InvokingJsfPagesWithStandardUrls

It is often desired to include a URL in a standard HTML page which, when clicked, invokes a JSF page passing query parameters from the URL to the backing bean for the page.

Accessing request params from a JSP page

The EL expression #{param.someName} or #{param['someName']} can be used in a JSP page (within a JSF tag's attribute) to reference the query param value directly.

The "param" object is just one of the implicit objects accessable via EL expressions; the full list is outlined in Section 5.3.1.2 of the JSF Specification.

Alternatively,

```
#{facesContext.externalContext.requestParameterMap.someName}
```

should also work.

Injecting params into a JSF managed bean

An EL expression can be used to "inject" the param value into a request-scope managed bean:

```
<managed-bean>
  <managed-bean-name>myBean</managed-bean-name>
  <managed-bean-class>example.MyBean</managed-bean-class>
  <managed-bean-scope>request</managed-bean-scope>
  <managed-property>
    <property-name>somePropertyName</property-name>
    <value>#{param.someQueryParamName}</value>
  </managed-property>
</managed-bean>
```

When the jsp page refers to #{myBean}, the bean will be created "on demand", and initialised as defined in the managed bean mapping.

Note that the bean *must* be request-scope however, as it's forbidden to initialise a session or app scoped bean with a request-scoped value (for obvious reasons).

Looking up request params from the bean

Accessing the request can be done from the bean:

```
var params = FacesContext.getCurrentInstance().getExternalContext().getRequestParameterMap();
String someValue = params.get("someName");
```

Use PrettyFaces (by OcpSoft)

For bookmarking, dynamic parameters, seo, and simplified navigation: this simple extension for JSF/JSF2 uses a Servlet Filter to intercept all requests. Request parameters can be injected directly into JSF managed beans via a minimalist configuration, managed bean methods can be invoked on each page request. "PrettyFaces URLRewriteFilter - Page Actions - and Navigation for JSF" is primarily designed to convert JSF urls: "/faces/page.xhtml?id=345" into parameterized, bookmarkable URLs: "/page/345", such as done by wordpress, or other client-facing web-applications.

PrettyFaces also offers simplified navigation, linking, and URL request-parameter validation.

Use Apache Shale

The Apache Shale framework for JSF has a ViewController class that provides a straightforward mechanism for dealing with GET requests as well as POST requests. On a GET, the prerender() method is called once JSF notices that there is no view state to be restored. This is a convenient place to access the request parameters, using one of the techniques above. (If you use the convenience base class AbstractViewController, there is also a convenience getRequestParameter() method available.) http://shale.apache.org/

Also see Shale remoting for executing method binding expressions specified by URLs. http://shale.apache.org/features-remoting.html

Using a custom servlet

A custom servlet can be used to receive the request, initialise any variables needed, then forward to the JSF page.

However, JSF requires that accesses pass through its own servlet in order to set up the JSF framework to handle the current request. Standard Servlet behaviour for forwards, however, does **not** run any servlets mapped to the forwarded-to URL. Your custom servlet therefore needs to duplicate the setup behaviour of the standard JSF class javax.faces.webapp.FacesServlet.

Here's the necessary code:

```
    LifecycleFactory lFactory = (LifecycleFactory)
        FactoryFinder.getFactory(FactoryFinder.LIFECYCLE_FACTORY);
    Lifecycle lifecycle =
        lFactory.getLifecycle(LifecycleFactory.DEFAULT_LIFECYCLE);
    FacesContextFactory fcFactory = (FacesContextFactory)
        FactoryFinder.getFactory(FactoryFinder.FACES_CONTEXT_FACTORY);
    FacesContext facesContext =
        fcFactory.getFacesContext(getServletContext(), request, response,
            lifecycle);
    Application application = facesContext.getApplication();
    ViewHandler viewHandler = application.getViewHandler();

    // specify what page you want to pretend to "come from"; normally
    // an empty string is ok
    String viewId = "";
    UIViewRoot view = viewHandler.createView(facesContext, viewId);
    facesContext.setViewRoot(view);

    // put your initialization stuff here, eg
    //    ExternalContext externalContext = facesContext.getExternalContext();
    //    externalContext.getRequestMap().put("someKey", "someValue");
    // or
    //    MyBean myBean = (MyBean) application.getVariableResolver().
    //       resolveVariable(facesContext, "myBean");
    //    myBean.setFoo("bar");
    // or
    //    MyBean myBean = (MyBean) application.createValueBinding("#{myBean}").
    //       getValue(facesContext);
    //    myBean.setFoo("bar");

    // specify what outcome value you want to use for navigation
    String outcome="SUCCESS";

    NavigationHandler navigationHandler = application.getNavigationHandler();
    navigationHandler.handleNavigation(facesContext, null, outcome);
    lifecycle.render(facesContext);
```

The Apache Tobago library already has a class NonFacesRequestServlet that does this (in fact the code above is copied from that class with minor modifications). See: http://svn.apache.org/viewvc/myfaces/tobago/trunk/core/src/main/java/org/apache/myfaces/tobago/servlet/NonFacesRequestServlet.java?view=markup

The're are some differences between the myFaces implementation & the Sun RI. There is a discussion on these issues on the myFaces User group. Link to the thread:

The subject title is : "Re: myFaces - servlet redirect" http://www.mail-archive.com/users%40myfaces.apache.org/msg15139.html

Making a fake postback

Users of facelets have another method available to them as described in the blog Faking a Postback with JSF + Facelets. This method uses an extended FaceletViewHandler which allows you to craft URLs such that they are indistinguishable from regular form submissions. Feedback on how this ViewHandler works with MyFaces is welcome via the blog comments.