# Custom resource paths

> ⚠ **Outdated example**
>
> These examples seem to be outdated. Recommended usage for Wicket 1.1, 1.2 and 1.3 can be found on another page .

> ⚠ **Further examples**
>
> An example of custom resource loading can be found in the `wicket-examples`, package `wicket.examples.customresourceloading`.

By default, Wicket loads the markup files from the Java packages. This is a good default, as it makes it possible to package components, including Pages, Panels and Borders, in a jar file, and they will still work without you having to do anything special for it!

However, there might be cases when you want to customize the lookup paths that Wicket uses for locating markup.

As an example, lets pretend we want to load the HelloWorld example from the root of the webapp directory. The place to configure this is the application:

```
public class HelloWorldApplication extends WicketExampleApplication
{
    /**
     * Constructor.
     */
    public HelloWorldApplication()
    {
        getPages().setHomePage(HelloWorld.class);
    }

    /**
     * @see wicket.Application#init()
     */
    protected void init()
    {
            try
        {
            URL webroot = getWicketServlet().getServletContext().getResource("/");
            Path resourcePath = new Path();
            resourcePath.add(new Folder(new URI(webroot.toString())));
            getSettings().setResourcePath(resourcePath);
        }
        catch (MalformedURLException e)
        {
            throw new RuntimeException(e);
        }
        catch (URISyntaxException e)
        {
            throw new RuntimeException(e);
        }
    }
}
```

Note from the above example that we overloaded method `init()`; If you want to use WicketServlet, e.g. because you want to access the ServletContext object, you should do this in the init() method, because the WicketServlet object is not set yet in the constructor.

Also note that a Path object can consist of multiple Folder object, thus making it possible to use a mixed, layered approach.

Now, the final thing we have to do to get this working, is to place HelloWorld.html in directory <webapp root dir>/wicket/examples/helloworld. The directory structure has to be the same as the package structure your markup component is in.

For even more advanced uses, it is possible to provide a custom resource loading mechanism. But that is another chapter 🙂 Here is that chapter to detail a way to control where HTML files are loaded from.

**Note that currently there is an issue with this depending too much on java.io.File instead of urls like it should do. Do not use this option for production systems (Wicket 1.0-rc3)**

setResourcePath seems to be gone from 1.0 final. The following does the same as above but works in 1.0 final:

```
protected void init()
{
  String root = getWicketServlet().getServletContext().getRealPath("/");
  getSettings().addResourceFolder(root);
}
```

Remark:
Don't even need to get the servlet context yourself, as that is what we try first for relative paths.

I have set the resource location for the html pages as below (html dir in the webapp root). The method addResourceFolder(String resourceFolder) starts from the webapp root by default, so you don't have to retrieve the servletContext at all. The pages still have to be placed in the same directory as their corresponding classes.
(Paul Voors)

```
protected void init()
{
  getSettings().addResourceFolder("/html");
}
```