actions (2.1 legacy document)

Table of contents

Table of contents

Warning:
This document was copied as is from the Cocoon 2.1 documentation, but has not yet been fully reviewed or moved to its new home.

1.

This is the proposal for a Action sitemap component. It shows the basic ideas behind actions. For more information look at the javadoc comments in the file Action.java in the org.apache.cocoon.acting package. For a possible implementation look at the file HelloAction.java or at the other actions in the same package. To give you an overview of the discussion about sitemap Actions here is an excerpt from the cocoon-dev mailing list: ----------<<>>---------- Peter Donald wrote: > > At 04:15 9/9/00 +0200, you wrote: > > <match type="uri" pattern="myapp/**"> > > <!-- the following Matcher make the overall request > > analysis and makes important information available > > to subsequent components through the objectModel and > > a Map for substitution in src= attributes. > > Maybe a Controller sitemap component would be more > > appropriate > > --> > > <match type="myapp-controller" pattern="not really used"> > > <select type="myapp-action-selector"> > > <when test="add"> > > <!-- here the {1} is delivered by the matcher > > "myapp-controller" > > --> > > <generate src="myapp/{1}/remove.xsp"/> > > </when> > > <when test="remove"> > > <generate src="myapp/{1}/remove.xsp"/> > > </when> > > <otherwise> > > <generate src="myapp/{1}/index.xml"/> > > </otherwise> > > </select> > > <select type="browser"> > > <when test="netscape"> > > <transform src="myapp/{1}/style-ns.xsl"/> > > <serialize type="html"/> > > </when> > > <when test="wap"> > > <transform src="myapp/{1}/style-wap.xsl"/> > > <serialize type="wap"/> > > </when> > > <otherwise> > > <transform src="myapp/{1}/style-ie.xsl"/> > > <serialize type="html"/> > > </otherwise> > > </select> > > </match> > > </match> > > <-- here we catch all requests not handled by the match above > > and redirect them to the login screen or an error page > > --> > > <match pattern="myapp/**"> > > <redirect-to uri="myapp/login"/> > > </match> > >Well, this example might not be the best one but I think its > >good enough to start a discussion about it. What do you think? > > I like the approach but before an implementation can be defined there is a > few questions that must be answered first. Namely > > * What is an Action ? > * How will you use actions ? > > What is an Action ? > ---------- > > In the above case you have associated an Action with a resource. ie the > Remove action is associated with myapp/{1}/remove.xsp resource. Do actions > have to be associated with a resource or are they independent of resources. > I would say that they are independent - an Action framework should be able > to be used via multiple interfaces whether via cocoon, turbine, telnet, > mail etc. So the "ShutdownRealWorldMachine" action is accessible via the C2 > interface, a telnet interface or via any number of other methods. The example above is probably misleading because we don't have a Action component in the sitemap so far. Generally speaking I think a Sitemap Action is an object that acts upon an application model based on data supplied by the environments objectModel (ie Request). It's its responsibility to make some data available to the Sitemap engine as further selection/matching criteria (a List object) as well as in the objectModel for other sitemap components Suppose the Interface of an Actions is like this: public Interface Action { public List act (Map objectModel); } and also suppose that the nested elements of the <map:act> elements are skipped if the action returns a null value instead of a List. <match type="uri" pattern="myapp/**"> <act type="myaction"> <select type="action-selector"> <when test="remove"/> <generate src="myapp/{1}/remove.xsp"/> </when> ... </select> </act> </match> The List object supplied by the action is used by the sitemap engine to replace occurrences like {1} from certain

