Servlet Lifecycle Management

Description

Like most Java Enterprise applications, Vitro servlets rely on the ServletContext to hold object that they will need to use when servicing requests. These objects are created by ServletContextListeners, which are run by the StartupManager.

The StartupManager creates instances of the listeners and runs them, accumulating information about their running in the StartupStatus.

As each listener runs, it may add messages to the StartupStatus. Each message will have a severity level associated with it:

- FATAL – The listener encountered a problem. Perhaps the application was configured incorrectly, or perhaps the system utilities are not performing as intended. The problem is severe enough that the application will not run. The message describes the problem, with suggestions on how to fix it.
- WARNING – The listener encountered a problem, but the problem will not prevent the application from running. The message describes the problem, and tells what parts of the application will be affected, with suggestions on how to fix the problem.
- INFO – No problem is indicated. The message contains information that may be helpful in monitoring the application.

If a FATAL status is recorded, the StartupManager will not execute any additional listeners. Access to the application will be blocked, and any attempt to access the application will display the StartupStatus in an error page.

If a WARNING status is recorded, the StartupManager continues as normal. Access to the application will be blocked one time, to display the StartupStatus. In subsequent requests, the application will respond normally.

When logged in, an administrator may view the StartupStatus from a link on the Site Admin page.

Specifying context listeners

In any Java Enterprise application, developers can specify context listeners in the deployment descriptor (web.xml). These listeners that will be activated when the application starts and when it shuts down.

In Vitro, the only listener in web.xml is the StartupManager. Here is the relevant section of Vitro’s web.xml:

```
<!--
   StartupManager instantiates and runs the listeners from startup_listeners.txt
   All ServletContextListeners should be listed there, not here.
-->
<listener>
  <listener-class>edu.cornell.mannlib.vitro.webapp.startup.StartupManager</listener-class>
</listener>
```

Vitro contains a list of startup listeners in a file at Vitro/webapp/config/startup_listeners.txt. This file is simple text with each line containing the fully-qualified class name of a startup listener. Blank lines are ignored, as are comment lines – lines that begin with a “hash” character. Here is a portion of that file:

```
#
# ServletContextListeners for Vitro,
# to be instantiated and run by the StartupManager.
#

edu.cornell.mannlib.vitro.webapp.config.ConfigurationPropertiesSetup

edu.cornell.mannlib.vitro.webapp.config.RevisionInfoSetup

edu.cornell.mannlib.vitro.webapp.email.FreemarkerEmailFactory$Setup

# DefaultThemeSetup needs to run before the JenaDataSourceSetup to allow creation
# of default portal and tab
edu.cornell.mannlib.vitro.webapp.servlet.setup.DefaultThemeSetup
```

Writing context listeners

Each listener must implement the ServletContextListener interface, and must have a zero-argument constructor.

When Vitro starts, the StartupManager will call contextInitialized() in each listener, in the order that they appear in the file. The listener can call methods on StartupStatus to record messages. If the listener is successful, it should record one or more INFO messages that provide a brief description of what it has done. If a problem is detected, the listener may record WARNING messages or ERROR messages, depending on the severity of the problem. The listener may also throw a RuntimeException from contextInitialized(), which the StartupManager will treat like an ERROR.

Here is an example of a basic listener. When contextInitialized() is called, the listener will perform some setup. If there is no problem, a call to StartupStatus.info() reports some basic information about the listener's actions. If a problem is found, a call to StartupStatus.warning() describes the nature of the problem (by reporting the exception) and how this problem will affect the application.

```
public static class Setup implements ServletContextListener {
   @Override
   public void contextInitialized(ServletContextEvent sce) {
       ServletContext ctx = sce.getServletContext();
       StartupStatus ss = StartupStatus.getBean(ctx);

       try {
           FreemarkerEmailFactory factory = new FreemarkerEmailFactory(ctx);
           ctx.setAttribute(ATTRIBUTE_NAME, factory);

           if (factory.isConfigured()) {
               ss.info(this, "The system is configured to "
                       + "send mail to users.");
           } else {
               ss.info(this, "Configuration parameters are missing: "
                       + "the system will not send mail to users.");
           }
       } catch (Exception e) {
           ss.warning(this,
                   "Failed to initialize FreemarkerEmailFactory. "
                           + "The system will not be able to send email "
                           + "to users.", e);
       }
   }

   @Override
   public void contextDestroyed(ServletContextEvent sce) {
       sce.getServletContext().removeAttribute(ATTRIBUTE_NAME);
   }
}
```

Note that the StartupManager treats ServletContextListeners just like you would expect from reading the Servlet 2.4 specification:

- Only one instance of the listener is created per JVM.
- The contextInitialized() method is called once when the system is starting.
- The contextDestroyed() method is called on that same instance when the system shuts down.