Building a Software Factory

In This Chapter

Continuous Builds and Continuous Integration

Continuous integration and continuous builds are mainstay features of modern software factories.
Continuous integration is the practice of development changes being tested, built, and reported in small increments during the development process.
Continuous build is an extension of this practice to provide a newly tested build of the software every time a developer commits changes.

Principles of Continuous Integration

Understanding continuous integration principles, in conjunction with continuous build tools, helps drive an efficient software factory, and by extension, efficient software development. These principles include:

- A unified code repository -- the team must have access to shared code (team sharing, SVN)
- Frequent commits -- every developer commits frequently to the repository so changes are incremental
- Commits are built -- when the code is committed by a developer, the software is built to show it is functional
- Builds are automatically tested -- When the code is built, unit tests are performed, so if errors are found in a unit test, the developer can correct those errors immediately
- Results are available to everyone -- if a build breaks, everyone can see where the issue lies and what needs to be done to fix it
- Successful builds are deployed immediately (automated deployment) -- this ensures that the latest software is available to everyone in the team (product owners, testers)

Developer's Role

With these principles in mind, the developer's role in the continuous integration process can be summarized as follows. Developers should:

- Commit their code frequently.
- Review the results of unit testing, and if unit testing fails on their commit, fix the issue as soon as possible.
- Update their workspaces before starting each day to get changes from others in the team.
- Write test scripts for unit testing (unless this is the responsibility of the testing team.)

Software Tools

Software tools can help with managing and practicing continuous integration; in fact, continuous builds are normally vastly software driven. Software considerations include:

- Repository software - software to manage the code base (SVN)
- Build software -- software to actually build and run unit testing (ANT)
- Communication software -- how to communicate information among the team members.
- Deployment software -- in our case, Servoy Application Server.
- Continuous build software -- software to manage the continuous build process and make sure each step is occurring.

Continuous Build Software

There are many different choices for continuous build software available, such as Jenkins, CruiseControl, Continuum, Bamboo, and others.

This tutorial shows how to use Jenkins for the task at hand. There is also a sample page about using CruiseControl, but this one will not be updated in the future.

Using Continuous Build Software with Servoy

In regards to Servoy projects, Jenkins/CruiseControl is used to fill the following needs in our software factory

- Checking the SVN for new versions of the software, or building the software when the SVN is committed.

- Running the unit tests.
- Providing a new build of the software for testing and availability.

Jenkins/CruiseControl are very flexible in how they can be configured and the methods and strategies can differ depending on a variety of factors. In the case of using them with Servoy, we utilize their ability to run ANT scripts. With these ANT scripts, another tool in Servoy Developer exports the solution into a `.servoy` file, then another ANT script runs a smart client based test client that will import the solution into a repository and run unit tests. Finally, if unit tests are passed, an export file is created in a folder - ready to be imported automatically into a test application server.

Rather than explain everything in detail that can be done with Jenkins/CruiseControl and a Servoy project, it is more beneficial to provide an example of Jenkins/CruiseControl and the supporting applications configured for a continuous build server. The Building the Servoy Software Factory section will cover in detail setting up a continuous build server for a Servoy project.

> If trying this in a headless Linux, make sure to have an X11 (UI) installed (required by the import test client). Xvfb can be used for example, for those that are meant to be headless.

