StrutsAction1Planning

Struts Action Framework 1 Planning

This page collects ideas and discussion on what could be included in future releases of Struts Action 1. This page is a whiteboard, meaning no one has committed to anything and therefore its purpose is to share information and brainstorm.

1.3.x and beyond

The 1.3.x series utilizes Struts Chain as the default request processor and adds an "extends" attribute to all the configuration elements (a la Tiles). We would also like to add a Ant-style properties file to make variable substitutions within the XML elements. The substitution feature is supported by Spring and iBATIS, among others. Substitutions are a useful way to configure applications and reduce redundancy.

Experimental Members

To lay the groundwork for future changes, three "experimental" classes and interfaces are being added to the framework so that early adopters can experiment with using Struts Chain to develop applications. We do consider these members experimental, and they are subject to change, based on our experience using them with our own applications.

ActionCommand - A Chain Command-like interface with one method:

```
void execute(ActionContext context)
```

Support for conventional Actions would stay in place, but as an alternative, a class could implement ActionCommand and unbind itself from the HTTP API.

ActionContext - A Chain Context that implements the logical Action class API.

Existing code could be converted by changing references to context.* and so forth. The context could be constructed by the Request Processor, as an optional Command in the Chain, so that it could be exposed this through thread-local, opening the door for POJO actions that don't implement a particular interface.

ViewContext (pending) - A Chain Context that implements the logical "Velocity Tools" API.

In a later release, when the new members stabilize, we could move the taglib dependencies from the servlet contexts to the ViewContext. View technologies could then look exclusively to the ViewContext rather than poke around in the various servlet contexts. (Of course, support for the original architecture would remain for some time, to give third party libraries the chance to migrate.)

After having a chance to work with ActionContext and ViewContext ourselves, we could introduce more support for these members in a later release. But for 1.3.x, they could be marked "experimental".

The Commons Chain WebContext we now pass around Struts Chain could be called the "StrutsContext" to differentiate it from the ActionContext and ViewContext.

(Are we now starting to call everything "Context" instead of "Action"? Not really. We use the "Context" suffix when a member extends Chain Context. This convention is unlike the current "Action" soup, since "Context" is a suffix that identifies a member's "family" history.)

Another experimental member is the catalog element, which could be used to to support using a Chain of ActionCommands in lieu of an Action.

Struts Action Framework 1.4.x considerations

One we get past 1.3.x, there are some other things that we might consider.

Consider combining DTDs. Right now, using "standard" extensions like Tiles and Validator mean using more than one configuration file. While using multiple configurations files can be a good thing, we should also try and support the idea of having a single configuration file. This might not work-out for Tiles, but we might be able to at least integrate the Validator configuration with the DynaForm configuration.

Consider refactoring for Spring. We identified the need for adding a IOC container to Struts some time ago, but stalled on the point of which to use. Since then, Spring has gained a lot of momentum. Spring is used by the MyFaces and Beehive teams, and its on the radar for Shale. There is already a Struts-Spring component in the Spring distribution and other common ground.

2007-02-09: Struts 1.3 has been branched. 1.3.x maintenance branch has been branched out. The HEAD represents future 1.4 version and is open for improvements.

Code Issues

1. [MichaelJ] **Consider the option to call all lifecycle functions explicitly** from an action class, allow to disable automatic form population.

    - [FrankZ] I personally would not want to see the auto-population and validation and such go away. I think them being declarative is a powerful notion. I DO however agree that a developer should be able to turn them off declaratively and do it all manually.

- Implemented on 2007-02-09: ActionForm autopopulation is now configurable from an action mapping via "populate" attribute. Automatic call to ActionForm.reset is now configurable from an action mapping via "reset" attribute. See http://svn.apache.org/viewvc?view=rev&revision=505640

2. **The ability to switch off auto-form population** [FrankZ] The idea here is that Struts would still instantiate an ActionForm and call reset(), but that's it. This can be useful if you want to use an ActionForm only as an output object, but want to handle input manually. This came up for me converting a non-Struts app to Struts, where there was no notion of an ActionForm in the previous framework.

