

Run

Table of contents

1 Preparing to run the sample applications.....	2
2 Execute the build and update process via Jobcenter.....	6
3 Execute via 'ctl'.....	7

Overview

This section describes how to prepare for and run a code module and data BuildAndDeploy workflow job or command and is pertinent to users responsible for releasing content changes from the source code repository and reference database to targeted deployment environments.

There are two interfaces available to execute the BuildAndDeploy commands:

- Using Jobcenter, the web-based graphical application,
- Using Ctl's `ctl` shell command

Instructions for running the BuildAndDeploy command using either interface are explained below.

1. Preparing to run the sample applications

In general, there are some one-off package installation and builder/service configuration commands to be run before normal ("day-to-day" build and deployment) operations can commence. In particular (regarding the sample applications that have been installed and configured in the previous tabs of this site) execute the following commands from the Ctl client designated as the server system:

Note:

While instructions for these one-off commands are given in terms of using the Ctl shell command it is quite possible to configure these (and any other) commands to execute from Jobcenter too.

Note:

The various builders and services that make up the sample applications are deployed under `"${CTIER_ROOT}/demo/elements"` ("`%CTIER_ROOT%\demo\elements`") by default. You can remove the demonstration applications from your system by deleting this directory and its contents.

- Deploy the Duke's Bank and "Hello, World" builders in order to install copies of Ant, JBoss and Tomcat to support the build by issuing this command on the system designated as the build box ("localhost" by default):

```
$ ctl -p default -t JBossAntBuilder -o dukesBank -c Deploy
Start: "Run the deployment cycle, coordinating package installation and
configuration." commands: Packages-Install,Configure
.
.
.
$ ctl -p default -t TomcatAntBuilder -o helloWorld -c Deploy
Start: "Run the deployment cycle, coordinating package installation and
configuration." commands: Packages-Install,Configure
```

Run

```
.  
. .  
.
```

- Checkout the Duke's Bank and "Hello, World" source code to the build box:

Note:
By default the source is checked out from Sourceforge's Subversion repository. If you would like to be able to make updates, transfer the source into your own repository and modify the development AntBuilder's BuilderScmConnection resource setting via ProjectBuilder or Workbench.

```
$ ctl -p default -t JBossAntBuilder -o dukesBank -c scmCheckout  
scmCheckout parameters:  
{basedir="/home/jboss/dukesbank/build/cruisecontrol-bin-2.7.1/projects/dukesbank",  
connection="https://moduleforge.svn.sourceforge.net/svnroot/moduleforge/elements/bra  
module="", label="{opts.label}", scmcommand="checkout" }  
. .  
.  
$ ctl -p default -t TomcatAntBuilder -o helloWorld -c scmCheckout  
scmCheckout parameters:  
{basedir="/home/jboss/ctier/demo/elements/build/cruisecontrol-bin-2.7.1/projects/hel  
connection="https://moduleforge.svn.sourceforge.net/svnroot/moduleforge/elements/bra  
module="", label="{opts.label}", scmcommand="checkout" }  
. .  
.  
.
```

- Deploy CruiseControl (automatically configuring the Duke's Bank and "Hello, World" builders as projects) by issuing the following command on the build box:

```
ctl -p default -t CruiseControl -o elements -c Deploy  
begin workflow command (1/4) -> "Stop " ...  
begin workflow command (1/1) -> "assertServiceIsDown " ...  
. .  
.  
.
```

Note:
The CruiseControl instance should be available from build box at <http://localhost:8081> or similar (depending on your node setup). In addition you'll find a "boot.log" and "cruisecontrol.log" in the CruiseControl installation directory.

Note:
As soon as CruiseControl comes up for the first time it will kick-off an initial build and put the resultant package into the ControlTier server's package repository.