Updating a Servoy Application Automatically

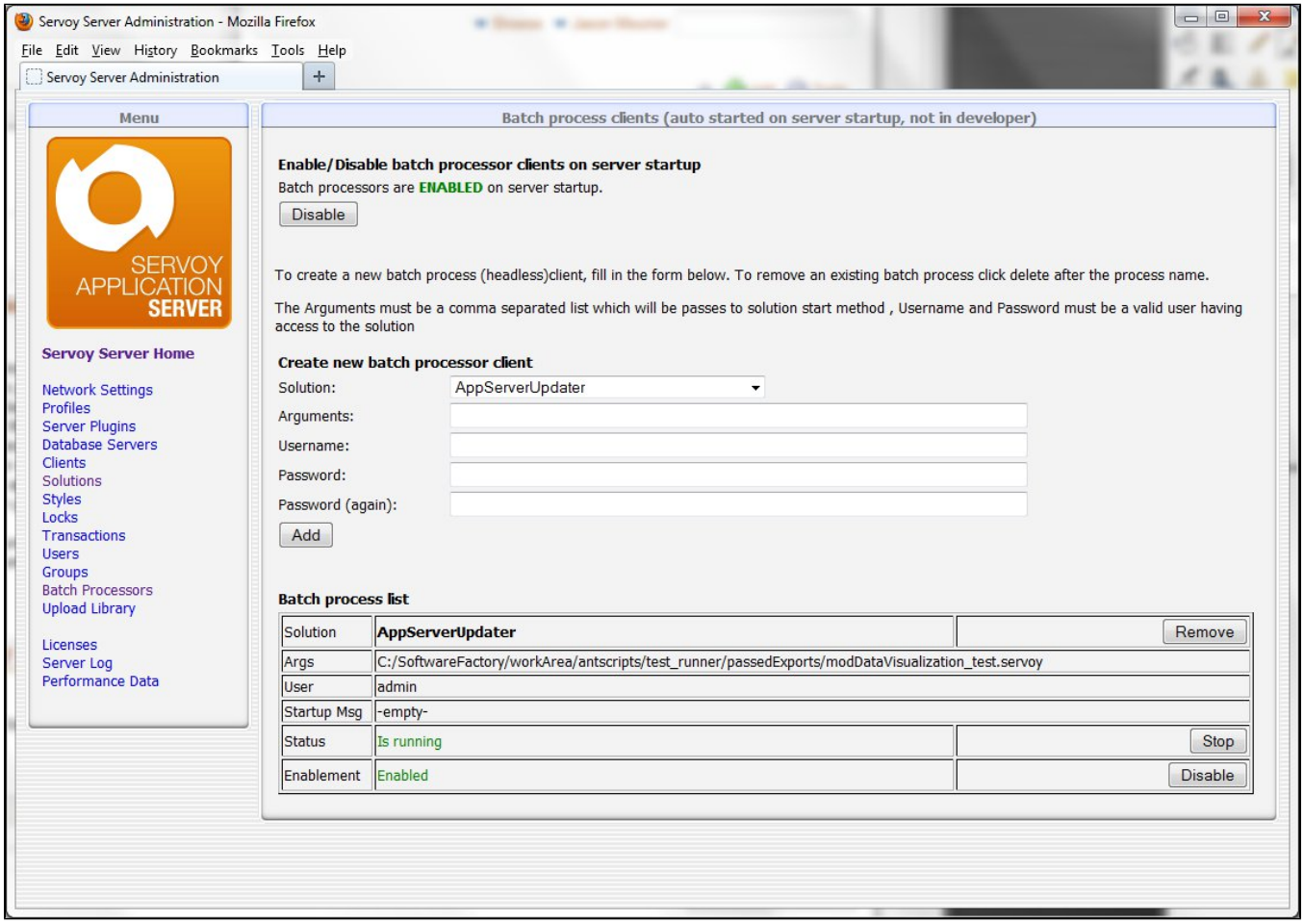
Once we have a proper `.servoy` file and it has passed unit testing (via our continuous build software), we would like to take it one step further and make the build available to the team for testing and use. This section talks about how to setup a Servoy Application Server to automatically import solutions.

Servoy Application Server Solution Import

At this time, there is no built in functionality to import `.servoy` files, nor is there any API available in the maintenance plug in for this functionality. It is possible to post to an application server and import a solution in that manner. An example importer solution is available at https://svn.servoy.com/examples/SoftwareFactoryUtils/trunk/AppServerUpdater

The importer solution will use the post the export file to the application server as if it were a web service. The solution import occurs, and the solution becomes available to the clients of that web server immediately. The import is triggered by changes to a `.servoy` file; when the file is changed via the build server process, an import occurs.

To use this solution, it is placed on an application server as ran as a batch processor. It takes one argument, the path and name of the file to import. A batch processor is started for each file to watch for changes.



The solution is open source and configurable. If any changes need to be made on how it operates, just open it in Servoy Developer and make the necessary changes.

One change needed will be the Application Server address which the import is desired to occur on. Change the `appServerAddress` variable in the importer scope to the correct address and port of the Application Server.

Another change is in the `solutionOpen` method. Add the FileWatcher license code, available at http://servoy-plugins.de

Other changes can be made to the `importSolution` method to adjust import settings if necessary.

Building the Servoy Software Factory

This section is a guide to setting up a Servoy software factory.

Continuous Build Server

The following information is a step by step guide to building a continuous build server. It includes a sample configuration for a set of sample solutions, as well as instructions on where to change this information for solutions.

Step 1 - Install/Configure Continuous Build Software

Perform all steps in the guide on how to install Jenkins and configure it to automatically build and test Servoy Solutions that can be found here.

(alternatively, it is possible to set up CruiseControl using this guide - that will no longer be updated after Servoy 7.2 release)

Step 2 - Install/Start Servoy Server Importer

If this is not done yet, install another instance of Servoy as an Application Server to act as the automatic build server. This can run the batch process for updating another Servoy Application Server, or it can just update itself and use it as the build server that holds the latest version of the software ready to be tested. Be sure to configure the ports accordingly so that there are no port conflicts.