1/3/2017

1. Google Web Toolkit (GWT) is a development toolkit for building and optimizing complex browser-based applications.
2. GWT is open source, cross-browser, completely free. It is licensed under the Apache License version 2.0.
3. **Write in Java, Run in JavaScript**
4. Prerequisites 🡺HTML, CSS, AJAX etc.

## What is GWT?

1. GWT provides developers option to write client side application in JAVA.
2. GWT compiles the code written in JAVA to JavaScript code.
3. GWT automatically generates JavaScript code **suitable for each browser**.

Overall, GWT is a framework to build large scale and high performance web application while keeping them as easy-to-maintain.

## Why to use GWT?

1. Being Java based, you can use JAVA IDEs like Eclipse to develop GWT application. Developers can use code auto-complete/refactoring/navigation/project management and all features of IDEs.
2. GWT provides full debugging capability. **Developers can debug the client side application just as a Java Application.**
3. **GWT provides easy integration with Junit and Maven.**
4. GWT provides **Widgets library**
5. **GWT is extensible and custom widget can be created to cater to application needs.**

## Disadvantages of GWT

1. **Not degradable**: If your application user disables Javascript then user will just see the basic page and nothing more.
2. **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**
* **JRE Emulation library**
* **GWT UI building library**

**GWT Java to JavaScript compiler**

1. 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**

1. 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.*

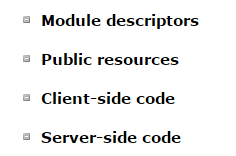
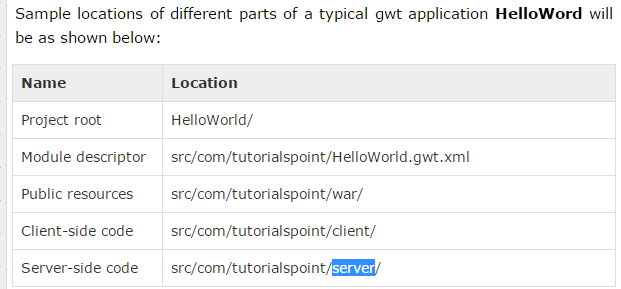
**GWT UI building library**

1. This part of GWT consists of many subparts which includes the actual UI components, **RPC support, History management**, and much more.

# GWT - Environment Setup

1. JDK 6 and above
2. IDE🡺Eclipse
3. GWT SDK & Plugin for Eclipse
4. tomcat-6.0.33 on windows

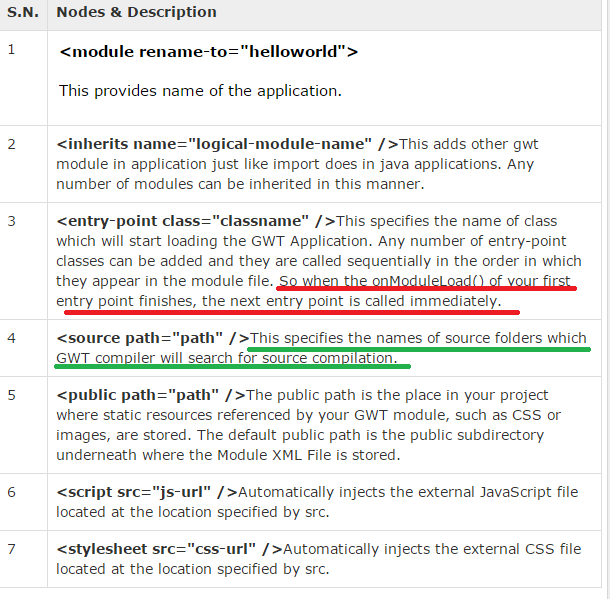
# GWT - Applications 🡺let us see what are the actual parts of a GWT application

1. A GWT application consists of following four important parts out of which last part is optional but first three parts are mandatory:
2. 

## Module Descriptors

1. A module descriptor is the configuration file in the form of XML which is used to configure a GWT application.
2. A module descriptor file extension is **\*.gwt.xml**,
3. 

