Functional Specification for ConvergedHttpSession

Author: jan.luehe@sun.com

Version: 0.1

1 Introduction

<List proposed feature(s). Introduce the basic vocabulary. Why is this interesting? List capabilities that may be normally expected, but are not being supported. Are there any limitations and caveats that need to be disclosed?>

In a converged, JSR 289 compliant container, HTTP sessions are exposed to servlet code as instances of the `javax.servlet.sip.ConvergedHttpSession` ("`ConvergedHttpSession`") interface, which extends the `javax.servlet.http.HttpSession` ("`HttpSession`") interface defined by the Servlet specification.

Through its `ConvergedHttpSession` interface, an `HttpSession` may be associted with (that is, added as a child protocol session to) a `javax.servlet.sip.SipApplicationSession` ("SipApplicationSession"). Once this type of association has occurred, the `ConvergedHttpSession` interface allows navigation from the `HttpSession` to its `SipApplicationSession` parent.

Features

<List all requirements and features you are implementing. List those which may be normally expected to be implemented but are not.>

All methods of the `javax.servlet.sip.ConvergedHttpSession` interface are implemented.

2 Design Overview

<Discuss the core concepts of the design. Provide diagrams. Show how this sub-system interacts with other features and sub-systems. You may write as much as you consider as useful for other developers to understand the basic design. Insert scenarios, use cases that will help the reader understand how the system will behave under various conditions.You can also look at this section as a map to navigate documented code! >

2.1 ConvergedHttpSession facade class

For any implementation object that needs to be exposed to application code using a standard interface, the web container in GlassFish defines a corresponding facade class which

- wraps the implementation object.
- implements the standard interface (thus hiding any implementation details), and
- delegates its method invocations to the nested implementation object.

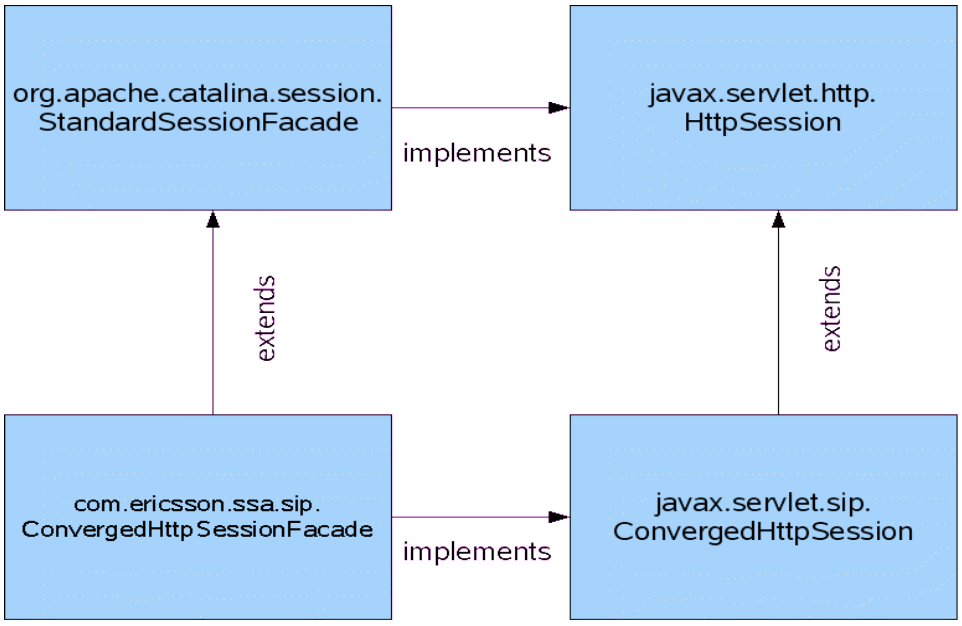
For example, in GlassFish, an `HttpSession` is implemented as an instance of `org.apache.catalina.session.StandardSession` ("`StandardSession`). Before a `StandardSession` object is returned to the servlet code that requested it, the `StandardSession` object wraps itself inside an instance of `org.apache.catalina.session.StandardSessionFacade` ("`StandardSessionFacade`"), as follows:

```
public HttpSession getSession() {

   if (facade == null){
     facade = new StandardSessionFacade(this);
   }
   return (facade);
 }
```

`StandardSessionFacade` implements the `HttpSession` interface by delegating to the encapsulated `StandardSession` object.

SailFin follows the session facade paradigm established by the GlassFish web container, by exposing converged HTTP sessions as instances of `com.ericsson.ssa.sip.ConvergedHttpSessionFacade` ("`ConvergedHttpSessionFacade`") , which extends `StandardSessionFacade` and implements the `ConvergedHttpSession` standard interface:



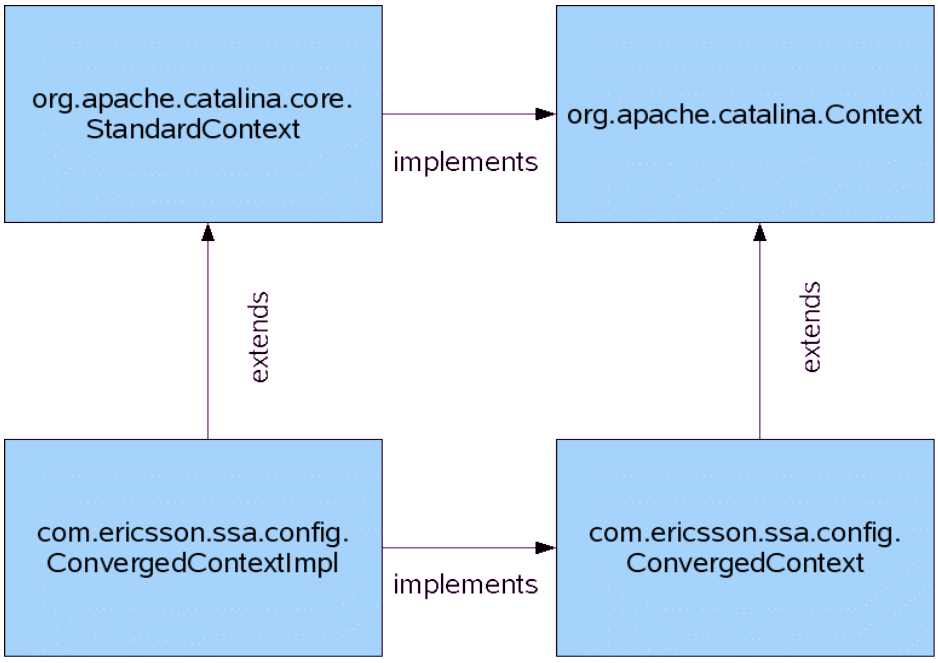
While `ConvergedHttpSessionFacade` implements any `HttpSession` methods by delegating to the nested `StandardSession` for which it is acting as a facade,

`ConvergedHttpSessionFacade` itself is responsible for implementing any of the extra methods defined by the `ConvergedHttpSession` interface.

2.2 Exposing the right kind of session

In SailFin, the web container must expose HTTP sessions as instances of `HttpSession` to the servlets of a pure web application, and as instances of `ConvergedHttpSession` to the servlets of a converged application. This means that the type of application (pure vs converged) determines the type of session facade to be exposed.

In SailFin, a pure web application is represented by an instance of `org.apache.catalina.core.StandardContext` ("`StandardContext`"), which implements the `org.apache.catalina.Context` ("`Context`") interface, whereas a converged application is represented by an instance of `com.ericsson.ssa.config.ConvergedContextImpl` ("`ConvergedContextImpl`"), which extends `StandardContext` and implements `com.ericsson.ssa.config.ConvergedContext` ("`ConvergedContext`"), which in turn extends `Context`:



SailFin delegates the creation of the appropriate session facade for a given `StandardSession` instance to the underlying application context. For this, a new method `createSessionFacade()` has been added to `StandardContext`, which wraps its argument (of type `StandardSession`) as a `StandardSessionFacade`, as follows:

```
public StandardSessionFacade createSessionFacade(
                                  StandardSession session) {
  return new StandardSessionFacade(session);
}
```

This method is overridden by the `ConvergedContextImpl` subclass to wrap the `StandardSession` argument as a `ConvergedHttpSessionFacade`, as follows:

```
public StandardSessionFacade createSessionFacade(
                                  StandardSession session) {
  return new ConvergedHttpSessionFacade(session);
}
```

When asked to create a session facade of itself, `StandardSession` now leverages this new factory method by passing itself ("`this`") as an argument to it, so the new implementation of its `getSession()` method now looks as follows:

```
public HttpSession getSession() {

  if (facade == null){

    StandardContext ctx = (StandardContext) manager.getContainer();
    if (ctx == null) {
      throw new IllegalStateException("No context");
    }
    facade = ctx.createSessionFacade(this);
  }

  return (facade);
}
```

2.3 Replicating a ConvergedHttpSession's association with its parent SipApplicationSession

SailFin adds a new instance variable `sipAppSessionId` (of type `String`) to `StandardSession`, which stores the id of the HTTP session's parent `SipApplicationSession` (if any), along with corresponding getter and setter methods:

```
public String getSipApplicationSessionId()

public void setSipApplicationSessionId(String id)
```

If present, a `StandardSession`'s `sipAppSessionId` will be automatically included in the session's serialized representation, and as such will be replicated and restored along with all the other session data.

If a call to `ConvergedHttpSession.getApplicationSession()` causes a new `SipApplicationSession` to be created, the id of the new `SipApplicationSession` is passed as an argument to the `setSipApplicationSessionId()` method of the `StandardSession` instance that is nested inside the `ConvergedHttpSessionFacade`.

Likewise, if `ConvergedHttpSession.getApplicationSession()` is called, and the `ConvergedHttpSessionFacade`'s nested `StandardSession` object already has a `SipApplicationSession` associated with it, the `SipApplicationSession`'s id is retrieved from the `StandardSession` by calling its `getSipApplicationSessionId()` method, and the corresponding `SipApplicationSession` is looked up and returned.

2.4 Use cases

The following examples illustrate the various use cases of `ConvergedHttpSession`.

In the absence of any standard name, `sasID` is used as the name of the URL parameter that carries an encoded `SipApplicationSession` id.

Use Case 1:

The HTTP request URL does not contain any encoded `jsessionid` or `sasID`.

The following HTTP servlet code:

```
ConvergedHttpSession chs = httpReq.getSession();
SipApplicationSession sas = chs.getApplicationSession();
```

causes the container to create brand new `ConvergedHttpSession` ("chs") and `SipApplicationSession` ("sas") objects, and to connect "chs" with "sas".

Use Case 2:

Assume a conference manager application, which stores the id of every ongoing application instance (i.e., conference) as an attribute in its `ServletContext`. Each conference id identifies an ongoing `SipApplicationSession`, and

therefore may be interpreted as a `sasID`. The conference manager application contains a servlet or JSP which retrieves this `ServletContext` attribute and populates a FORM page with the `sasID` of every ongoing conference. The FORM is returned to the client, for the client to pick the `sasID` of the conference she would like to join. The selected `sasID` is submitted as a query parameter named `conferenceID`. The servlet (of the conference manager application) that is supposed to process the submitted FORM retrieves the value of the `conferenceID` query parameter, and looks up the corresponding `SipApplicationSession`.

In this case, the HTTP request URL does not contain any encoded `jsessionid` or `sasID` (it contains a query parameter named `conferenceID`, though).

