



## **Developers' Guide**

### **Framework**

**Version 2.0.3.1**

**20<sup>th</sup> January 2012**

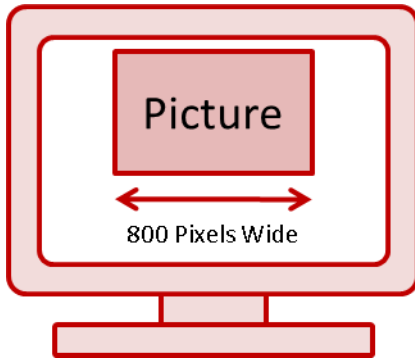
© 2011 51Degrees.mobi Limited. All rights reserved.

The copyright in and title to the document "Developers' Guide – Framework" belongs to 51Degrees.mobi Limited. No part of it whatsoever may be reproduced in any form without the prior authority of 51Degrees.mobi Limited and/or any original source as appropriate. Any agreed copy or extract must be marked with all proprietary notices which appear on the original and will be subject to the requirement that you will acknowledge on the face of each part of the reproduced material that it belongs to 51Degrees.mobi Limited.

## 5 Images

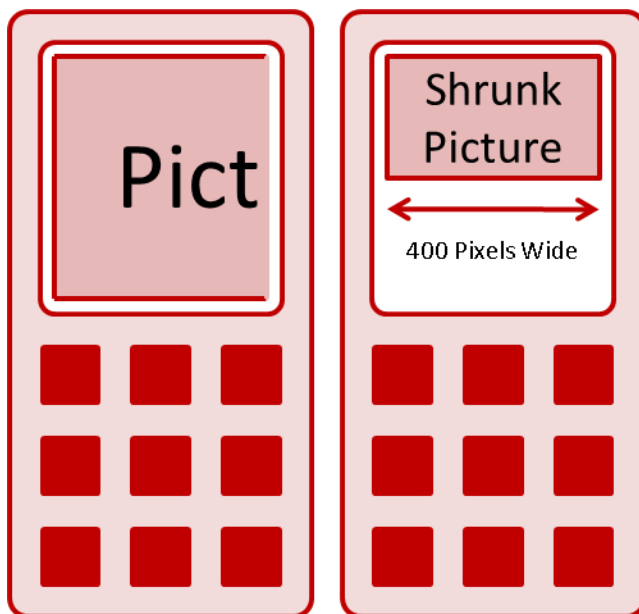
### 5.1 Concept

When designing traditional web sites, designers and developers will typically lay out pages and images specifying measurements in pixels. Consider a page with a width of 800 pixels. An image that is 100% the width of the page will have been created 800 pixels wide. If the screen is wider than 800 pixels then a border will be present either side of the main page display area. See Figure 1.



**Figure 1 - Picture sized for a traditional web site**

If such an approach were to be used with a mobile web application the page dimensions may become too wide for the screen of the mobile device. Even the most advanced “retina” displays do not support a portrait width of more than 800 pixels, and if they did the page elements would become too small to read. The user will either need to use horizontal scrolling to view the content, or will need to use the zoom feature of the mobile browser if available. See Figure 2.



**Figure 2 - A mobile device with screen width of 400 pixels can display 50% of the image or shrink it by 50%**

An alternative approach would be to specify the dimensions of the image as a percentage of the containing element or screen size. This approach will resolve the display issue described earlier. However a new problem is created. Without 51Degrees.mobi the web server knows nothing about the mobile devices display and will

send the full size image to the mobile device. Continuing the earlier example, a picture of 800 pixels will be sent from the web server even though the mobile device will only require an image of 400 pixels width. Assuming the image retains the same aspect ratio when shrunk; approximately 75% of the data sent from the web server to the mobile device will be wasted representing a significant performance and possible cost overhead. See Figure 3.

