

Introduction

The **Image** widget displays an image at a given URL. The image widget can be in 'unclipped' mode *thedefaultmode* or 'clipped' mode. In clipped mode, a viewport is overlaid on top of the image so that a subset of the image will be displayed. In unclipped mode, there is no viewport - the entire image will be visible. Methods will operate differently depending on the mode that the image is in. These differences are detailed in the documentation for each method.

Class declaration

Following is the declaration for **com.google.gwt.user.client.ui.Image** class:

```
public class Image
    extends Widget
    implements SourcesLoadEvents, HasLoadHandlers,
        HasErrorHandlers, SourcesClickEvents,
        HasClickHandlers, HasAllMouseHandlers,
        SourcesMouseEvents
```

CSS style rules

Following default CSS Style rule will be applied to all the Image widget. You can override it as per your requirements.

```
.gwt-Image { }
```

Class constructors

S.N.	Constructor & Description
1	Image Creates an empty image.
2	protected ImageElementelement This constructor may be used by subclasses to explicitly use an existing element.
3	Imagejava.lang.Stringurl Creates an image with the specified url.
4	Imagejava.lang.Stringhtml, intleft, inttop, intwidth, intheight Creates a clipped image with a specified URL and visibility rectangle.

Class methods

S.N.	Function name & Description
------	-----------------------------

1	void addClickListener <i>ClickListenerlistener</i>	Adds a listener interface to receive click events.
2	void addLoadListener <i>LoadListenerlistener</i>	Adds a listener interface to receive load events.
3	void addMouseListener <i>MouseListenerlistener</i>	Adds a listener interface to receive mouse events.
4	void addMouseWheelListener <i>MouseWheelListenerlistener</i>	Gets this widget's parent panel.
5	int getHeight	Gets the height of the image.
6	int getOriginLeft	Gets the horizontal co-ordinate of the upper-left vertex of the image's visibility rectangle.
7	int getOriginTop	Gets the vertical co-ordinate of the upper-left vertex of the image's visibility rectangle.
8	java.lang.String getUrl	Gets the URL of the image.
9	int getWidth	Gets the width of the image.
10	void onBrowserEvent <i>Eventevent</i>	Removes a previously added listener interface.
11	static void prefetch <i>java. lang. Stringurl</i>	Causes the browser to pre-fetch the image at a given URL.
12	void removeClickListener <i>ClickListenerlistener</i>	This method is called immediately before a widget will be detached from the browser's

document.

13

void removeLoadListener*LoadListenerlistener*

Removes a previously added listener interface.

14

void removeMouseListener*MouseListenerlistener*

Removes a previously added listener interface.

15

void removeMouseWheelListener*MouseWheelListenerlistener*

Removes a previously added listener interface.

16

void setUrl*java. lang. Stringurl*

Sets the URL of the image to be displayed.

17

void setUrlAndVisibleRect*java. lang. Stringurl, intleft, inttop, intwidth, intheight*

Sets the url and the visibility rectangle for the image at the same time.

18

void setVisibleRect*intleft, inttop, intwidth, intheight*

Sets the visibility rectangle of an image.

19

static Image wrap*Elementelement*

Creates a Image widget that wraps an existing element.

Methods inherited

This class inherits methods from the following classes:

- com.google.gwt.user.client.ui.UIObject
- com.google.gwt.user.client.ui.Widget
- java.lang.Object

Image Widget Example

This example will take you through simple steps to show usage of a Image Widget in GWT. Follow the following steps to update the GWT application we created in *GWT - Create Application* chapter:

Step	Description
------	-------------

- | | |
|---|--|
| 1 | Create a project with a name <i>HelloWorld</i> under a package <i>com.tutorialspoint</i> as explained in the <i>GWT - Create Application</i> chapter. |
| 2 | Modify <i>HelloWorld.gwt.xml</i> , <i>HelloWorld.css</i> , <i>HelloWorld.html</i> and <i>HelloWorld.java</i> as explained below. Keep rest of the files unchanged. |

3 Compile and run the application to verify the result of the implemented logic.

Following is the content of the modified module descriptor **src/com.tutorialspoint/HelloWorld.gwt.xml**.

```
<?xml version="1.0" encoding="UTF-8"?>
<module rename-to='helloworld'>
  <!-- Inherit the core Web Toolkit stuff. -->
  <inherits name='com.google.gwt.user.User' />

  <!-- Inherit the default GWT style sheet. -->
  <inherits name='com.google.gwt.user.theme.clean.Clean' />

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

  <!-- Specify the paths for translatable code -->
  <source path='client' />
  <source path='shared' />

</module>
```

Following is the content of the modified Style Sheet file **war/HelloWorld.css**.

```
body{
    text-align: center;
    font-family: verdana, sans-serif;
}
h1{
    font-size: 2em;
    font-weight: bold;
    color: #777777;
    margin: 40px 0px 70px;
    text-align: center;
}
```

Following is the content of the modified HTML host file **war/HelloWorld.html**.

```
<html>
<head>
<title>Hello World</title>
  <link rel="stylesheet" href="HelloWorld.css"/>
  <script language="javascript" src="helloworld/helloworld.nocache.js">
  </script>
</head>
<body>

<h1>Image Widget Demonstration</h1>
<div ></div>

</body>
</html>
```

Let us have following content of Java file **src/com.tutorialspoint/HelloWorld.java** which will demonstrate use of Image widget.

```
package com.tutorialspoint.client;

import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.user.client.ui.Image;
import com.google.gwt.user.client.ui.RootPanel;
import com.google.gwt.user.client.ui.VerticalPanel;

public class HelloWorld implements EntryPoint {
    public void onModuleLoad() {
```

```
// Create a Image widget
Image image = new Image();

//set image source
image.setUrl("http://www.tutorialspoint.com/images/gwt-mini.png");

// Add image to the root panel.
VerticalPanel panel = new VerticalPanel();
panel.add(image);

RootPanel.get("gwtContainer").add(panel);
}
```

Once you are ready with all the changes done, let us compile and run the application in development mode as we did in [GWT - Create Application](#) chapter. If everything is fine with your application, this will produce following result:

