IBM Software Group

Using Headless Build ANT Script to Generate Deployable EAR File for Server Deployment

Richard Gregory (gregoryr@ca.ibm.com)
Software Developer, WebSphere BPM
1 November 2011

WebSphere® Support Technical Exchange

ON DEMAND BUSINESS™

Agenda

- Automated builds in WID vs WPS
- WID build script and Ant tasks
- Running Ant scripts using WID
- Automated component testing
- WPS build script and WPS/WAS Ant tasks
- Running Ant scripts using WPS
- Additional info on serviceDeploy

Automating builds

- Two fundamental approaches to building and deploying modules using Ant
  - Using headless WID
    - Same as building, exporting using WID workbench
  - Using WPS serviceDeploy

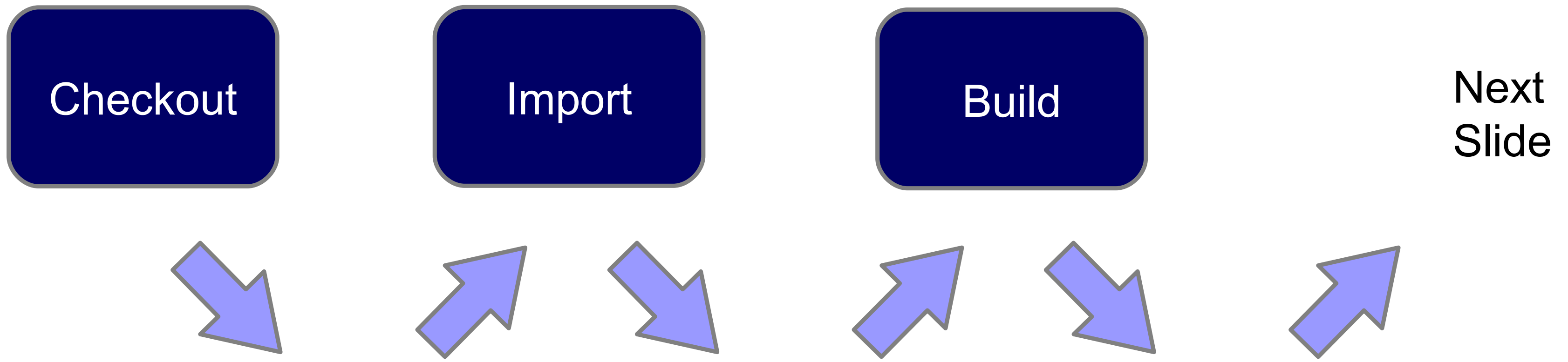
Why two approaches?

- Recommended: serviceDeploy
  - Simpler
  - Intended to be the command-line tool for packaging SCA applications prior to deployment
- Headless WID
  - Avoids limitations of serviceDeploy
    - Some generated artifacts can only be created using WID builds

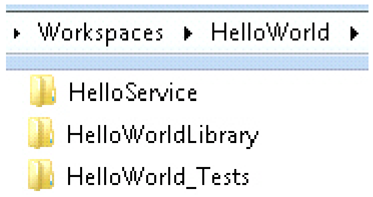
Basic WID build script

- Checkout source projects from repository
- Import projects into workspace
- Build projects in workspace
- Deploy projects to server
- Run unit tests against deployed projects

Basic WID build script



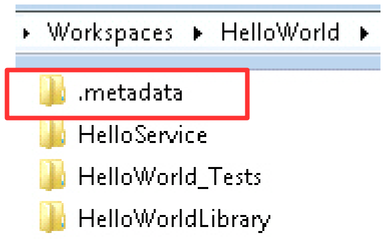
Projects in file system



Workspaces ▸ HelloWorld ▸

HelloService
HelloWorldLibrary
HelloWorld_Tests

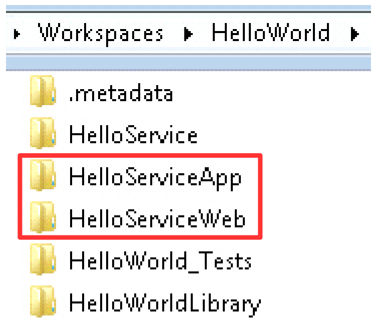
Projects in workspace



Workspaces ▸ HelloWorld ▸

.metadata
HelloService
HelloWorld_Tests
HelloWorldLibrary

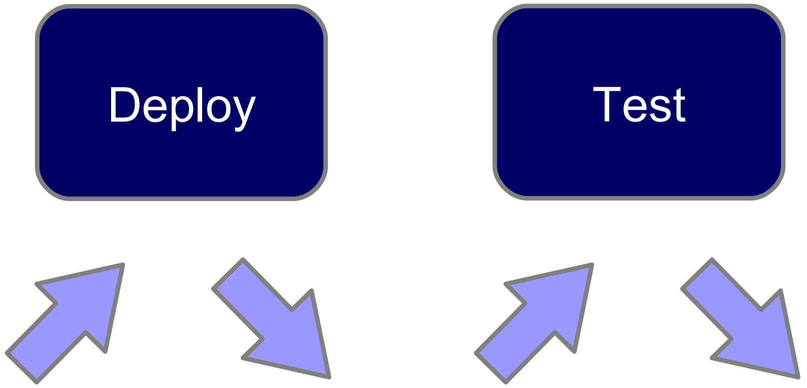
Projects built



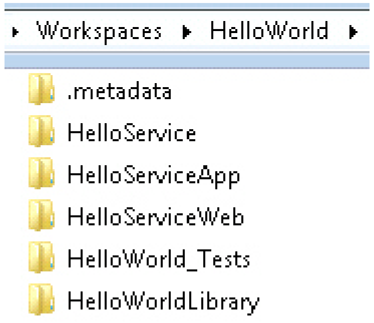
Workspaces ▸ HelloWorld ▸

.metadata
HelloService
HelloServiceApp
HelloServiceWeb
HelloWorld_Tests
HelloWorldLibrary

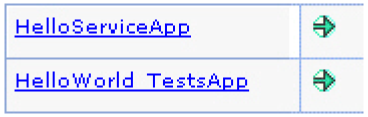
Basic WID build script



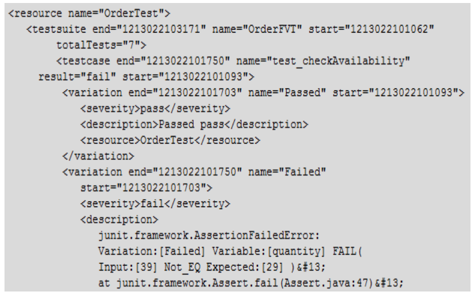
Projects built



Projects on server



Test results



```
<resource name="OrderTest">
   <testsuite end="1213022103171" name="OrderFVT" start="1213022101062"
        totalTests="7">
        <testcase end="1213022101750" name="test_checkAvailability"
     result="fail" start="1213022101093">
         <variation end="1213022101703" name="Passed" start="1213022101093">
            <severity>pass</severity>
            <description>Passed pass</description>
            <resource>OrderTest</resource>
         </variation>
         <variation end="1213022101750" name="Failed"
            start="1213022101703">
            <severity>fail</severity>
            <description>
               junit.framework.AssertionFailedError:
               Variation:[Failed] Variable:[quantity] FAIL(
               Input:[39] Not_EQ Expected:[29] )&#13;
               at junit.framework.Assert.fail(Assert.java:47)&#13;
```