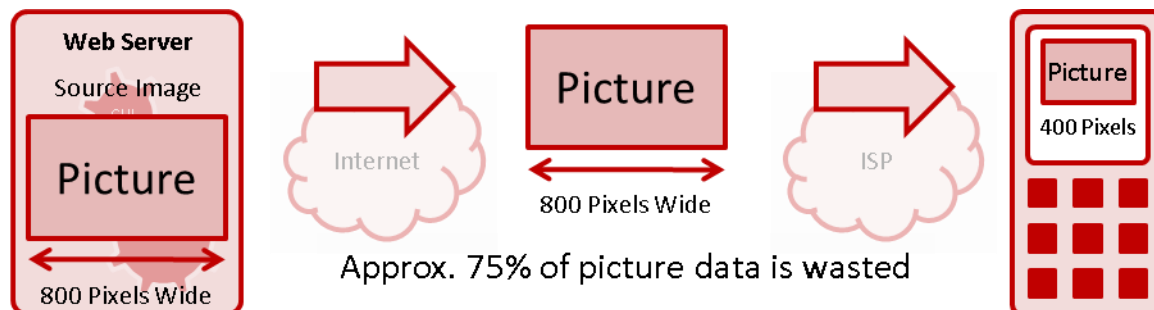


Figure 3 - Web servers sends 75% more data than required by the mobile device

51Degrees.mobi enables images to be sized using percentage measurements, or any other measurement, without wasting precious data. A new component called “Image Handler” is added to the web server to dynamically create an image specifically for the requesting device and the element in which the image will be displayed using one or more image sources. See Figure 4.

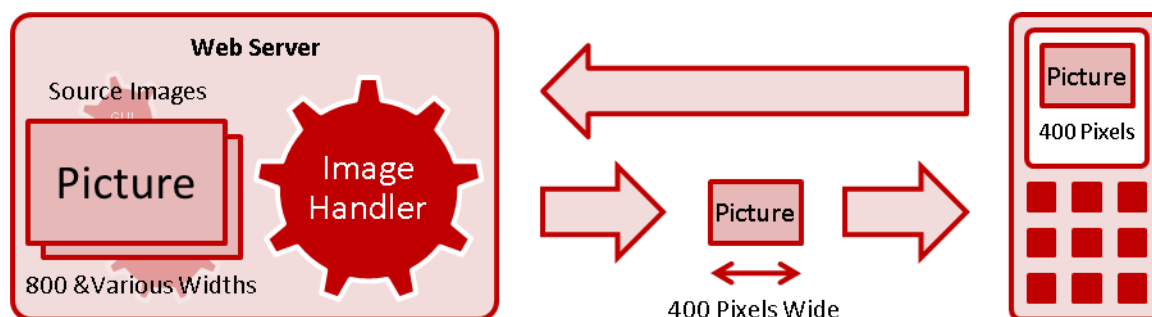


Figure 4 - Image Handler

The **Error! Reference source not found.** is enabled by default when creating a 51Degrees.mobi mobile web application. See the **Error! Reference source not found.** section of the Web.Config chapter for configuration details.

Different source images in various formats and dimensions can be provided to the image control. For example a large finger sized icon designed for a high resolution touch screen mobile device may not shrink well to 16 pixels wide. In this scenario several image sources may be needed for each of the approximate sizes required.

Image dimensions can be calculated by JavaScript executing on the mobile device, or before the page is sent by the server. Client side JavaScript calculation of dimensions will produce the best results.

Two methods have been added to 51Degrees.mobi image controls derived from the standard ASP.NET controls to support this feature.

1. The CalculateSizeMode property of image controls is used to determine the method used to calculate the image dimensions.

- Multiple images sources of varying sizes can be provided in the `AltImageList` collection of an image control.

This chapter explains how to use these two methods to optimise mobile web applications.

## 5.2 Calculating Image Size

The `CalculateSizeMode` property is used to control the method used to calculate the size of the image. It can be set to the value `Server`, or one of three `Client` values. Each setting is explained in the following section.

### 5.2.1 Server

Because 51Degrees.mobi Framework sits on top of the Foundation component the web server has an accurate understanding of mobile device capabilities. One of the enhanced capabilities of which the Web Server is aware is the dimensions of the mobile device screen. Therefore the values returned from the `ScreenPixelsWidth` and `ScreenPixelsHeight` properties of the `HttpBaseCapabilities` class can be relied on and used to set the width or height dimension of an image in code at runtime.

The following code could be used to provide an image the width of the screen at runtime using the `CalculateSizeMode` value of `Server`.

ASP.NET

```
<mob:Image id="Image1" runat="server" CalculateSizeMode="Server"
ImageUrl="~/Image.png"/>
```

C#

```
protected void Page_PreRender(object sender, EventArgs e)
{
    Image1.Width = Request.Browser.ScreenPixelsWidth;
}
```

The image size calculation happens on the web server during the `Page_PreRender` event and the resulting size in pixels is contained in the HTML page sent to the mobile device. As the page is loaded on the mobile device an image of the precise pixel width calculated in the `Page_PreRender` event earlier will be fetched and the Image Handler will provide a precisely sized image using the most suitable source image. See Figure 5 where a 400 pixel wide screen and image will be used.

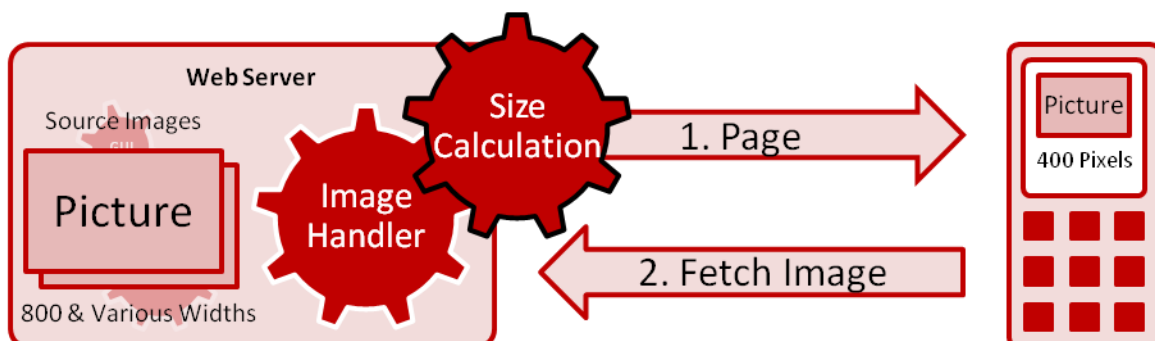


Figure 5 - Image size calculated by the web server

## 5.2.2 Client

The image size can also be calculated on the mobile device using JavaScript. This approach has the advantage of knowing the exact size in pixels of the image element and its containers. As a result it is often more accurate and suitable for situations where it is not possible or practical to calculate the image size on the server.

Three different modes of calculation are available. `ClientWidth` and `ClientHeight` calculate the size using the width or height values of the containing parent element. The aspect ratio of the image is retained. `ClientBoth` can be used to calculate both the width and height of the image but the aspect ratio will not be retained.

Each mode utilises a small piece of JavaScript that will execute, as soon as the page has been loaded, to calculate the size of the image and its corresponding URL for the Image Handler.

The following code has been adapted from the previous example for use with `ClientWidth` mode. Notice the width is specified as 100%.

ASP.NET

```
<mob:Image id="Image1" runat="server" CalculateSizeMode="ClientWidth"
width="100%" ImageUrl="~/Image.png"/>
```

The web server requires no additional code and will send a page to the mobile device with any explicit size information for the image in pixels. The JavaScript will execute on the mobile device as the page loads calculating the width of the image and fetching the image from the Image Handler. See Figure 6.

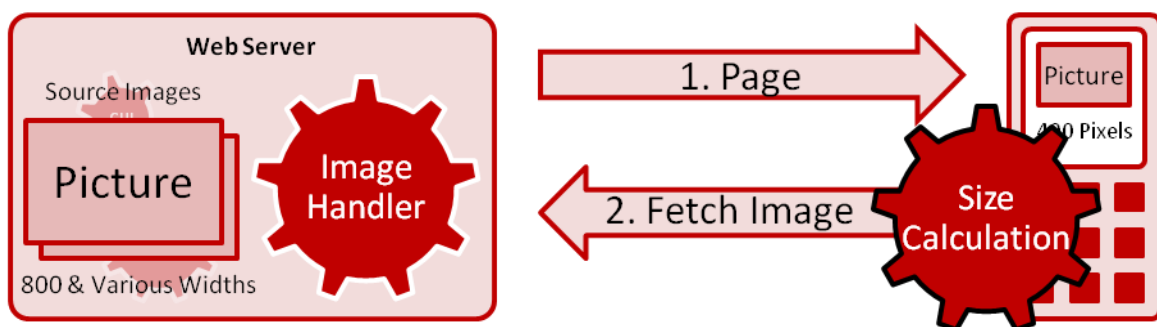


Figure 6 - Image size calculated on the mobile device

The client mode JavaScript size calculation will first check the image element to determine if a size property has been provided. For example; `width="50px"`. If so then this value will be converted to pixels and used. If not then the parent containers will be checked until a positive size in pixels can be obtained.