    - [MichaelJ] +1 I even had the ticket opened for this issue, but as Martin Cooper pointed out, this can be achieved simply by not associating the form with setup action. On the other hand, having form name explicitly in the action mapping makes config file more readable.
    - Implemented on 2007-02-09: (see previous bullet)

3. **Base Action class should implement event handling a.k.a. dispatching** [MichaelJ] If event is recognized for an action, it is dispatched to a corresponding method. If event is not found in the request, then execute() is called like for a regular action.

4. **Built-in support for RAP (Redirect After Post) pattern** [FrankZ] I'm not sure how best to accomplish this, but this should be a very easy thing for a developer to do, the framework should do any required heavy lifting. Again, not so much a rough spot as it is something I think a lot of people would appreciate.

    - [MichaelJ] To simplify redirect-after-post pattern, Struts must implement Flash scope, that is, the scope that survives after redirect, but is automatically cleaned up after redirected request finishes. Currently similar pattern is applied to the messages queued to session scope.

5. **Pure POJO ActionForms** [FrankZ] An ActionForm should not need to extend ActionForm. The framework would have to be smart enough to not call validate() and such in that case. This would allow for the Action to be the ActionForm as well, and this is really the underlying goal because many people view them being separate as a rough spot.

    - [MichaelJ] Definite +1

6. **Built-in AJAX support** [FrankZ] I am not entirely sure what the best form for this is, although I believe the AjaxTags paradigm still has a great deal of merit. In any case, in today's world, not offering some degree of AJAX out of the box is probably a rough spot for many.

    - [MichaelJ] I believe that I can combine my JSP Controls Tag Library with Tiles, building a real component subframework on top of Struts, having built-in Ajax features as well (actually, AHAH. It is coarse-grained in-place update of a component within a composite page.) I believe that it can coexist nicely with more finegrained approach of AjaxTags.

7. Implement scope that lasts beyond one roundtrip, but shorter than session (a.k.a. RolloverScope, Conversation scope, Flash scope).

Best Practices Issues

1. **Dispatch-type actions should be more front-and-center** [MichaelJ] Dispatch-action style request processing should become a cornerstone technique instead of "yet another way of doing things". It works especially well in data entry form processing or menu processing. See DataEntryForm.

    - [FrankZ] As an oft-stated opponent, GENERALLY, of Dispatch-type Actions, I wouldn't be thrilled with this being sold as a "best practice". I would have no problem with it being described in more detail and the pluses and minuses stated for people to be able to make a more informed decision either way.

2. **Preference for session-scoped ActionForms** [MichaelJ] Best practices should explain that there is nothing wrong in having session-scoped action forms. Best practices should teach how to build stateful web resources.

    - [FrankZ] -1. Saying there is "nothing wrong" with it, I do not believe is true. Speaking as someone who works all day in a large distributed enterprise environment, I am very well aware of the pitfalls of storing things in session and in letting session get too big. I'm not saying I never store anything in session, of course I do! 🙂 But I think people need to know the positives and negatives of doing so and decide what is appropriate in a given use case rather than being told this or that is a "best practice".
    - [MichaelJ] My original wording did not have "preference". What I am saying is that session-scoped forms should not be a taboo, and their benefits (as well as pitfalls) should be clearly explained.
    - [MichaelJ] After Rollover scope is implemented it should be preferred over session scope for redirect-after-post and other multi-request but relatively short term interactions.

Struts Action Framework 1.5.x considerations

Based on our own work with the "experimental" members introduced in 1.3.x, we might consider some other changes.

Consider a "smart" action type. The idea is that a command in Struts chain could look at the type indicated by the ActionMapping so both Action classes and ActionCommand implementations are supported. People could then mix-and-match Actions with ActionCommands (or even chains of ActionCommands). We might even try placing an ActionCommand interface on ActionForm, so people could skip having a seperate Action or ActionCommand class. The ActionForm could do it all.

Consider a "populate" method on ActionForm. From an OOP standpoint, it might be cleaner if an ActionForm populated itself rather than rely on a "god" class to populate it from the outside.

