TextArea{ color: green; }  
.gwt-TextArea-readonly{ background-color: yellow; }  
war/HelloWorld.html.
```

```
<html> <head>  
<title>Hello World</title>  
<link rel="stylesheet" href="HelloWorld.css"/>  
<script language="javascript" src="helloworld/helloworld.js">  
</script>  
</head><body>  
<h1>TextArea Widget Demonstration</h1>  
<div id="gwtContainer"></div>  
</body></html >
```

src/com.text/HelloWorld.java

```
package com.tutorialspoint.client;  
import com.google.gwt.core.client.EntryPoint;  
import com.google.gwt.user.client.ui.RootPanel;  
import com.google.gwt.user.client.ui.TextArea;
```



```

import com.google.gwt.user.client.ui.VerticalPanel;
public class HelloWorld implements EntryPoint{
public void onModuleLoad(){
//create textarea elements
TextArea textArea1 =new TextArea();
TextArea textArea2 =new TextArea();
//set width as 10 characters
textArea1.setCharacterWidth(20);
textArea2.setCharacterWidth(20);
//set height as 5 lines
textArea1.setVisibleLines(5);
textArea2.setVisibleLines(5);
//add text to text area
textArea2.setText(" Hello World! \n Be Happy! \n Stay Cool!");
//set textbox as readonly
textArea2.setReadOnly(true);
// Add text boxes to the root panel.
VerticalPanel panel =newVerticalPanel();
panel.setSpacing(10);
panel.add(textArea1);
panel.add(textArea2);
RootPanel.get("gwtContainer").add(panel);
}
}

```

GWT Widget Library

- Widgets included with Service Portal can be customized to suit your own needs or as a basic code sample for you to refer to as you are building your own widgets.
- Widgets included with Service Portal can be customized to suit your own needs or as a basic code sample for you to refer to as you are building your own widgets.

- Each widget provides end-user functions such as data visualization, remote device control, alarms management and displaying static custom html content.

Digital Gauges

Useful for visualization of temperature, humidity, speed and other integer or float values shown in Figure 4.16.

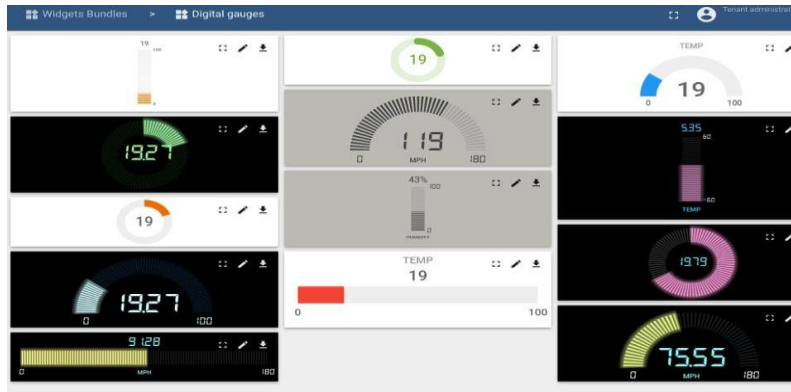


Figure 4.16 Digital Gauges

Analog Gauges

Similar to digital gauges, but have a different style in Figure 4.17.



Figure 4.17 Analog Gauges

Charts

Useful for visualization of historical or real-time data with a time window in Figure 4.18.