Simple WID Ant Script: MyBuild.xml

```
<target name="checkout">
   <cvs command="export -r ${cvs.stream} -d ${library}" cvsroot="${cvsroot}" cvsrsh="ssh" dest="${extract
   <cvs command="export -r ${cvs.stream} -d ${module}" cvsroot="${cvsroot}" cvsrsh="ssh" dest="${extract.
   <cvs command="export -r ${cvs.stream} -d ${testproj}" cvsroot="${cvsroot}" cvsrsh="ssh" dest="${extrac
</target>

<target name="importProject" depends="checkout">
   <importProject projectName="${library}"/>
   <importProject projectName="${module}"/>
   <importProject projectName="${testproj}"/>
</target>

<target name="build" depends="importProject">
   <projectBuild projectName="${library}"/>
   <projectBuild projectName="${module}"/>
   <projectBuild projectName="${testproj}"/>
</target>

<target name="deploy" depends="build">
   <wid.deploy projectName="${library}" userid="${user}" password="${password}" profile="${profile}" conn
   <wid.deploy projectName="${module}" userid="${user}" password="${password}" profile="${profile}" conne
   <wid.deploy projectName="${testproj}" userid="${user}" password="${password}" profile="${profile}" con
</target>

<target name="run" depends="deploy">
   <get dest="${testfile}" src="${url}"/>
   <wid.undeploy projectName="${testproj}"/>
   <wid.undeploy projectName="${module}"/>
   <wid.undeploy projectName="${library}"/>
</target>
```

Simple WID Ant Script

- Many script variations
  - For example, iterate over a list of project names
    - Project names to build stored in property file
    - Projects to build computed from dependencies

WID Ant tasks

- Checkout source projects
  - Tasks such as <cvs> bring projects into file system

```
<cvs command="export -r ${cvs.stream} -d ${module}"
     cvsroot="${cvsroot}"
     cvsrsh="ssh"
     dest="${extract.dir}"
     package="TestCase/${module}"
     output="${module.log.filename}.log"
     quiet="true"/>
```

WID Ant tasks

- Import projects into workspace: <importProject>
  - Brings projects into WID workspace
  - Necessary if project folders were not previously part of a workspace

```
<importProject projectName="${module}"/>
```

- Alternative: <importPI>
  - Brings projects into the workspace from a project interchange file

WID Ant tasks

- Build projects: <projectBuild>
  - Equivalent to building using WID workbench

```
<projectBuild projectName="${module}"/>
```

WID Ant tasks

- Deploy modules to server: <wid.deploy>
  - Equivalent to adding to the Servers view
  - Starts the server if needed
  - Starts the application after installing
  - To avoid a plain text password in the script, see wsadmin command reference

```
<wid.deploy projectName="${module}"
            userid="${user}"
            password="${password}"
            profile="${profile}"
            connectionType="${connection.type}"
            port="${connection.port}"/>
```

WID Ant tasks

- Automated testing: <get>
  - Http call to Testcase servlet in test project
  - Puts results in a file
  - URL depends on what is being tested.
    - See automated component test slides

```
<get dest="${testfile}" src="${url}"/>
```

Running Ant scripts using WID

- Command to launch headless WID

```
java.exe %VMARGS% -cp %STARTUP_JAR%
org.eclipse.core.launcher.Main -application
com.ibm.wbit.comptest.ant.RunAntWid -buildFile
MyBuild.xml
```

  - VMARGS: same values as WID\eclipse.ini
  - STARTUP_JAR: WID\plugins\org.eclipse.equinox.launcher.jar
- See WID\bin\runAntWid.bat
- Use runAntWID as-is or customize as needed

Running Ant scripts using WID

- Using runAntWID to launch headless WID

```
C:\WID7\bin>set WORKSPACE=C:\Workspaces\HelloWorld
C:\WID7\bin>runAntWid.bat -buildfile C:\Builds\BuildHWSample.xml
```

- Don't use runAnt because it may result in build failures
  - runAntWid sets up a different classpath needed for SCA projects

Running Ant scripts using WID

- Troubleshooting: investigate headless build errors
  - Try opening the workspace used in the Ant script with WID
    - Turn off auto builds to preserve previous build state
  - Compare a workspace built by WID with one built using Ant script
    - Differences may give hint to cause

