codehaus

Vincent Massol Think Tank

On software quality, collaborative development and Open Source.

Using Cargo 0.4 for functional testing

[Think tank]

[**vmassol**] 19:43, Wednesday, 17 November 2004

Introduction

Automated tests are good. Automated Functional tests are even better as they are the proof that your application is working. In addition, with automated functional tests you can also automate your delivery process. However, writing automated functional tests is hard. The main reason it is hard is because you need to control your execution environment (database, application server, etc).

Cargo is a framework that you can use to automatically install, configure and execute J2EE containers. Thus it allows you to control your execution environment (for the J2EE container at least) and permits completely automated functional tests for J2EE applications.

Example

Let's walk through an example. Imagine we wish to start up Tomcat 4.1.31 before such test runs. Here's what we could write:

```
public class MyTest extends TestCase
{
     private Container container;

     protected void setUp()
     {
          // (1) Optional step to install the container from a URL pointing to its distribution
          Installer installer = new ZipURLInstaller(
              "http://www.apache.org/dist/jakarta/tomcat-4/v4.1.31/bin/jakarta-tomcat-4.1.31.zip");
          installer.install();

          // (2) Create the Cargo Container instance wrapping our physical container
          container = new Tomcat4xContainer();
          container.setHomeDir(installer.getHomeDir());
      }

     public void testSomething()
     {
          // (3) Statically deploy some WAR
          Deployable war = container.getDeployableFactory().createWAR("src/testinput/my.war");
          container.addDeployable(war);

          // (4) Start the container
          container.start();

          // (5) Perform any test you wish here
          [...]
      }

     protected void tearDown()
     {
          // (6) Stop the container
          container.stop();
      }
}
```

Step 1 is optional. You can also rely on the container being already installed on the test machine if you wish. However, it's nice to completely automated the testing and assume nothing (or very little - We still need an OS and a JDK on the machine). In this example we're fetching the Tomcat 4.1.31 installation from the web. We could fetch it from our intranet or from a location on the machine or from our SCM.

In Step 2, we have not told Cargo what container Configuration to use. Thus Cargo will use a default Configuration and it will configure it so that your container will execute in a temporary directory that it will create in your OS system tmp dir. If you wish to control this you can use:

```
Configuration configuration = new CatalinaStandaloneConfiguration(container, "target/tomcat4x");
configuration.setProperty(ServletPropertySet.PORT, "8080");
container.setConfiguration(configuration);
```

in step 3, we create a Cargo wrapper around a physical WAR and we add it to our container so that it is deployed when the container starts.

We then start the container (step 4), perform any testing we wish (step 5) and ensure the container is always stopped at the end of our test (step 6).

If we wish to start and stop the container only once during our whole test suite we can use a standard JUnit TestSetup.

Conclusion

This is just a short introduction to Cargo to demonstrate how easy it is to start/stop a container. The API is of course richer. Also, we're showing here how to use Cargo for functional testing of J2EE application but Cargo is also meant to be used by any application that requires a container to be up and running. It could also be used by IDE plugin writers, etc.

For more information on Cargo, please see the **Cargo website** and join us on the **Cargo mailing lists**. You'll be warmyl welcomed! :-)

Comments

Vincent,
This looks great. I'll be needing this type of stuff in the coming months for my new startup.

Why are steps #3 and #4 in testSomething() instead of in setup()?

My personal feeling is that the Intstaller stuff, while "neat", isn't that useful. Most people setting QA systems tend to have troulbe starting/stopping the app server and deploying their apps -- not installing the app server itself. I'd recommend keeping step #1 as a lower priority as Cargo develops.

Just my two cents.

--***Patrick Lightbody***, November 21, 2005 11:21 PM

"Clair comme de l'eau de roche".
Last week, I was just asking myself : "I have a Cargo module in my Netbeans installation, but what is it ?" And I forgot to ask google !. This morning I have just finished to write a functional test, with this line in the Test comment : "This test suppose that a Tomcat-ebxmlrr server is running .... "

So thanks for this very clear introduction, I will try it.
Now I am just asking myself how to configure a datasource ?

--***Laurent Foret***, November 22, 2005 02:17 PM

I am planning to use Cargo for a project I am working on. My intended usage is on the lines of what you have described here. Thanks a lot for developing this project!

Cheerio,
Binil

--***Binil Thomas***, November 22, 2005 02:42 PM

Hi everyone,

Thanks for all your positive comments. If you have any question about Cargo I invite you to join on the Cargo user or dev mailing lists (see http://archive.codehaus.org/cargo/).

--***Vincent Massol***, November 22, 2005 02:49 PM