Eusko Jaurlaritzaren Informatika Elkartea
Sociedad Informática del Gobierno Vasco

Manual de tareas ant – WL11

Fecha: 12/11/2010

Referencia:

EJIE S.A.
Mediterráneo, 14
Tel. 945 01 73 00*
Fax. 945 01 73 01
01010 Vitoria-Gasteiz
Posta-kutxatila / Apartado: 809
01080 Vitoria-Gasteiz
www.ejie.es

Eusko Jaurlaritzaren Informatika Elkartea
Sociedad Informática del Gobierno Vasco

Control de documentación

Título de documento:	

Histórico de versiones

Código	Versión	Fecha	Resumen de cambios
	1.0	12/11/2011	

Cambios producidos desde la última versión

Control de difusión

Responsable:	Ander Martínez
Aprobado por:	
Firma:	Fecha:
Distribución:	

Referencias de archivo

Autor:	CAC
Nombre archivo:	
Localización:	

Contenido

Introducción

El objetivo principal de este documento es informar sobre las diferentes tareas de ant que se pueden utilizar en el entorno de EJIE para Weblogic 11.

Este documento recoge la información referente a tareas de ant proveniente de los siguientes documentos:

Nombre	Fichero	Descripción
Manual de tareas ant	Manual de tareas de ant.doc	Se extraen de este documento aquellas tareas ant para Weblogic 8 que también funcionan en Weblogic 11.
Oracle Weblogic Server 11 – Normativa de albergue de aplicaciones	WLS11_NormasAlbergue.doc	Tareas ant para la gestión de aplicaciones en Weblogic 11 en el entorno de EJIE: compilación, despliegue, recursos, etc.
ERPI. Normativa de desarrollo de servicios	Normativa_desarrollo_ERPI_Servicios v1 1.doc	Tareas relacionadas con la normativa de desarrollo de servicios para la nueva plataforma de integración.
Dashboard-EJIE wl11 – Manual de Usuario	Dashboard-EJIE wl11. Manual de usuario v1.1.doc	Tareas relacionadas con la instalación, configuración y procedimiento de uso de Dashboard-EJIE

Configuración de ant

La versión ANT utilizada es la versión 1.7.1.

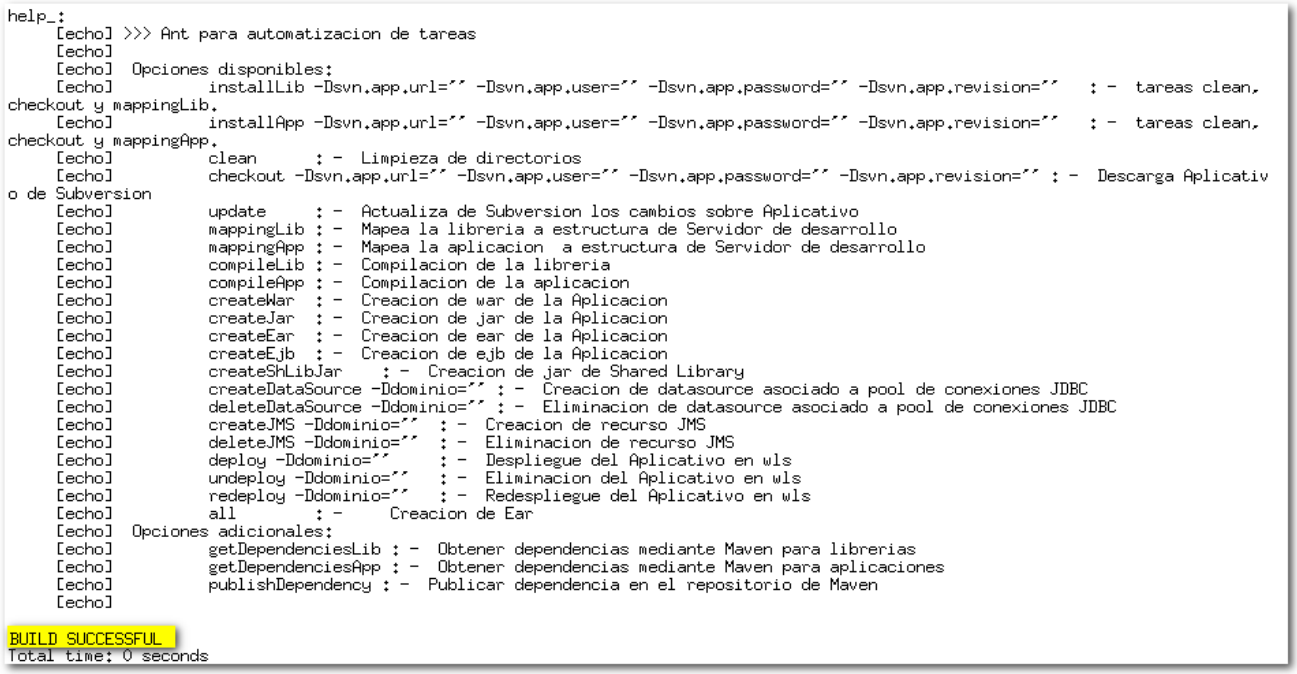
Los scripts ANT para Weblogic 11 se encuentran en la ruta `/j2se/apache-ant-1.7.1/ejie_wls11`. Dentro de esta ruta podemos encontrar el siguiente contenido:

```
>--- j2se/apache-ant-1.7.1/ejie_wls11
        --- build_wls11.xml
        --- build_ejie.xml
        >--- scripts
              --- build-ejie-compilacion.xml
              --- build-ejie-creacion-entorno.xml
              --- build-ejie-variables.xml
              --- build-ejie-dependencies.xml
              --- build-ejie-recursos.xml
              --- build-ejie-despliegue.xml
        >--- jelly
              --- build-ejie-mapping-ear.jelly
              --- build-ejie-create-war.jelly
              --- build-ejie-dependencies-war.jelly
              --- build-ejie-create-ejb.jelly
        >--- python
              --- delete_DataSource.py
              --- create_DataSource.py
              --- delete_JMS.py
              --- create_JMS.py
```