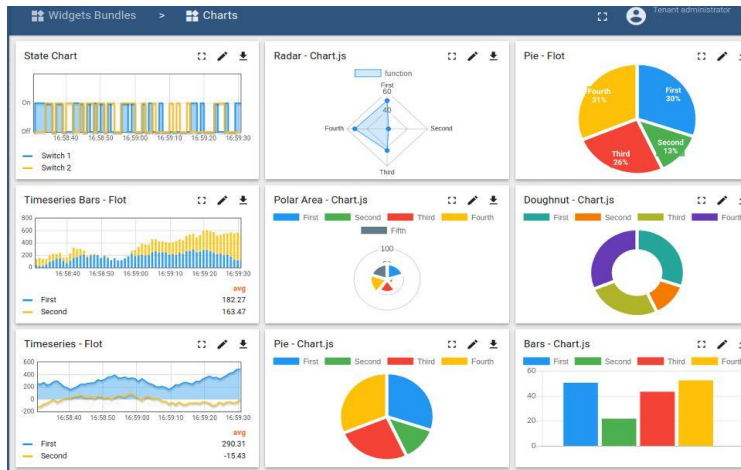


Figure 4.18 Charts

Maps widgets

The maps widgets are shown in Figure 4.19.

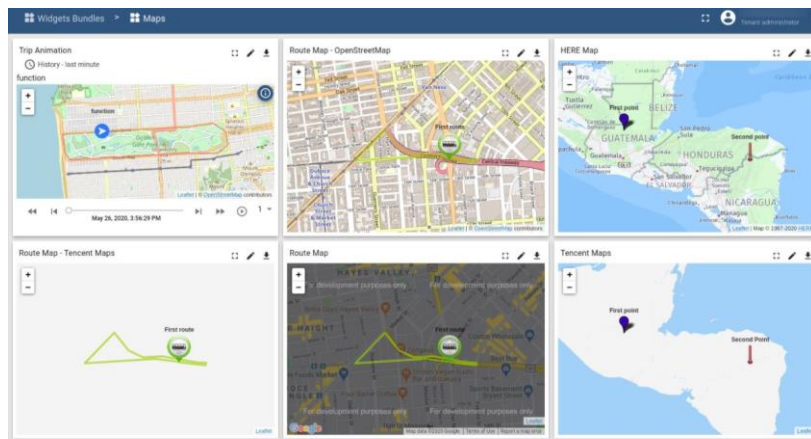


Figure 4.19 Maps Widgets

Announcements widget

Users can view all active announcements. You can use this base system widget as-is in your Service Portal or clone it to suit your own business needs as shown in Figure 4.20.

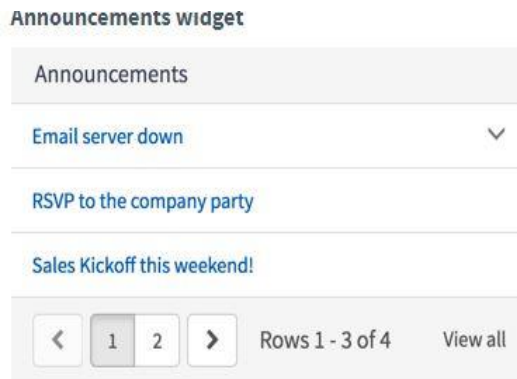


Figure 4.20 Announcement Widgets

- **Announcements widget**-Users can view all active announcements. You can use this base system widget as-is in your Service Portal or clone it to suit your own business needs.
- **Approvals widget**-Users can approve or reject items directly within Service Portal. You can use this base system widget as-is in your Service Portal or clone it to suit your own business needs.
- **Approval Info widget**-The Approval Info widget works in tandem with the Approval widget to display details about the approval request. You can use this base system widget as-is in your Service Portal or clone it to suit your own business needs.
- **Approval Record widget**-The Approval Record widget shows the full record for an approval including the activity stream. You can use this base system widget as-is in your Service Portal or clone it to suit your own business needs.
- **Breadcrumbs widget**-The breadcrumbs widget allows users to easily navigate around a portal. You can use this base system widget as-is in your Service Portal or clone it to suit your own business needs.
- **Breakout Game widget**-Add a fun, interactive 404 page to pages that do not exist using the Breakout Game widget. You can use this base system widget as-is in your Service Portal or clone it to suit your own business needs.
- **Calculator widget**-The calculator widget does simple calculations. You can use this base system widget as-is in your Service Portal or clone it to suit your own business needs.
- **Carousel widget**-Showcase specific items in your catalog using a scrolling list of images in the carousel widget. You can use this base system widget as-is in your Service Portal or clone it to suit your own business needs.