Running Ant scripts using WID

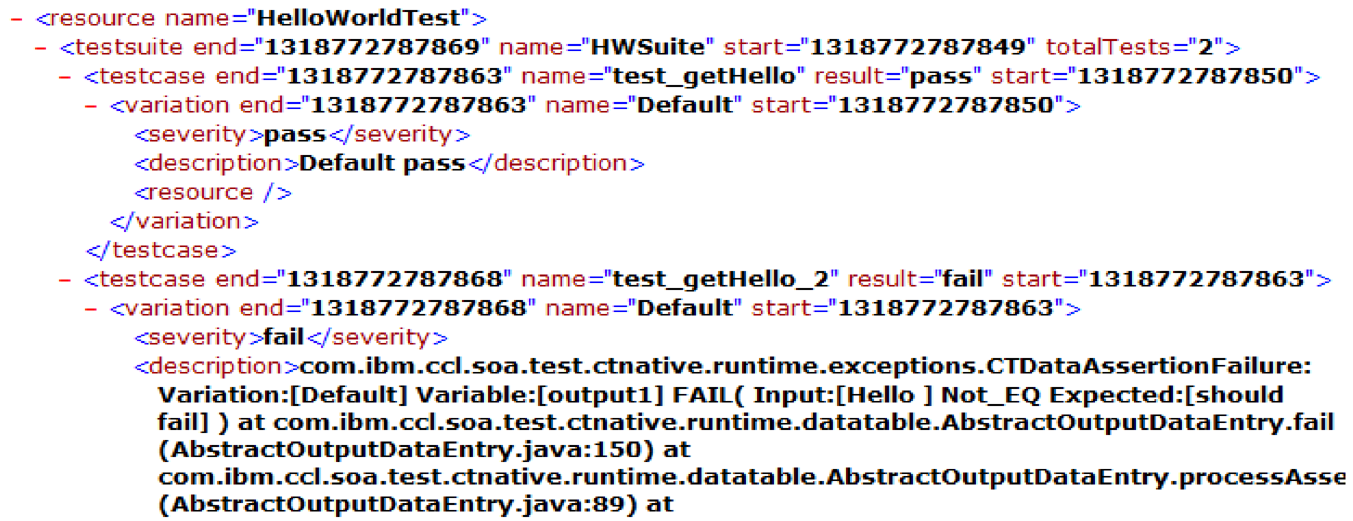
- Keep scripts simple as possible to isolate problems
- Other RAD Ant tasks generally work
  - ImportProjectSet has been reported to cause problems (WID index not populated properly)

Automated component tests

- To run component tests after deploying:
  - http://*hostname*:*port*/*TestProjectName*Web/TestServlet
- To run individual test suites or test cases:
  - TestServlet?suite=*TestSuiteName*
  - TestServlet?suite=*TestSuiteName*&testcases=*testcase1*,*testcase2*
- To provide security credentials
  - TestServlet?username=*username*&password=*password*

Automated component tests

- Results returned as an XML string
  - Not currently in standard JUnit format, but users typically use XSLT to use with JUnit tools
- Example results