Let’s see the above screen shot explanation below,

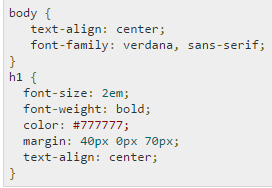


## Public resources

1. These are all files referenced by your GWT module, such as Host HTML page, CSS or images.
2. The location of these resources can be configured using <public path="path" /> element in module configuration file. **By default, it is the public subdirectory underneath where the Module XML File is stored.**
3. **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.**
4. **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.**
5. The most important public resource is host page which is used to invoke actual GWT application.
6. A typical HTML host page for an application might not include any visible HTML body content at all but it is always expected to include GWT application via a <script.../> tag as follows:



Now let’s see the sample style sheet which we have included in our host page:

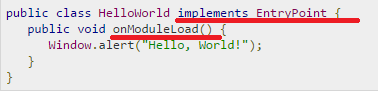


## Client-side code

1. This is the actual Java code written implementing the business logic of the application and that the GWT compiler translates into JavaScript.
2. The location of these resources can be configured using **<source path="path" /> element** in module configuration file.

For example **Entry Point** code will be used as client side code and its location will be specified using <source path="path" />.

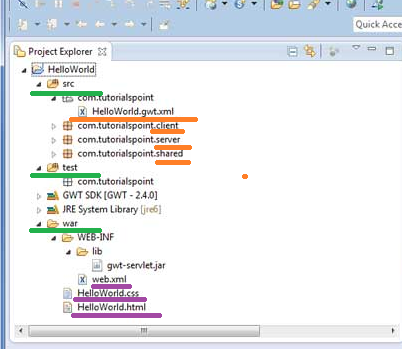
1. When a module is loaded, every entry point class is instantiated and its **EntryPoint.onModuleLoad()** method gets called.



## Server-side code

# GWT - Create Application

Below is the Project Structure,



Explanation for the above screen shot,

1. We have 3 folders namely,
2. src🡺 contains

* Source code (java classes)
* 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

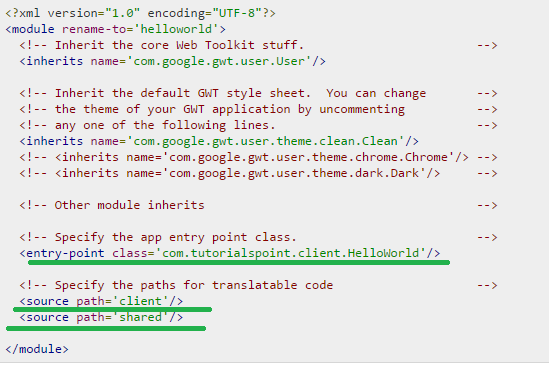
1. test🡺 contains,

* Test code (java classes) source files

1. War 🡺 contains,

* 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, hots HTML which will invoke GWT UI Application

Finally in this Topic I am going to Just cover the Modified configuration file, if I wanted modified css, html or java file then see the Tutorials point



# GWT - Style with CSS

1. GWT widgets rely on cascading style sheets (CSS) for visual styling.
2. By default, the class name for each component is  .**gwt- <classname>**
3. For example, the Button widget has a default style of ***gwt-Button***and similar way TextBox widgets has a default style of ***gwt-TextBox***.
4. Let’s consider the practical Scenario