- As a general rule, you can find an object's configuration properties (for example, in this case, the Cruisecontrol installation directory - ccDir) as follows:

```

$ ctl -p default -t CruiseControl -o elements -c Properties
DEBUG: Properties#execute ...
[MULTI_LINE]
# elements [CruiseControl] #

Elements module library continuous integration server

## Attributes ##

* basedir:
"/Users/jboss/ctier/demo/elements/build/cruisecontrol-bin-2.7.1/projects/dukesBank"
* ccDir:
"/Users/jboss/ctier/demo/elements/build/cruisecontrol-bin-2.7.1"
* cruiseControlInterval: "300"
* cruiseControlJavaHome: "${env.JAVA_HOME}"
* cruiseControlJmxPort: "8001"
* cruiseControlMailHost: "localhost"
* cruiseControlPackageBase: "cruisecontrol-bin-2.7.1"
* cruiseControlPath: "/bin:/usr/bin"
* cruiseControlRmiPort: "1100"
* cruiseControlThreads: "1"
* cruiseControlWebPort: "8081"
.
.
.

```

- Install and configure the development and staging Duke's Bank JBoss and HSQLDB database instances and "Hello, World" Tomcat instances executing these commands from the ControlTier server node:

```

$ ctl -p default -t Site -o developmentDukesBank -c Deploy
Start: "Run the coordinated deployment cycle across the configured
Sites." commands: dispatchCmd
begin workflow command (1/1) -> "dispatchCmd -command Deploy
-resourcetype [^\.]* -resourcename .*" ...
dispatching command: "Deploy " to: (JBossServer) developmentDukesBank,
(HsqldbRdb) developmentDukesBank ...
.
.
.
$ ctl -p default -t Site -o developmentHelloWorld -c Deploy
Start: "Run the coordinated deployment cycle across the configured
Sites." commands: dispatchCmd
begin workflow command (1/1) -> "dispatchCmd -command Deploy
-resourcetype [^\.]* -resourcename .*" ...
dispatching command: "Deploy " to: (TomcatServer) developmentHelloWorld
...
.
.
.
$ ctl -p default -t Site -o stagingDukesBank -c Deploy
Start: "Run the coordinated deployment cycle across the configured

```

Run

```
Sites." commands: dispatchCmd
begin workflow command (1/1) -> "dispatchCmd -command Deploy
-resourcetype [^\.]* -resourcename .*" ...
dispatching command: "Deploy " to: (JBossServer) stagingDukesBank,
(HsqldbRdb) stagingDukesBank ...
.
.
.
$ ctl -p default -t Site -o stagingHelloWorld -c Deploy
Start: "Run the coordinated deployment cycle across the configured
Sites." commands: dispatchCmd
begin workflow command (1/1) -> "dispatchCmd -command Deploy
-resourcetype [^\.]* -resourcename .*" ...
dispatching command: "Deploy " to: (TomcatServer) stagingHelloWorld ...
.
.
.
```

The initial development and staging Duke's Bank JBoss instances are available from the application server box(es) at <http://localhost:8180> and <http://localhost:8280>, or similar (depending on your node setup). You'll find the JBoss "boot.log" and "server.log" under the "default" server instance in the JBoss installation directory for each environment. e.g:

```
$ cd ${CTIER_ROOT}/demo/elements
$ ls */dukesbank/jboss-4.0.3SP1/server/default/log/server.log
development/dukesbank/jboss-4.0.3SP1/server/default/log/server.log
staging/dukesbank/jboss-4.0.3SP1/server/default/log/server.log
```

Note:

The JBossServer type uses the "spawn" attribute of the [Exec](#) Ant task. As a result, the console output of the JBoss "run.sh" is always lost.

The development and staging "Hello, World" Tomcat instances are available at <http://localhost:8082> and <http://localhost:8083> (again depending on your node setup). You'll find the Tomcat "catalina.out" files in each instance's logs directory. e.g.:

```
$ cd ${CTIER_ROOT}/demo/elements
$ ls */helloworld/apache-tomcat-6.0.14/logs/catalina.out
development/helloworld/apache-tomcat-6.0.14/logs/catalina.out
staging/helloworld/apache-tomcat-6.0.14/logs/catalina.out
```

To recap, execution of these deployment commands completes configuration of:

- The build box hosting both the CruiseControl continuous integration service and the Duke's Bank JBossAntBuilder and "Hello, World" TomcatAntBuilder.
- The development application server box hosting JBoss and Tomcat.