- **Change Password widget**-Users can change their passwords using the Change Password widget.

- **Cool Clock widget**-Show different times around the world using the Cool Clock widget. You can use this base system widget as-is in your Service Portal or clone it to suit your own business needs.
- **Data table from instance definition widget**-Display a filtered list on your portal using the data table from instance definition widget. You can use this base system widget as-is in your Service Portal or clone it to suit your own business needs.
- **Data Table from URL definition widget**-The Data Table from URL definition widget displays the table you select from the list. You can use this base system widget as-is in your Service Portal or clone it to suit your own business needs.
- **Form widget**-The form widget is a platform form within the Service Portal UI with a few differences. You can use this base system widget as-is in your Service Portal or clone it to suit your own business needs.
- **Header menu widget**-The Header Menu widget controls which options appear in the page header. You can use this base system widget as-is in your Service Portal or clone it to suit your own business needs.
- **Hello World widgets**-The Hello World widgets are included with Service Portal as examples of how to use and create widgets. You can use this base system widget as-is in your Service Portal or clone it to suit your own business needs.
- **HTML widget**-Use the HTML widget to directly inject HTML, text, lists, or content in general into a page. You can use this base system widget as-is in your Service Portal or clone it to suit your own business needs.
- **Icon Link widget**-Link to any other item. You can use this base system widget as-is in your Service Portal or clone it to suit your own business needs.
- **Icon menu list widget**-A simple list with a glyph icon next to each link. You can use this base system widget as-is in your Service Portal or clone it to suit your own business needs.
- **Language Switch widget**-Add the Language Switch widget to a landing or homepage to allow your users to change the language of the page. You can use this base system widget as-is in your Service Portal or clone it to suit your own business needs.
- **Link button widget**-The Link Button widget is a button you can nest in any other widget that links to another destination. You can use this base system widget as-is in your Service Portal or clone it to suit your own business needs.
- **Login widget**-The login widget controls user access to your site. You can use this base system widget as-is in your Service Portal or clone it to suit your own business needs.
- **My Requests widget**-The My Requests widget stores all of your open requests in one place. You can use this base system widget as-is in your Service Portal or clone it to suit your own business needs.

- **My Requests widget**-Use the My Requests widget (my-requests-v2) to enable requesters to view open or closed requests in Service Portal. Requests, incidents, and tasks are displayed in a single view that is based on the filter conditions and display settings in the My Request Filter module. For information on defining filters for this module, refer to the Related information section.
- **Organization Chart widget**-The Organization Chart widget shows employees in a tree structure relative to their manager. You can use this base system widget as-is in your Service Portal or clone it to suit your own business needs.
- **Simple List widget**-The Simple List widget can be used to display any list in the system within Service Portal. You can use this base system widget as-is in your Service Portal or clone it to suit your own business needs.
- **Sample Footer widget**-The Sample Footer widget is an example of a footer you can use in your portal. You can use this base system widget as-is in your Service Portal or clone it to suit your own business needs.
- **Ticket Attachments widget**-Use the attachment widget to attach items to tickets. You can use this base system widget as-is in your Service Portal or clone it to suit your own business needs.
- **Ticket Conversations widget**-Record of ticket items. Users can use this widget to communicate back and forth with the fulfiller and the receiver. You can use this base system widget as-is in your Service Portal or clone it to suit your own business needs.
- **Ticket Fields widget**-The Ticket Fields widget displays information about a request that a user has made. You can use this base system widget as-is in your Service Portal or clone it to suit your own business needs.
- **Ticket Location widget**-Share your location in a ticket. You can use this base system widget as-is in your Service Portal or clone it to suit your own business needs.
- **User Profile widget**-Display user profile information. You can use this base system widget as-is in your Service Portal or clone it to suit your own business needs.

References

1. Federico Kerek , “ Essential GWT: Building for the Web with Google Web Toolkit 2 ”, Addison-Wesley Professional,2010.