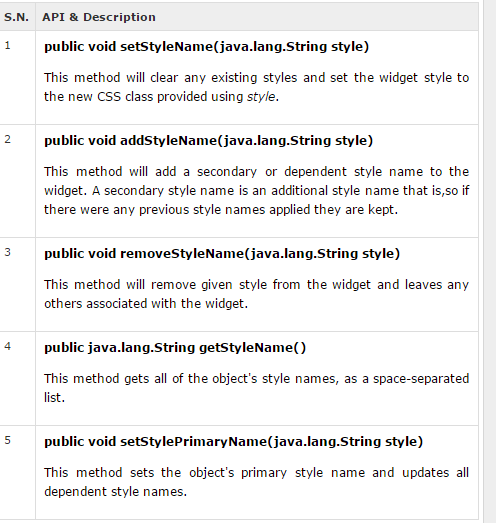


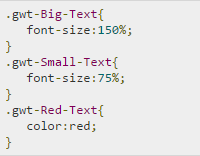
1. By default, neither the browser nor GWT creates default **id** attributes for widgets 🡺 **ms this means only default class is created**
2. You must explicitly create a unique id for the elements which you can use in CSS.
3. For Example 🡺 In order to give a particular button with id say for example **my-button-id**  and I wanted to have larger font, you could put the following rule in your application's CSS file:

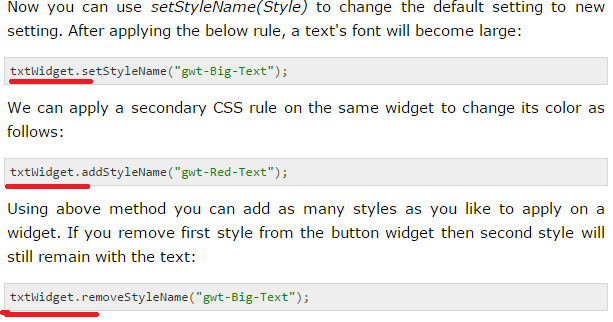


1. Now Let us see the Java Code where I add the above mention id for a widget/Element,

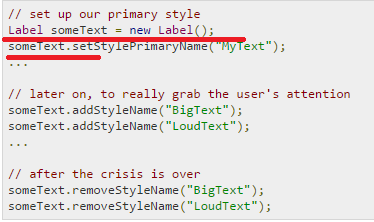
Note: 🡺 Needed the clear Information on the Following Classes

1. EntryPoint
2. Button
3. TextBox
4. Label
5. RootPanel
6. DOM
7. Now let’s see few methods/API related to the CSS Styling
8. For example,

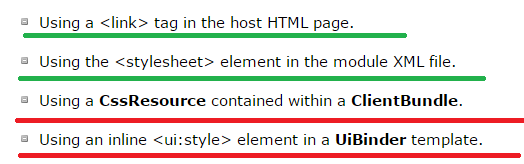


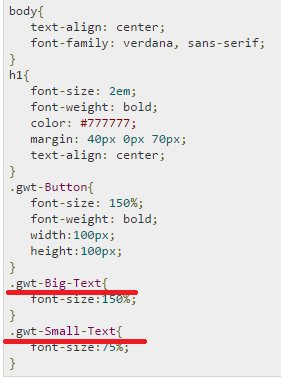


## Primary & Secondary Styles

1. **By default, the *primary style* name of a widget will be the default style name for its widget class for example *gwt-Button* for Button widgets. When we add and remove style names using AddStyleName() method, those styles are called secondary styles.**
2. The final appearance of a widget is determined by the sum of all the secondary styles added to it, plus its primary style.
3. 

## Associating CSS Files

1. There are multiple approaches for associating CSS files with your module. 
2. Let’s Consider an example, ms 🡺Where In MY Css file I have added 2 new Style names namely .gwt-Big-Text and .gwt-Small-Text, let’s see in which element this attributes gets added up, In the following code





Let us have following content of Java file **src/com.tutorialspoint/HelloWorld.java** which will take care of adding two buttons in HTML and will apply custom CSS style.



Note 🡺 for a clear understanding on above program, I need to do practically,

Actually 44th point ad above screen shot has some link, look at those two in first one DOM class is used and Here RootPanel class is used what is the difference between this two.

# GWT - Basic Widgets

1. Every user interface considers the following three main aspects

* **UI elements**
* **Layouts**
* **Behavior**

**UI elements**:

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

**Layouts:**

1. They define how UI elements should be organized on the screen and provide a final look and feel to the GUI

**Behavior:**

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

## GWT UI Elements: