◆ ◆ ◆ CHAPTER 8

Creating a Session Persistence Module

GlassFish Server enables you to create a session persistence module in the web container for high availability-related functionality by extending the `StrategyBuilder` interface . Using the `StrategyBuilder` class in an HK2 service makes the session manager extensible because you can implement a new persistence type without having to modify the web container code.

For information about other high-availability, session persistence solutions, see Chapter 10, "Configuring High Availability Session Persistence and Failover," in *Oracle GlassFish Server 3.1 High Availability Administration Guide*.

The following topics are addressed here:

Extending the `StrategyBuilder` Class

You can extend the `StrategyBuilder` class by creating a new web session manager type.

```
package com.sun.enterprise.web;

import com.sun.enterprise.deployment.runtime.web.SessionManager;
import org.apache.catalina.Context;
import org.jvnet.hk2.annotations.Contract;

@Contract
public interface PersistenceStrategyBuilder {

    public void initializePersistenceStrategy(
            Context ctx,
            SessionManager smBean,
            ServerConfigLookup serverConfigLookup);

    public void setPersistenceFrequency(String persistenceFrequency);

    public void setPersistenceScope(String persistenceScope);
```

```
    public void setPassedInPersistenceType(String persistenceType);
}
```

Here is an example of how to extend the `StrategyBuilder` class by creating a new web session manager and setting a store for it:

```
@Service(name="xyz")
public class XYZStrategyBuilder extends PersistenceStrategyBuilder {

    public org.apache.catalina.session.Manager getManager() {
        Manager manager = super.getManager();
        manager.setStore(MyStore);
        //MyStore could delegate to for example the BackingStore in the case of our
          in-memory replication return manager;

or
    //Create a new manager, set the appropriate Store and return it
    return XYZManager;
    }

    public void init(StandardContext ctx, SessionManager sessionManager) {
        add listeners, values etc to the ctx,
    }
}
```

If a `Manager` is provided, then it will be used in GlassFish Server.

Note – If a backing store is required, it is the responsibility of the `Manager` to make sure that the `findSession` method correctly uses the `Store` that the `Manager` provides.

EXAMPLE 8–1 Extending `StrategyBuilder` With a Custom Web Session Manager

This example defines a session manager type that is named `MyHASolution`.

```
@Service(name="MyHASolution")
public class MyHASolutionStrategyBuilder extends PersistenceStrategyBuilder {

    public org.apache.catalina.session.Manager getManager() {
        return new MyHASolutionManager();

    }

    public void init(StandardContext ctx, SessionManager sessionManager) {
        // add listeners, valves etc to the ctx,
    }

}
```

EXAMPLE 8–2 Session Manager Configuration in the `glassfish-web.xml` File

This example sets the `persistence-type` attribute of the `session-manager` element of `glassfish-web.xml` to `myHASolution`

EXAMPLE 8–2 Session Manager Configuration in the glassfish-web.xml File *(Continued)*

Based on the domain.xml and glassfish-web.xml settings, the web container looks up the appropriate StrategyBuilder in the Habitat and uses it.

```
<glassfish-web-app>
  <session-config>
    <session-manager persistence-type="myHASolution"/>
  <session-config>
<glassfish-web-app>
```