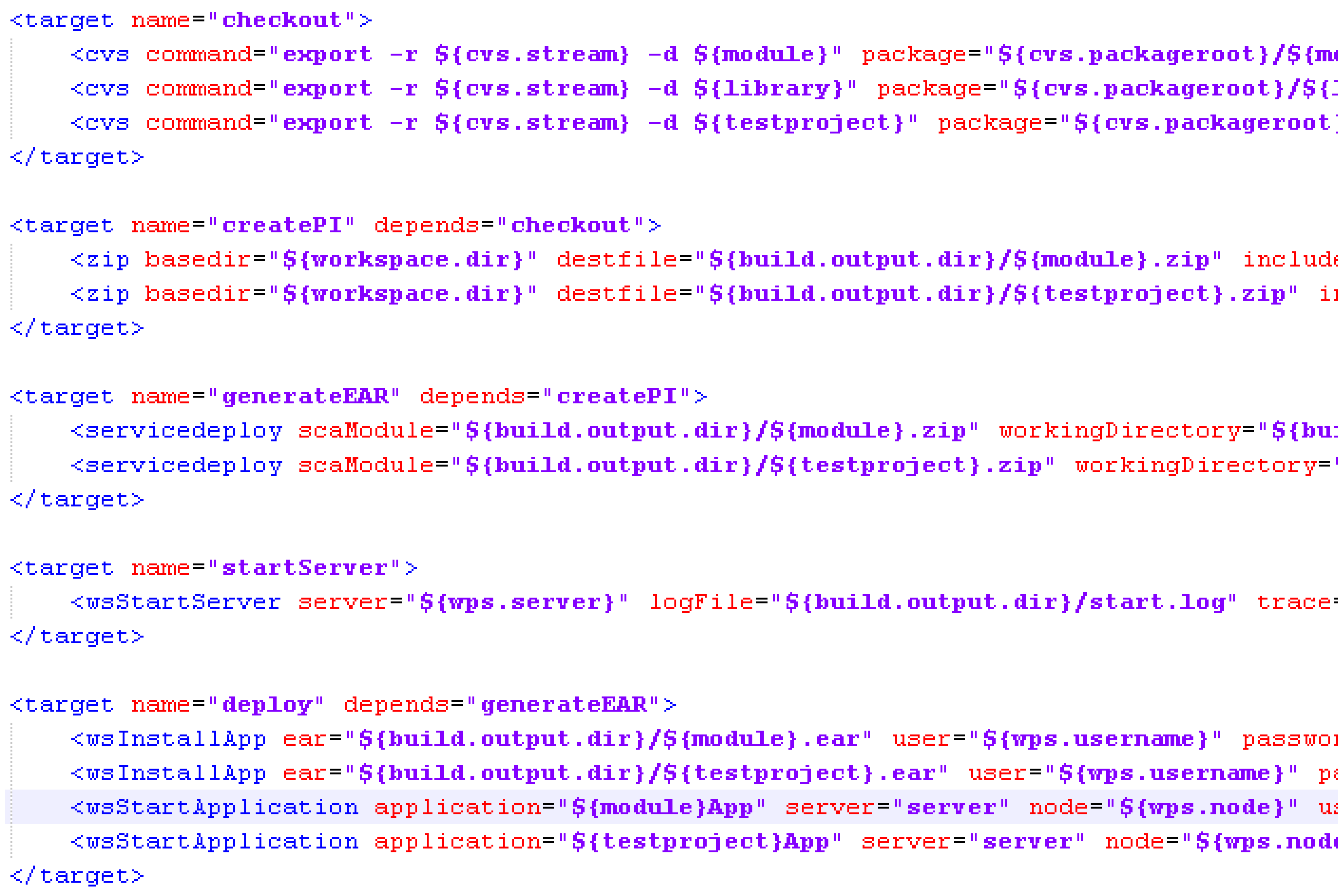


```
- <resource name="HelloWorldTest">
  - <testsuite end="1318772787869" name="HWSuite" start="1318772787849" totalTests="2">
    - <testcase end="1318772787863" name="test_getHello" result="pass" start="1318772787850">
      - <variation end="1318772787863" name="Default" start="1318772787850">
          <severity>pass</severity>
          <description>Default pass</description>
          <resource />
        </variation>
      </testcase>
    - <testcase end="1318772787868" name="test_getHello_2" result="fail" start="1318772787863">
      - <variation end="1318772787868" name="Default" start="1318772787863">
          <severity>fail</severity>
          <description>com.ibm.ccl.soa.test.ctnative.runtime.exceptions.CTDataAssertionFailure:
            Variation:[Default] Variable:[output1] FAIL( Input:[Hello ] Not_EQ Expected:[should
            fail] ) at com.ibm.ccl.soa.test.ctnative.runtime.datatable.AbstractOutputDataEntry.fail
            (AbstractOutputDataEntry.java:150) at
            com.ibm.ccl.soa.test.ctnative.runtime.datatable.AbstractOutputDataEntry.processAsse
            (AbstractOutputDataEntry.java:89) at
```