Consider a "FormContext" mechanism. Rather than "throw-away" a request-based ActionForm, the object could be seralized as a hidden-field or session-object and restored on the next request. Many other frameworks support this behavior now. Struts would have a slightly different spin, since we look at the form as an named entity rather than as an anonymous aggregation of other objects.

1. **Building stateful components** [MichaelJ] Best practices should teach how to build stateful and independent web components with Struts.

2. **ActionForm as true I/O buffer** [MichaelJ] ActionForm should be used as I/O buffer instead of simply collecting user input from request.

    - [FrankZ] Could you explain this further? I'm not at all clear on what you mean.

    - [MichaelJ] This is what 60% of Struts users do anyway. I guess you do it as well. You have actionform associated with input action. You submit a form, actionform is populated. In case of error or whatnot you render the same JSP page and pull the data from the actionform, it is already there, saved for you by Struts. I use this all the time for data entry use case, but this does not seem to be an officially endorsed and advertised practice.

Struts Action Framework 1.6.x considerations

Consider multiple controllers. One reason we introduced modules was because "there can only be one" Struts controller in an application. With the context-based changes we making, we might be able to introduce a mechanism to support a collection of Struts controllers, each identified by its servlet or filter name, each with it's own URI pattern. The key to being able to do something like this is for view members to look to the ViewContext rather than the various servlet contexts. Each Struts controller would place the appropriate ViewContext in its own requests.

Consider an alternate configuration file. There are idiosyncrasies in the element/attribute names of the struts-config which often confuse new developers. By either supporting alternative configuration loaders, or applying a stylesheet to a XML configuration, we could support both the "classic" and a new configuration (based on something like Struts Jericho).

Portlet (JSR-168) Whiteboard

There are three major issues with supporting JSR-168 (and probably a bunch of smaller ones as well):

- The framework APIs assume servlet API objects (ServletContext, ServletRequest, ServletResponse), whereas JSR-168 talks about PortletContext, PortletRequest, and PortletResponse. We'd either need to change the calling sequence for Action.execute() – problematic for backwards compatibility – or fake it somehow in a portlet environment.
- The lifecycle of a portlet request is actually divided into two chunks – processing and then rendering. From a Struts perspective, that means making sure that the first part of the request processor pipeline need to happen in the "process" part, and the forwarding to the resulting page needs to happen in the "render" part.
- Today, Struts owns the process of calculating URLs for pages and actions. Because it's in a webapp, it knows exactly what to do for the developer. However, in a portlet container it's actually the portal server that manages URLs, so a Struts-based portlet would need to interact with the portlet APIs for this purpose.

A strong goal should be that an application should be usable either as a webapp or as a portlet, with few (ideally no) changes. Therefore, we should build whatever it takes to support this into the standard framework distribution, which could then be used in both environments.

Other considerations

Code Issues

1. [MichaelJ] **Consider bringing FormDef into Struts core** and recommend using dynaforms with nested business objects as a best practice. Reason: I/O conversion is the suckiest part of Struts 1.

2. [MichaelJ] **Deprecate automatic validation**. Newbies always stumble upon the fact that their action class is not called when validation fails. Instead, promote explicit validation.

- [FrankZ] Definite -1 from me. Again, I'm +1 to being able to turn it on and off, but I very much believe it should be there. Perhaps there is some room for better logging in the cases you mention?

3. [MichaelJ] **Deprecate "input" attribute** of "action" tag in struts-config.xml. At least, rename it to "error" or something. A frequent misconception is to think that the lifecycle starts with displaying an "input" page, which is obviously not true.

   - [FrankZ] +0. I don't disagree with you at all, and I think there are probably other things that could similarly be changed. However, I think it is very important that anything done with Struts 1 be evolutionary and take backwards-compatibility into consideration in a big way. I think if you want revolution you go for Struts 2 (a minor revolution in that it's theoretically still compatible) or Shale.

4. [MichaelJ] **Add "form" attribute** with the same meaning as "name" attribute; deprecate "name".

   - [FrankZ] +0. Same comment as the above point.

5. [FrankZ] **Actions should be instantiated for each request**, therefore removing the thread safety concerns that exist today.