In order for the dimensions of an image to be calculated on the mobile device, JavaScript with Document Object Model (DOM) needs to be supported by the browser. Most mobile devices purchased in 2010 will meet this requirement. Older mobile devices may not. If a Client mode is used with devices that do not support JavaScript DOM then `Server` mode will be used automatically. An image no larger than either the screen size or the largest image source available will be returned if an explicit size has not been provided. For this reason a mobile web application that is likely to be used by older mobile devices should consider using `Server` mode for image sizing to ensure a consistent experience.

### 5.2.3 Summary

The following table describes key details of the mode along with an example.

Mode	Key Details	Example	Example Diagram
<b>ClientWidth</b>	<ol style="list-style-type: none"> <li>The width of the image will be calculated using JavaScript on the mobile device once the page has been loaded.</li> <li>If an explicit image width has been provided this will be converted to pixels and used as the width measurement. If no width has been provided the width of the parent container will be used.</li> <li>An image of the required width will be fetched from the web server for display.</li> <li>All height information is ignored and the aspect ratio of the image maintained.</li> <li>If the mobile device does not support JavaScript with DOM Server mode will be used.</li> </ol>	<p>Consider the following page design.</p> <ul style="list-style-type: none"> <li>A picture and its caption should fill 50% of the screen width.</li> <li>The screen is 400 pixels wide.</li> <li>Text should flow around the pictures container.</li> <li>An Image and Label control are used to display the picture and its caption. No size information is defined and the controls are placed inside a Panel control.</li> <li>The containing Panel control has a width of 50%.</li> <li>The Image control is set to mode ClientWidth.</li> </ul> <p>When the JavaScript calculates the width of the image the container's width will be used and an image of 200 pixels width requested from the server.</p> <p>Note: The containing panel will need to use display: inline-block CSS styling to appear in line with the text on the right.</p>	

Mode	Key Details	Example	Example Diagram
<p><b>ClientHeight</b></p>	<ol style="list-style-type: none"> <li>The height of the image will be calculated using JavaScript on the mobile device once the page has been loaded.</li> <li>If an explicit image height has been provided this will be converted to pixels and used as the height measurement. If no height has been provided the height of the parent container will be used.</li> <li>An image of the required height will be fetched from the web server for display.</li> <li>If the mobile device does not support JavaScript with DOM Server mode will be used.</li> </ol>	<p>Consider the following page design.</p> <ul style="list-style-type: none"> <li>A list box of values is displayed with an icon to its right.</li> <li>A ListBox and ImageButton control are used to display the two page elements. The ListBox is set to 80% of the screen width. The ImageButton's height is set to 100%.</li> <li>The two controls are placed next to each other within a Panel that does not have any size information provided.</li> <li>The ImageButton is set to use ClientHeight mode.</li> </ul> <p>When the page is processed by the mobile device the largest item within the container will determine its height. In this example the ListBox is the largest item and the Panel container will adapt to the ListBox height. When the JavaScript calculates the height of the icon image it will use the height of the container in pixels and request an appropriately sized image from the web server. The precise height will vary as each mobile device platform implements List Box element differently. However the Icon will be the same height as the List Box.</p> <p>Note: Both the List Box and the Icon should use the CSS style attribute display:inline to ensure they appear next</p>	<p>The diagram illustrates a container (dashed red box) containing two elements: a 'List Box' and an 'Icon'. The container's height is determined by the tallest element, the List Box. A vertical double-headed arrow on the right side of the container is labeled 'Height 100%', indicating that the container's height is set to the height of its tallest child element.</p>