Simple WPS Ant script

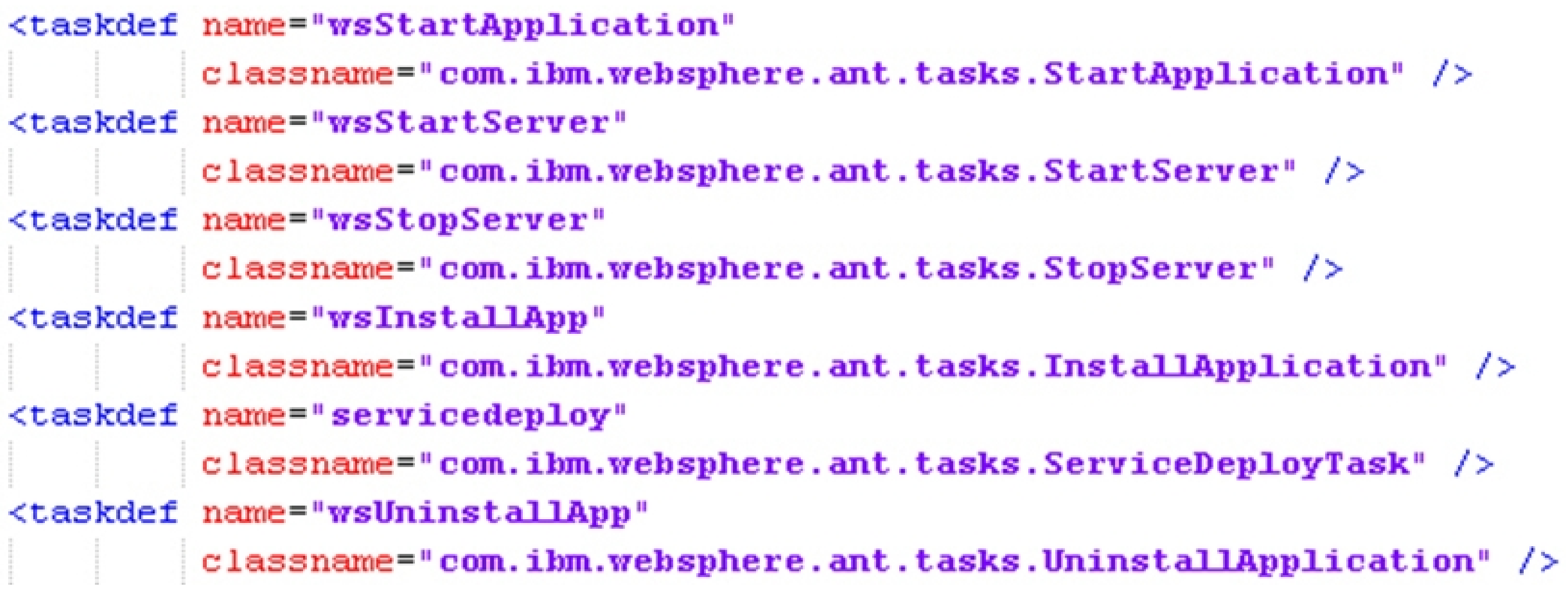
- Checkout source projects from repository
- Zip projects
- Create deployable applications using serviceDeploy
- Install and start applications on server
- Run unit tests against deployed projects

Simple WPS Ant script



```
<target name="checkout">
    <cvs command="export -r ${cvs.stream} -d ${module}" package="${cvs.packageroot}/${m
    <cvs command="export -r ${cvs.stream} -d ${library}" package="${cvs.packageroot}/${
    <cvs command="export -r ${cvs.stream} -d ${testproject}" package="${cvs.packageroot
</target>

<target name="createPI" depends="checkout">
    <zip basedir="${workspace.dir}" destfile="${build.output.dir}/${module}.zip" include
    <zip basedir="${workspace.dir}" destfile="${build.output.dir}/${testproject}.zip" i
</target>

<target name="generateEAR" depends="createPI">
    <servicedeploy scaModule="${build.output.dir}/${module}.zip" workingDirectory="${bu
    <servicedeploy scaModule="${build.output.dir}/${testproject}.zip" workingDirectory='
</target>

<target name="startServer">
    <wsStartServer server="${wps.server}" logFile="${build.output.dir}/start.log" trace
</target>

<target name="deploy" depends="generateEAR">
    <wsInstallApp ear="${build.output.dir}/${module}.ear" user="${wps.username}" passwo
    <wsInstallApp ear="${build.output.dir}/${testproject}.ear" user="${wps.username}" p
    <wsStartApplication application="${module}App" server="server" node="${wps.node}" u
    <wsStartApplication application="${testproject}App" server="server" node="${wps.nod
</target>
```