El fichero `build_wls11.xml` contiene la fachada con todas las tareas proporcionadas para facilitar las tareas de compilación y despliegue. Este fichero deberá ser copiado con el nombre `build.xml` en el directorio **`/aplic/bbb/ant`** de las aplicaciones que queremos compilar y desplegar, donde bbb sería sustituído por el código de la aplicación.

```
cp /aplic/build_wls11.xml /aplic/bbb/ant/build.xml
```

Para comprobar que las tareas de ant están ya disponibles para su uso, una vez copiado el fichero anterior, se ejecuta el comando `ant help`, debiendo obtenerse el siguiente resultado:



Si el resultado obtenido, en vez del esperado, muestra las siguientes líneas:

```
# ant help
bash: ant: command not found
```

se debe a que ant no está correctamente configurado para el usuario. Para configurarlo se ejecuta el siguiente comando: `/<directorio-instalacion-ant>/bin/antEnv.sh`.

Entre otras operaciones este comando establece las variables ANT_DIR y JAVA_HOME que permiten ejecutar correctamente ANT.

Sintaxis de ant

Puesto que `Ant` necesita el fichero de configuración `build.xml`, para poder ejecutar las distintas operaciones de configuración hay que 'situarse' en el subdirectorio `/aplic/bbb/ant` de la aplicación.

La sintaxis es la siguiente:

```
ant [options] [target [target2 [target3] ...]]
```

Las opciones disponibles son las siguientes:

-help o **-h** imprime este mensaje.

-projecthelp o **-p** imprime los targets del proyecto.

-version, imprime la versión.

-diagnostics, imprime información que puede ser útil para detectar problemas.

-**quiet** o **-q**, imprime la información mínima.

-verbose o **–v**, imprime información adicional.

-debug o **-d** imprime información de debugging.

-emacs o **-e** produce información de logging sin adornos.

-**lib *path***, especifica un path para buscar los .jar y .class

-logfile *file* o **-l *file***, escribe el resultado de la ejecución en un fichero.

-logger *classname*, fija la clase que va a implementar el logging.

-listener *classname*, añade una instancia de clase como un proyecto de listener.

- **noinput**, no permite la entrada interactiva

-buildfile *filename* o **-file *filename*** o **-f *filename***, utiliza el buildfile especificado.

-find *file*, busca el fichero de configuración, build.xml por defecto, desde el directorio actual hacia sus directorios padre, usando el primero que encuentre.

-propertyfile *name*, carga todas las propiedades del fichero con la propiedad –D que le precede.