The following HTTP servlet code:

```
// Look up existing SAS
String confID = request.getParameter("conferenceID");
SipApplicationSession sas = SipSessionsUtil.getAppSession(confID);
// Create new HTTP session. Assume its id is "1234"
ConvergedHttpSession chs = httpReq.getSession();
String jencoded = chs.encodeURL("http://myserver:8080");
URL jAndSasId = sas.encodeURL(new URL(jencoded));
```

causes a new `ConvergedHttpSession` "chs" to be created, which has **not** yet been connected to any `SipApplicationSession`.

The printable representation of the `jAndSasId` URL, which has both the `jsessionid` and `sasID` encoded in it:

```
http://myserver:8080/jsessionid=1234;sasID=xyz
```

is returned to the client in a 302 HTTP response. A subsequent request with this URL will cause the `ConvergedHttpSession` with id "1234" to be connected to the `SipApplicationSession` with id "xyz" (see Use Case 3).

Use Case 3:

HTTP request URL:

```
http://myserver:8080/jsessionid=1234;sasID=xyz
```

The following HTTP servlet code:

```
ConvergedHttpSession chs = httpReq.getSession();
```

```
SipApplicationSession sas = chs.getApplicationSession();
```

causes the `ConvergedHttpSession` "chs" with id "1234" to be resumed. In addition, "chs" will be connected to the `SipApplicationSession` "sas" with id "xyz".

Use Case 4:

HTTP request URL:

`http://myserver:8080/sasID=xyz`

The following HTTP servlet code:

```
ConvergedHttpSession chs = httpReq.getSession();
SipApplicationSession sas = chs.getApplicationSession();
```

causes the container to create a brand new `ConvergedHttpSession` "chs" and to connect it to the `SipApplicationSession` "sas" with id "xyz".

3 Performance

<How do you want performance team to measure this sub-system? Any micro benchmarks necessary?Any goals? Anticipated scalability limits or goals?>

The checking for the possible presence of an encoded `SipApplicationSession` id in a request URL, and its parsing from the request URL, have been moved into `ConvergedHttpSessionFacade`, to avoid any negative performance impact on pure web applications.

4 Management

<Describe how performance, management status, and diagnostic information is exposed. How does this feature handle dynamic configuration changes?>

Interfaces

<How is this feature(s) configured by administrator? Does it introduce new commands or modify existing ones? Show syntax of expected administrative commands and response codes. What is the schema for new configuration? Show the DTD snippets. What are their default values? What are the validation rules? List stability level for each of the above [committed|evolving|unstable|standard]. Does it consume interfaces from other projects or sub-systems (imported) or produce interfaces for consumption (exported).

`javax.servlet.sip.ConvergedHttpSession` is a standard interface defined by JSR 289.

5 Packaging, Files, and Location

<Does this feature add new jar files or extend existing ones? Where are they located?>

This feature does not add any new JAR files.

6 Quality

<Guidelines you wish to provide on how this feature is tested. Scenarios that must be tested.>

The scenarios tested must cover those listed in Section 2.3.

7 Documentation Requirements

<List the required documentation to support this product feature.>

There are no documentation requirements for this feature, which will be enabled automatically.

8 Open Issues

1. Need to come up with (proprietary?) name for the URL parameter that carries the SipApplicationSession id
2. Need to follow up with JSR 289 EG to have Section 13.4 ("Encoding URLs") of the spec clarified, which currently reads as follows:

   When the HTTP Request comes back to the converged container with this URL, the container MUST associate the new HttpSession with the encoded Application Session. In case the HTTP request is not a new request but a follow-on request already associated with a HTTP Session then the converged containers MUST use the HTTP session association mechanism described in the HTTP Servlet specification, to route the request to the right HTTP Session and MUST NOT use the encoded SipApplicationSession.

   The current wording may be misinterpreted to conflict with Use Case 3 in Section 2.3 of this Functional Specification, in which the requested `ConvergedHttpSession` is resumed **and** connected to the `SipApplicationSession` whose id is encoded in the request URL.

   Section 13.4 of the JSR 289 specification should be clarified to say that any `SipApplicationSession` id encoded in the request URL must be ignored only if the resumed `ConvergedHttpSession` already has a `SipApplicationSession` associated with it.