- The development database server box hosting HSQLDB.
- The staging application server box hosting JBoss and Tomcat.
- The staging database server box hosting HSQLDB.

Along with the server box hosting the ControlTier server (Workbench, WebDAV & Jobcenter) you may have deployed software to as many as six boxes (or as few as one).

Note:

By this stage you will have seen what may appear to be several redundant installations of the same software be executed (Java, ATG, JBoss, Tomcat). These are necessary in order to ensure each of the application "service" instance deployments can operate completely independantly of one another.

2. Execute the build and update process via Jobcenter

In [Configure Step #3](#), new jobs will have been defined for the BuildAndUpdate process for each application. While these jobs represents the end-to-end automation of the build and deployment process, a broad set of additional jobs show how Jobcenter can be a general purpose operational console. After logging into Jobcenter the list of the defined jobs will be displayed hierarchically by environment and application.

For example, choosing and running one of the BuildAndUpdate jobs in the Build folder will execute the underlying Ctl BuildAndUpdate command for that application.

Considering the specific set of jobs loaded to Jobcenter as part of the sample applications:

- Locate and select the Duke's Bank "BuildAndUpdate" job in the "Build" folder and "Run Job Now"

Note:

The "buildstamp" is equivalent to the "release version" of the application and is central to ControlTier's package naming scheme. Running the BuildAndUpdate command (and the the Build command from CruiseControl) automatically sets the buildstamp associated with the builder to something similar to "trunk-1.2.3.31"; a standard tag and major/minor/release/revision number scheme implemented by the ControlTier Builder module. This scheme can be adjusted in a number of ways and can be completely replaced by your site convention as necessary.

Note:

Contraints on how the Duke's Bank BuildAndUpdate command changes package assignments restrict the command to updating the JBoss ear package version. The development HSQLDB instance is deployed using a pre-assigned database dump file package version.

- You can follow the job's progress using the "Tail Output" view in Jobcenter
- Once the job has succeeded there will a JBossEar package in the package repository (almost certainly the same one as that generated by the CruiseControl build earlier), and the development environment will running this new build.

Run

- Finally, find the Duke's Bank "Update" job in the "Staging/DukesBank/JBossServer" folder and "Choose Options and Run Job..." filling in the buildstamp (e.g. "trunk.1.2.3.31") before hitting "Run Job".

Note:

This demonstrates the other common mode of operation where a runtime environment (in this case "staging") is updated with a designated pre-built package from the ControlTier package repository.

At this stage the development environment is up and running the Duke's Bank sample application: <http://localhost:8180/bank/main>, and the staging version of the application is at <http://localhost:8280/bank/main>, or similar (depending on your node setup). You can login to either site using the username "200" and the password "j2ee".

Running the corresponding jobs for the "Hello, World" application will deploy that application to the development (<http://localhost:8082/myapp>) and staging (<http://localhost:8083/myapp>) environments too.

3. Execute via 'ctl'

An alternative to executing these commands via Jobcenter is to execute them directly via the `ctl` shell command. The general usage is shown below:

```
$ ctl -p project -t Updater -o name -c BuildAndDeploy -- \  
-buildstamp buildstamp
```

A typical convention is to use the date and time as the `-buildstamp` argument. For example:

```
$ ctl -p project -t Updater -o name -c BuildAndDeploy -- \  
-buildstamp 200711071500
```

If you are an experienced Ctl user, you may also know how to run individual parts of the build and update workflow by running the appropriate command from one of the subordinate commands. For example, to run just the Deploy:

```
$ ctl -p project -t Updater -o name -c Deploy
```

Or to run just the build, you can execute the Build workflow separately:

```
$ ctl -p project -t Updater -o name -c Build -- \  
-buildstamp buildstamp
```