   - [MichaelJ] -0. Not sure that this bothers me anymore, especially if dynaforms will be adopted as standard practice. In this case a dynaform would be a container for a nested business object. I would rather combine Action and ActionForm and have an option to choose scope just right now a scope can be chosen for an ActionForm. That is, I am against strictly per-request actions.

6. **Custom attributes on tags** [FrankZ] All Struts tags that render HTML should allow for arbitrary attributes. I again propose a "specCompliant" attribute on the <html:html> tag. When true, no arbitrary attributes are allowed (this would be the default if the attribute is not present). If false, any attribute can be added. This has been a hassle for me a couple of times where I wanted to store some custom information on a tag for client-side purposes.

   - [MichaelJ] +1

7. **Built-in dependency injection** [FrankZ] This should be modeled after what is offered in JSF. If we took the code from the DependencyFilter in Java Web Parts, added in true injection (shouldn't be a terribly big deal), I believe we would have even better capabilities than in JSF. Spring is of course still out there for those that need/want more! This may not be so much a rough spot as just a fairly low-hanging piece of fruit (because most of the work is already done by virtue of leveraging DependencyFilter) that I think people would appreciate having.

8. **ThreadLocal ActionContext** [FrankZ] Yes, I think this is one place we should flat-out rip off Webwork 🙂 Backwards-compatibility would have to be considered, but I'd like Actions to have to deal with a single object, and for that object to be accessible via the ThreadLocal mechanism. This should also open the door for POJO Actions.

9. **Pre and post-processing abilities** [FrankZ] A developer should be able to specify a class and method to call before and after an Action executes, on a per-mapping basis. This should be independant of modifying a chain. Should just amount to adding the appropriate config file changes and two commands to the default chain. This is for handling things like common setup of view-type Actions, etc.

   - [MichaelJ] +0. I prefer having 1:1 correspondence between mapping and action class. With autopopulation out of the way, I can call whatever I want right in the beginning and at the end of execute() method. Same thing, but simpler, I think.

10. **MAYBE roll StrutsWS into Struts** [FrankZ] Heck, I'm not even sure *I* would +1 this idea! Might be worth considering at least.

Documentation Issues

1. **Chain documentation** [MichaelJ] Struts 1.3.x introduces the concept of chain of commands, but there is no comprehensive documentation on how this new architecture can be used for initial request processing (as a substitute to solid RequestProcessor) as well as for business processing (using Commands instead of Actions, etc).

2. **Taglib docs are confusing** [MichaelJ] Taglib reference uses too many words for simple things. Words like "attribute", "value", "property" used in different contexts, which does not help much in understanding how the tags really work. Also, Taglib docs should show one best way of doing things instead of saying that one can use this, or this or this in combination with this.

3. **Visual representation of tags that render widgets** [MichaelJ] Tags that render widgets should have examples and pictures. Here is some initial work in attempt to improve this: [StrutsWidgets].

Best Practices Issues

1. **Focus on best and most common approaches** [MichaelJ] Instead of having several ways to do things, Struts 1 should focus on one best and the most common way, even if it involves a couple more lines of code. Less is more.

2. **Webresource-centered approach** [MichaelJ] Up until now Struts 1 best practices center around a View, usually a JSP page. Instead, best practices should teach the webresource-centered approach.

    - [FrankZ] Speaking for myself, I would need to see a good definition of this approach before I was +/- on it. I have never seen a clear explanation that made me say "oh yeah, I see, that's better!"

    - [MichaelJ] Check out DataEntryForm. Well, it uses dispatching action and session-scoped form, but if you look at it from 10000 ft, it gives the idea of what web resource is about. Basically, it is one-two actions, one form and one or more JSPs. A resource does NEVER forwards to a JSP page that does not belong to the resource. In the best case we can have nice and simple 1:M relationship between action and JSPs.

3. **Use of nested properties** [MichaelJ] Emphasize using of nested properties, using business objects as nested properties, using dynaforms holding business objects.

Relevant Proposals

- Commons Chain of Responsiblity package
- Struts Chain Request Processor
- Struts Classic Release Plan 1.3.0
- struts-faces taglib
- Struts Jericho
- Struts Shale
- Struts Ti
- Struts OverDrive