ActionContext

One of the handier feature in Webwork, in my limited experience, is the ActionContext object. This is a ThreaLocal object that is a wrapper around many common objects (Request, Response, ServletContext, etc.).

The thing that makes it so handy is that you can get a reference to it from any class at any time using:

```
ActionContext context = ActionContext.getContext();
```

Think about that for a moment... you can get it from **any** class at **any** time. This means that you can have a helper class called from an Action, and if it happens to need access to ServletContext (maybe it is writing a file and needs ServletContext to get a path to it), you can do so, you **do not** have to explicitly pass that information to the helper.

Two other important things about this is that (a) you should always be able to go to the same place to get the information you need, the ActionContext, as opposed to Struts where it is scattered across the Request, Response, ActionMapping, etc. objects, and (b) your Actions can be POJOs because there are no requirements for any method to have a specific signature. Think of the method signature of the Struts Action class:

```
ActionForward execute(ActionMapping, ActionForm, ServletRequest, ServletResponse)
```

In Webwork, no parameters are passed to the called method (whether the default execute() method or one you define in the Action mapping) because you can get at the same information via ActionContext within the method.

There is, however, one potential negative here. You have to be careful that you resist the temptation to write your business delegates to use this feature. In other words, generally-accepted "good" architecture says that your Actions should be very lightweight and should call on classes that perform your "business logic". You may be tempted, because it is so very easy, to grab request parameters directly from within those classes for instance. This is something you probably will want to avoid as it does, in a way, couple those business delegates to the webapp.

I said "in a way" there because the ActionContext actually abstracts away some of the web-centric objects. For instance, you can call getParameters() on ActionContext and you will get a Map back. This is of course a generic collection, not something web-centric, so in a sense that is "safe". Still, my feeling is this is one potential gotcha to at least be aware of.

That concern aside though, there is little doubt in my mind at least that the ActionContext, because of the fact that it is ThreadLocal, makes for cleaner, more generic code throughout your application, and that can only be a good thing!

Webwork API ActionContext reference:
http://www.opensymphony.com/webwork/api/com/opensymphony/xwork/ActionContext.html