FlowscriptAndSessionReplication

Goal

- make session replication work with Flowscript

Some (nearly free) side-effects are:

- store the continuations at the client (no state at the server)
- store the continuations at the server's (filesystem, database) so that they survive server shutdowns

Background

Cocoon uses Mozilla Rhino for its Flowscript implementation. Looking into a Cocoon sitemap reveals that flowscripts are registered in

```
<map:flow language="javascript">
  <map:script src="flow.js"/>
  <map:script src="flow1.js"/>
</map:flow>
```

For every sitemap a scope (A scope is a set of JavaScript objects. Execution of scripts requires a scope for top-level script variable storage as well as a place to find standard objects like Function and Object - find more at http://www.mozilla.org/rhino/scopes.html) that is put into the users session.

Additionally to the scope, Cocoon has to manage all continuations that are created while executing the functions of the flowscript. Since Cocoon 2.1.6, these continuations can be stored in the Session too (its configurable in cocoon.xconf).

```
 +------------------------+
 |SESSION1                |
 |+----------------------+|
 ||scope (sitemap A)     ||
 |+----------------------+|
 |+----------------------+|
 ||scope (sitemap B)     ||
 |+----------------------+|
 |+----------------------+|
 ||scope (sitemap C)     ||
 |+----------------------+|
 |+----------------------+|
 ||ContinuationsHolder   ||
 ||(all continuations of ||
 || this user)           ||
 ||                      ||
 ++----------------------++
 |+----------------------+|
 ||other session attr.   ||
 ||                      ||
 ||                      ||
 ||                      ||
 |+----------------------+|
 +------------------------+
```

What's the problem?

Serialize and deserialze objects

Description

To enable session replication between two application servers (e.g. Tocmat, see http://jakarta.apache.org/tomcat/tomcat-5.5-doc/cluster-howto.html) you have to make sure that all session attributes are serializable (implement `java.io.Serializable`).

Unfortunatly this wouldn't work for Cocoon because of two problems:

- As the scope has references to many Cocoon internal objects (see the Cocoon object) serialization wouldn't work because many of them are not serializable

- Serializing all the large objects would be very expensive

How do session replicators work? What triggers the replication? I remember to have read somewhere that it's triggered by the session.setAttribute() method, in which case changing the state of a session attribute but not the attribute itself causes problems.--[SylvainWallez] *See e.g. the Tomcat documentation.* ReinhardPoetz

Possible way to solve this problem?

(outlined by Christopher Oliver)

I think the first thing you need to do is determine exactly what needs to be serialized into the replicated session. A given Rhino global scope object can contain a lot of references to Java objects that may not be serializable or it may not be desirable to serialize them.

All such objects need to be identified and "replaced" with something else in the serialization stream. You can override the replaceObject() and resolveObject() methods of ObjectOutputStream and ObjectInputStream to implement this. When the stream is deserialized you can then substitute suitable replacement objects from the target JVM. See the org.mozilla.javascript.serialize package in the Rhino distro for an example of this strategy.

Since the FOM objects refer to underlying Cocoon objects that should not be serialized (the session itself, the source resolver, etc) the above strategy would at least need to applied to them. In addition, objects created by the user that are referenced via the Rhino global scope or the local variables of a continuation may need this treatment.

See also

- http://lxr.mozilla.org/mozilla/source/js/rhino/src/org/mozilla/javascript/serialize/ (prototyp for Rhino scope serialization/deserialization)
- http://java.sun.com/j2se/1.4.2/docs/guide/serialization/spec/serialTOC.html
- http://java.sun.com/j2se/1.4.2/docs/api/java/io/ObjectOutputStream.html
- http://java.sun.com/j2se/1.4.2/docs/api/java/io/ObjectInputStream.html
- http://java.sun.com/j2se/1.4.2/docs/api/java/io/Serializable.html
- http://wiki.apache.org/cocoon/RhinoWithContinuations

Plugging in the solution into existing containers

Description

The serialization/deserialization process of Java objects has to be overriden (see explanations by Chris). This mechanism has to be plugged into the container without having to modify it.

Possible soluation

- Override the methods `writeReplace` and `readResolve` of the `ContinuationsHolder` and of the Javascript scope. This will require a wrapper object for the scope.
- Write own implementation of `ObjectInputStream` and `ObjectOutputStream` that are used by the container as default.

Delta management

Description

Exchanging e.g. the `ContinuationsHolder` because only one Continuation has been added, would be very expensive.

Rather than having a single [ContinuationHolder] in the session that keeps all continuations, it may be better to have each continuation stored as a separate session attribute. That may both trigger the replication system and reduce the replication load. --[SylvainWallez]

Possible solution

Add some kind of delta management into the serialization/deserialization process. But how ...?

Components managed by the Cocoon service manager

Description

If components are used in flow that are managed by the Cocoon service manager, we have to make sure that the references in the target JVM are still valid.

```
function xy() {
  var x = cocoon.getComponent("myComponent");
  var y = cocoon.createObject("myObjectImplementingAvalonInterfaces");
}
```

cocoon.getComponent() can return a serializable proxy, but cocoon.createObject() must return a serializable object. There's also a problem with CForms: a form contains its own data model, which can be serialized, but also pointers to its definition, which is shared between form instances and can contain arbitrary references to non-serializable objects and components.--[SylvainWallez] *In this case the Form object has to provide its own serialization /deserialization mechanism that make sure that its state is exchanged correctly.* ReinhardPoetz

Possible solution

- Forbid the usage of components
- *... how to manage references to service manager or service selector? May be through serializable proxies... Or may be by documenting that no flow method should have non-serializable objects on stack...* --Vadim

More to read ...

- onjava.com, Session Replication in Tomcat 5 Clusters, Part1, http://www.onjava.com/pub/a/onjava/2004/11/24/replication1.html
- onjava.com, Session Replication in Tomcat 5 Clusters, Part2, http://www.onjava.com/pub/a/onjava/2004/12/15/replication2.html
- onjava.com, Clustering and Load Balancing in Tomcat 5, Part1, http://www.onjava.com/pub/a/onjava/2004/03/31/clustering.html
- onjava.com, Clustering and Load Balancing in Tomcat 5, Part2, http://www.onjava.com/pub/a/onjava/2004/04/14/clustering.html
- Weblogic: Failover and Replication in a Cluster, http://e-docs.bea.com/wls/docs81/cluster/failover.html#1022034
- IBM: There's more than one way to cluster a Web application, http://www-128.ibm.com/developerworks/java/library/j-jtp07294.html
- IBM: Websphere documentation, http://publib.boulder.ibm.com/infocenter/ws51help/index.jsp
- IBM: High-impact Web tier clustering, http://www-128.ibm.com/developerworks/java/library/j-cluster1/index.html
- Designing J2EE Applications for Real-Life Clustered Environments, http://www.onjava.com/pub/a/onjava/2004/07/14/clustering.html