WPS and WAS Ant tasks



```
<taskdef name="wsStartApplication"
         classname="com.ibm.websphere.ant.tasks.StartApplication" />
<taskdef name="wsStartServer"
         classname="com.ibm.websphere.ant.tasks.StartServer" />
<taskdef name="wsStopServer"
         classname="com.ibm.websphere.ant.tasks.StopServer" />
<taskdef name="wsInstallApp"
         classname="com.ibm.websphere.ant.tasks.InstallApplication" />
<taskdef name="servicedeploy"
         classname="com.ibm.websphere.ant.tasks.ServiceDeployTask" />
<taskdef name="wsUninstallApp"
         classname="com.ibm.websphere.ant.tasks.UninstallApplication" />
```

- Note: When using runAntWid, taskdefs are not necessary

WPS and WAS Ant tasks

- Checkout source projects (Same as WID script)
  - Tasks such as <cvs> bring projects into file system

```
<cvs command="export -r ${cvs.stream} -d ${module}"
     cvsroot="${cvsroot}"
     cvsrsh="ssh"
     dest="${extract.dir}"
     package="TestCase/${module}"
     output="${module.log.filename}.log"
     quiet="true"/>
```

WPS and WAS Ant tasks

- Zip projects: <zip>
  - Input to serviceDeploy is a project interchange file
    - One PI file per module
  - Project interchange file is a zip of the source project and dependencies

```
<zip basedir="${workspace.dir}"
     destfile="${build.output.dir}/${module}.zip"
     includes="${module}/**/**/*, ${library}/**/**/*"
     excludes="**/CVS/**" />
```

WPS and WAS Ant tasks

- Create deployable application: <serviceDeploy>

```
<servicedeploy scaModule="${build.output.dir}/${module}.zip"
               workingDirectory="${build.working.dir}"
               outputApplication="${build.output.dir}/${module}.ear"
               wasHome="${wps.home}"
               cleanStagingModules="true"
               keep="true" />
```

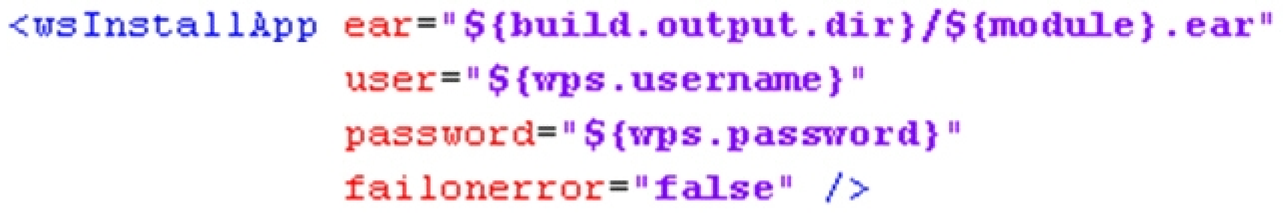
WPS and WAS Ant tasks

- Optional: Start the server: <wsStartServer>

```
<wsStartServer server="${wps.server}"
               logFile="${build.output.dir}/start.log"
               trace="false"
               failonerror="false" />
```

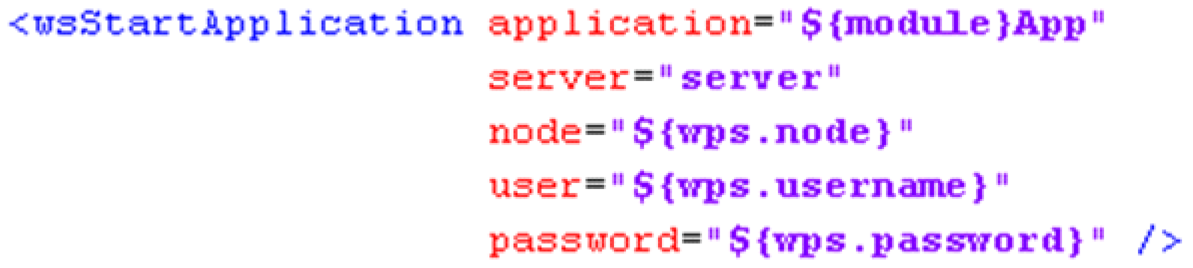
WPS and WAS Ant tasks

- Install and start applications
  - <wsInstallApp>



```
<wsInstallApp ear="${build.output.dir}/${module}.ear"
              user="${wps.username}"
              password="${wps.password}"
              failonerror="false" />
```