Mode	Key Details	Example	Example Diagram
		<p>to each other. The List Box width needs to be 80% to ensure there is sufficient room for the Icon to appear next to it.</p>	
<p><b>ClientBoth</b></p>	<ol style="list-style-type: none"> <li>1. The width and height of the image will be calculated using JavaScript on the mobile device once the page has been loaded.</li> <li>2. The dimensions of the image, or container if not provided, will be converted to pixels and passed to the web server to provide an appropriate image. The aspect ratio may not be contained and this mode should be used with caution.</li> <li>3. If the mobile device does not support JavaScript with DOM Server mode will be used.</li> </ol>	<p>The width and height properties of an image contained on a page are set to 50% and 80% of the container respectively. The width and height dimensions will be calculated based on the screen size and the mobile device will request an image of a fixed width and height.</p>	

Mode	Key Details	Example	Example Diagram
<p><b>Server</b></p>	<ol style="list-style-type: none"> <li>The server will calculate the size of the image if at least one dimension in pixels has been provided.</li> <li>This mode will work on any mobile device, but relies on the developer calculating at least one of the image dimensions in code.</li> <li>An image will not be returned that is larger than the screen.</li> </ol>	<p>An image control with ID Image1 should fill 50% of the screen width to enable text to appear to its right. In Server mode the developer would need to set the width of the image explicitly from code at runtime using a line similar to the following:</p> <pre>Image1.Width = Request.Browser.ScreenPixelWidth / 2;</pre> <p>Note: The Image1 control will need to use the display: inline-block CSS style attribute to appear to the left of the text.</p>	
<p><b>None</b></p>	<p>Normal ASP.NET image control behaviour applies.</p>	<p>Not Applicable</p>	<p>Not Applicable</p>

## 5.3 Image Sources

Standard image controls contain a single property named `ImageUrl` to specify the source of the image. `ImageUrl` will be set to a value which references a file or a URL. 51Degrees.mobi image controls support this property and provide an additional collection enabling the developer to specify multiple source images. This new collection can be used to specify multiple source images with different dimensions and image formats.

### 5.3.1 Formats

Image sources can be provided in PNG, JPG and GIF format. PNG and JPG images sources are preferable to GIF to avoid problems with dithering when scaling GIF format images. GIF image dithering will be particularly noticeable when using 51Degrees.mobi in medium trust mode as the Image Handler's ability to manipulate the image data is restricted. Other formats may result in unpredictable behaviour.

### 5.3.2 URLs

All URL formats including both HTTP, absolute and relative file names are supported by the `AltImages` property. As the web server needs to be able to retrieve the image to determine its size and characteristics, it's essential the web server can retrieve the URL provided. This may be a consideration for mobile web applications that host images on different web servers and hostnames and have firewalls limiting the URLs to which the web server is allowed access.

### 5.3.3 Why?

Different source images can be useful when a wide range of image sizes may be required by different mobile devices. Consider an iPhone 3 and 4 where the resolution of the iPhone 4 screen is twice that of the 3 but use the same physical dimensions. An image occupying the full screen width of an iPhone 4 will be twice as wide when measured in pixels as the equivalent image on an iPhone 3. Some images will scale well in this situation. See Figure 7 where the image of the balloons can be shrunk to 50% of its original width without losing clarity.



Full Size Image



50% Reduction

**Figure 7 – The balloon image scales down without loss of quality**

#### 5.3.3.1 Icons

However consider a situation where an image will be used as an icon. Touch screen mobile devices will require icons that are at least 1 cm wide. Mobile devices that use a joystick, click wheel, or track ball can use smaller icons. Considering all these situations an image may be required between 16 and 72 pixels in width. An icon of 72 pixels wide is unlikely to reduce well to 16 pixels in width. Conversely an icon of 16 pixels width will not scale well to 72 pixels width.

The top row in Table 1 shows 6 different sized icons specifically designed for the image size at which they're displayed. Notice how the 16 and 24 pixel icons are two dimensional and do not include a wand. The 32 and 48 pixels wide icons start to introduce a wand and 3 dimensional depth using grey shadow. Finally the 72 and 128 pixel wide icons introduce a darker shadow.

The second row shows 5 icon sizes scaled down from the 128 pixel wide version. Notice the 16 and 32 pixel width versions are harder to distinguish when compared to the customised versions in the rows above.