attributes like src= but this is optional. So the Action is not really associated to a resource. It's the selector which does this association. > An action is essentially a snippet of code that is executed in response to > a request in a certain context (or Environment in C2s case). The action can > add and change the context/environment data. Agreed. > How granular can actions be ? You should be able to be as granular as you want. > Does session validation count as an action ? Why not? > How about authorisation and authentication ? I still suppose to leave authentication to the container but I know someone will do it with a sitemap component :/ Authorisation is more dependant on the application context and there are the possibilities to use AuthorisationMatchers, AuthorisationSelectors, AuthorisationActions or a authorisation logicsheet, what ever suit your needs best. > What about form validation ? Even here, it depends. If you only want to validate form data a Selector can be used to achieve that and in the <map:when> elements you regenerate the resource if validation fails or choose an action to put the form data into your application model and generate the next screen or whatever. > When I program actions I use a extremely granular approach. No problem, you should be able to do things like that <match type="uri" pattern="myapp/**"> <act type="session validation"> <act type="form-validation"> <select type="validation-check"> <when test="ok"> <act type="model-update"/> <generate src="next-page"/> </when> <otherwise> <generate src="this-page"/> </otherwise> </select> </act> </act> <generate src="login"/> </match> > There is also the idea of latent actions. For instance the latent Action is > transmitted between end-user and cocoon until they are activated. Actions > may also be made latent (or deactivated) in a couple of cases. Like when > you try to submit form but are not logged in - it will save action/form > data (or put action into latent state) and then after login commit the > action (or re-activate action). Isn't this a matter how components play together? > How will you use actions ? > ---------- > > In many cases when I program to a an actions approach each request can > result in many actions being executed. For instance it is not uncommon for > an action chain to occur like the following. > > SessionValidatorAction --> RoleAssignerAction --> FormValidationAction --> > FormDBEntryAction > > The SessionValidatorAction will check the session and if not exist then it > will put a magic token in environment so that after action is executed then > the rest of action-chain and output resource is ignored and the user is > redirected to a login page. > > The RoleAssignerAction (lame name I know ;->) will check if the current > user implements a certain role and if not redirect them to appropriate > NoEntry.html type page. > ... > > So when I design a sitemap for a web-application I want to somehow be able > to do the following. > > * Anything under webapp/ must run SessionValidatorAction > * Anything under webapp/admin must run RoleAssignerAction and check for > "admin" role > * Then specific resources webapp/a.xml, webapp/b.xml and webapp/admin/c.xml > must run FormValidationAction with appropriate form schema and the > appropriate FormDBEntryAction Didn't get the last one? What is a FormDBEntryAction for? Probably an XSP page? > * A user can also arbitrarily submit an action from any page (via post > request or perhaps a ?action=blah tacked onto URL) that must be executed. <match type="uri" pattern="webapp/**"> <act type="session-validation"/> <match type="uri" pattern="webapp/admin"> <act type="assign-role"> <select type="role-selector"> <when test="admin"> <match type="uri" pattern="webapp/admin/c.xml"> <act type="form-validation src="admin/form-schema-c.xsd"/> <!-- the following next-page action has knowledge of the sequence of pages and returns a List with the first element corresponding to the "next page" appropriate depending on values in the objectModel signaling successful validation by the previous action (type="form-validation"). The following three lines could be put into a sitemap resource definition and replaced by <redirect-to resource="next-page"/> --> <act type="next-page"> <generate src="{1}"/> </act> </match> <otherwise> <match type="uri-regexp" pattern="webapp/(a|b)\.xml"> <act type="form-validation

src="form-schema-{1}.xsd"/> <act type="next-page"> <generate src="{1}"/> </act> </match> </when> </select> </act> </match> </match> > > ---------- > --------- > > So what would I want to see in a Cocoon-Application framework ??? > > Well actions are independent of resources and exist in a separate namespace. > > Each request can in theory result in a action-pipeline. > > Each action can add stuff to it's context (or Environment). > > Each action can in theory short-cut the action pipeline and move to another > action-resource pipeline and also store remaining submitted actions as > latent actions. > > An action pipeline must not necessarily be associated with a resource (it > should instead be able to specify a resource that it goes to post processing). > > It may also be good to have an action that's sole purpose is to extract > explicit action requests and execute/store them (ActionExtractorAction + > ActionDispatcherAction ???) Please answer these question yourself after you've read my explanations. > But anyways I mean in no way to imply C2 is bad and if you want to add > hooks into sitemap generation to allow for this sort of stuff (or even > better do it yourself ;>) I would gladly switch all my development across > to C2 and I suspect many others would too :P. Implementing the framework to use actions like I've mentioned through out this mail is a matter of an hour or two. But you're right implementing general actions for general usage is another thing. Giacomo

1. Comments

add your comments