  - <wsStartApplication>



```
<wsStartApplication application="${module}App"
                    server="server"
                    node="${wps.node}"
                    user="${wps.username}"
                    password="${wps.password}" />
```

WPS and WAS Ant tasks

- Automated testing: <get>
  - Same as WID Ant script
- Note: do not call WPS or WAS Ant tasks when running Ant script using WID
  - Need to run in their own JVM

Running Ant scripts using WPS

- Command to launch ant script using WPS

  ```
  ws_ant.bat -f MyBuildScript.xml
  ```

- Example

  -
    ```
    C:\WID75\runtimes\bi_v7\bin>ws_ant.bat
    -f \Workspaces\BuildHWSampleSD.xml
    ```

Running Ant scripts using WPS

- Troubleshooting: if problems occur try
  - Compare EARs on server
    - Deployed from WID vs installed from serviceDeploy
  - Compare EARs generated
    - From WID vs from serviceDeploy

ServiceDeploy limitations

- Before V7.0:
  - Component Test Projects not recognized
  - Java™ code not generated for
    - Custom mediations and maps
    - Adapter bindings
- Starting from 7.0.0.3
  - Adapter binding Java code not generated

Building projects before and after v7.0

- Prior to 7.0, SCA projects were built in WID or serviceDeploy
  - Installing an app on the server only created general J2EE artifacts

Building projects before and after v7.0

- 7.0 and beyond, SCA projects are build during app install.
  - WID does some building (e.g. Maps can be tested when not running on the server) but not necessary for deployment
  - ServiceDeploy simply packages projects as an EAR that is ready for install

Summary

- Covered differences between running WPS and WID ant scripts
- Examples of each type of script with descriptions of individual Ant tasks
- Running automated component tests
- Troubleshooting

References

- Automated builds: WID documentation:
  http://publib.boulder.ibm.com/infocenter/esbsoa/wesbv7r5/index.jsp?topic=%2Fcom.ibm.wbpm.wid.admin.doc%2Ftopics%2Ftscripttest.html
- ServiceDeploy: WPS documentation:
  http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r0mx/index.jsp?topic=%2Fcom.ibm.websphere.wps.doc%2Fdoc%2Frdev_servicedeploy.html
- ServiceDeploy Ant Task documentation:
  http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r0mx/topic/com.ibm.websphere.wps.doc/doc/tdep_usingant.html
- wsadmin command reference:
  http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/rxml_commandline.html

Additional WebSphere Product Resources

- Learn about upcoming WebSphere Support Technical Exchange webcasts, and access previously recorded presentations at: http://www.ibm.com/software/websphere/support/supp_tech.html
- Discover the latest trends in WebSphere Technology and implementation, participate in technically-focused briefings, webcasts and podcasts at: http://www.ibm.com/developerworks/websphere/community/
- Join the Global WebSphere Community: http://www.websphereusergroup.org
- Access key product show-me demos and tutorials by visiting IBM® Education Assistant: http://www.ibm.com/software/info/education/assistant
- View a webcast replay with step-by-step instructions for using the Service Request (SR) tool for submitting problems electronically: http://www.ibm.com/software/websphere/support/d2w.html
- Sign up to receive weekly technical My Notifications emails: http://www.ibm.com/software/support/einfo.html

Connect with us!

1. **Get notified on upcoming webcasts**
   Send an e-mail to wsehelp@us.ibm.com with subject line "wste subscribe" to get a list of mailing lists and to subscribe
2. **Tell us what you want to learn**
   Send us suggestions for future topics or improvements about our webcasts to wsehelp@us.ibm.com
3. **Be connected!**
   Connect with us on Facebook
   Connect with us on Twitter

Questions and Answers