	Image Size					
Source	16x16	24x24	32x32	48x48	72x72	128x128
Customised						
Shrunk						

Table 1 - Icons

We recommend the use of an icon library that provides icon sizes between 16 and at least 72 pixels width in PNG, JPG and GIF format. All these images should be provided to an Image control that is displaying an icon via the AltImagesList property. The 51Degrees.mobi Image Handler will automatically determine the best source image to use for each request.

The following code provides an example of the ASP.NET that would be used.

ASP.NET

```
<mob:ImageButton id="ImageIcon" runat="server" AlternateText="Icon"
CalculateSizeMode="Server">
  <mob:AltImage ImageUrl="~/Images/Icon256.gif" />
  <mob:AltImage ImageUrl="~/Images/Icon128.gif" />
  <mob:AltImage ImageUrl="~/Images/Icon72.gif" />
  <mob:AltImage ImageUrl="~/Images/Icon48.gif" />
  <mob:AltImage ImageUrl="~/Images/Icon32.gif" />
  <mob:AltImage ImageUrl="~/Images/Icon16.gif" />
  <mob:AltImage ImageUrl="~/Images/Icon256.jpg" />
  <mob:AltImage ImageUrl="~/Images/Icon128.jpg" />
  <mob:AltImage ImageUrl="~/Images/Icon72.jpg" />
  <mob:AltImage ImageUrl="~/Images/Icon48.jpg" />
  <mob:AltImage ImageUrl="~/Images/Icon32.jpg" />
  <mob:AltImage ImageUrl="~/Images/Icon16.jpg" />
  <mob:AltImage ImageUrl="~/Images/Icon256.png" />
  <mob:AltImage ImageUrl="~/Images/Icon128.png" />
  <mob:AltImage ImageUrl="~/Images/Icon72.png" />
  <mob:AltImage ImageUrl="~/Images/Icon48.png" />
```

```
<mob:AltImage ImageUrl="~/Images/Icon32.png" />
<mob:AltImage ImageUrl="~/Images/Icon16.png" />
</mob:Image>
```

In situations where an image size has not been provided the ImageUrl image source will be returned. If no ImageUrl is specified then the smallest image from the list will be returned. This can occur when client mode is specified but the mobile device does not support JavaScript and DOM.

### 5.3.3.2 Image Buttons

Where possible image buttons should be avoided to reduce page weight and improve performance. Consider styling image buttons via the style properties of the ImageButton control.

However in some situations they cannot be avoided. For example; where the user’s attention should be drawn to the button and its function because it enables a critical process such as payment or confirmation. Often such buttons will contain text to provide instructions and help the user. The readability of text can reduce as the button reduces in size and can become unreadable. See Figure 8. In these situations it may be desirable to provide image buttons specifically designed for smaller screen sizes. In Figure 8 the word “Details” has been removed to improve clarity.

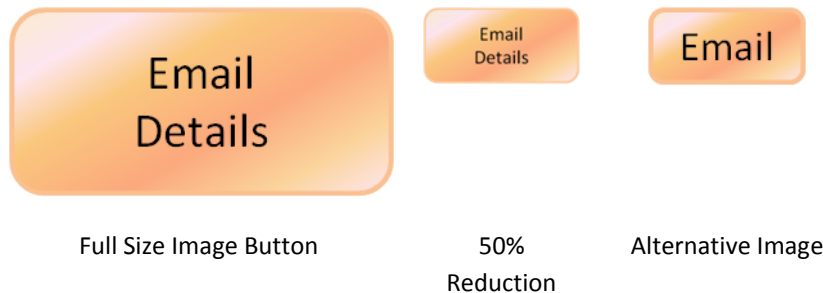


Figure 8 - Image button with text

### 5.3.3.3 Important Words

Some images may contain important words that must be clearly visible such as copyright notices or embedded captions. These words may become unreadable as the image is reduced in size to fit on smaller displays. Figure 9 shows a picture that has been reduced in size by 50% and a copyright notice that is no longer visible. An alternative image at 50% reduction is provided with a readable copyright notice which remains visible.



Figure 9 - Copyright notice

### 5.3.4 Image Controls

The following controls implement the `CalculateSizeMode` and `AltImageList` properties described in this chapter.

- `AdRotator`
- `Hyperlink`
- `Image`
- `ImageButton`