Preserving JSF Request Parameters and REST URLs

By Roger Keays, 13 February 2013

Have you ever used a URL pattern something like this?

```
http://localhost/app/widgets/WidgetEditor.xhtml?id=300
```

or perhaps this?

```
http://localhost/app/widgets/300/edit
```

Well, as a JSF developer you've probably already realized a little problem. **How do you remember the id** from request to request? There are a few simple solutions.

1. **Put the entity in the session scope** when you first fetch it and forget the id. This method creates a user session for every request and doesn't support multiple browser tabs.
2. **Put the entity in the view scope** when you first fetch it. This method supports multiple tabs but still create a session for every user.
3. Use plain old **HTML hidden input field** to remember the id:

   ```
   <input type="hidden" name="id" value="${param.id}"/>
   ```

   This works just fine until you start adding other request parameters like ? `type=blue&currency=AUD` and forget to add the hidden fields to retain them. It's also a weird mix of JSF and HTML.
4. **Add an f:param** to the command button. This also works fine as long as you remember to add the parameter to every command button:

   ```
   <h:commandButton action="${bean.save}" value="Save">
     <f:param name="id" value="${param.id}"/>
   </h:commandButton>
   ```

But there could be a better way.

Your **URLs inevitably encode some sort of state**, even if that state is simply "what page am I on". Out of the box, JSF got this right. All forms are posted back to the original View ID. But this doesn't tell the whole story because **the View ID is not the Request URL** and we lose state that is encoded in request parameters or RESTful URL schemes like `/app/widgets/300/edit`.

We can fix this fairly easily.

All we need to do is **post back to the original Request URL** instead of the View ID and we're done. The forms come out looking like:

```
<form action="/app/widgets/WidgetEditor.xhtml?id=300&type=blue">..</form>
```

or

```
<form action="/app/widgets/300/edit">..</form>
```

instead of

```
<form action="/app/widgets/WidgetEditor.xhtml"/>..</form>
```

To do this in JSF you need to implement a custom `ViewHandler#getActionURL()` method that looks like this:

```
/**
 * We always post back to the original Request URL, not the viewID
 * since we sometimes encode state in the Request URL such as object id,
 * page number, etc.
 */
@Override
public String getActionURL(FacesContext faces, String viewID) {
    HttpServletRequest request = (HttpServletRequest)
            faces.getExternalContext().getRequest();

    // remaining on the same view keeps URL state
    String requestViewID = request.getRequestURI().substring(
            request.getContextPath().length());
    if (requestViewID.equals(viewID)) {

        // keep RESTful URLs and query strings
        String action = (String) request.getAttribute(
                RequestDispatcher.FORWARD_REQUEST_URI);
        if (action == null) {
            action = request.getRequestURI();
        }
        if (request.getQueryString() != null) {
```

```
            return action + "?" + request.getQueryString();
        } else {
            return action;
        }
    } else {

        // moving to a new view drops old URL state
        return super.getActionURL(faces, viewID);
    }
}
```

Depending on your app, you might like to preserve request parameters across views also.

Your ViewHandler should extend ViewHandlerWrapper and be registered in **faces-config.xml** like so:

```
<application>
   <view-handler>com.example.MyViewHandler</view-handler>
</application>
```

That's it. You can stop worrying about losing the state in your URLs. If you deploy this code you will want to step through it in a debugger to confirm it works correctly for your JSF/webapp configuration.

Have fun.

About Roger Keays

Roger Keays is an artist, an engineer, and a student of life. He has no fixed addressand has leftfootprints on 40-something different countries around the world. Roger is addicted to surfing. His other interests are music, psychology, languages, the proper use of semicolons, and finding good food.