-inputhandler *class*, se indica la clase que manejará las peticiones de entrada.

-D*property*=value, define el valor de una propiedad para el proyecto

Targets

1. Target de ayuda

Ver WLS11_NormasAlbergue.doc

El target disponible es el siguiente:

1.1 help

Devuelve la lista de tareas disponibles que proporcionan las actuales corrientes de compilación. Para su ejecución es necesario situarse en el directorio /aplic/bbb/ant.

Ejemplo:

ant help

2. Targets para el trabajo con repositorios de código fuente

Ver WLS11_NormasAlbergue.doc

Estos targets permiten descargar el código de la aplicación desde un repositorio de código fuente y organizar la estructura de ese código descargado según la estructura que necesita el servidor para poder contener y ejecutar la aplicación.

Para su ejecución hay que situarse en el directorio **/aplic/bbb/ant**.

Los targets disponibles son los siguientes:

2.1 clean

Elimina los ficheros del entorno de trabajo que incluye los siguientes directorios: /aplic/bbb/tmp, /aplic/bbb/src, /aplic/bbb/bin y /aplic/bbb/dist.

Ejemplo:

ant clean

2.2 checkout

Descarga en la carpeta /aplic/bbb/tmp el contenido indicado en la url de Subversión para su posterior compilación y despliegue.

Ejemplo:

ant checkout

-Dsvn.app.url=svn://cvs.ejiedes.net/bbb/branch/aplic/bbb

-Dsvn.app.user=bbb

-Dsvn.app.password=bbb

-Dsvn.app.revision=HEAD

2.3 update

Descarga en la carpeta /aplic/bbb/tmp aquellos ficheros del repositorio de Subversion que hayan sido modificados. Para ejecutar esta tarea por primera vez de forma correcta es obligatorio haber llamado previamente a la tarea checkout.

Ejemplo:

ant update

2.4 mappingLib

Convierte la estructura de directorios de pc local de una librería a la estructura de servidor de desarrollo para librerías. El código organizado según la estructura de servidor de desarrollo queda en el directorio /aplic/bbb/src. El código orígen con estructura de directorios de pc local una vez terminada la reestructuración permanece en el directorio /aplic/bbb/tmp.

Ejemplo:

ant mappingLib

2.5 mappingEar

Convierte la estructura de directorios de pc local de una aplicación a la estructura de servidor de desarrollo para aplicaciones. El código organizado según la estructura de servidor de desarrollo queda en el directorio /aplic/bbb/src. El código orígen con estructura de directorios de pc local una vez terminada la reestructuración permanece en el directorio /aplic/bbb/tmp.

Ejemplo:

ant mappingEar

2.6 traspasoLib

Convierte la estructura de directorios de pc local de una librería a la estructura de servidor de desarrollo para librerías y la vuelca en el area de traspaso. El código organizado según la estructura de servidor de desarrollo queda en el directorio /traspaso/bbb/src. El código orígen con estructura de directorios de pc local una vez terminada la reestructuración permanece en el directorio /aplic/bbb/tmp.

Ejemplo:

ant traspasoLib

2.7 traspasoEar

Convierte la estructura de directorios de pc local de una aplicación a la estructura de servidor de desarrollo para aplicaciones y la vuelca en el area de traspaso. El código organizado según la estructura de servidor de desarrollo queda en el directorio /traspaso/bbb/src. El código orígen con estructura de directorios de pc local una vez terminada la reestructuración permanece en el directorio /aplic/bbb/tmp.

Ejemplo:

ant traspasoEar

2.8 installLib

Esta tarea elimina los ficheros del entorno de trabajo que pudieran existir de ejecuciones previas, descarga el código del repositorio de código fuente y realiza la organización del código descargado según la estructura de servidor de desarrollo para librerías. Para ello reune en una única tarea la ejecución de las tareas clean, checkout y mappingLib.

2.9 installApp

Esta tarea elimina los ficheros del entorno de trabajo que pudieran existir de ejecuciones previas, descarga el código del repositorio de código fuente y realiza la organización del código descargado

según la estructura de servidor de desarrollo para aplicaciones. Para ello reune en una única tarea la ejecución de las tareas clean, checkout y mappingApp.

Ejemplo:

ant installApp

-Dsvn.app.url=svn://cvs.ejiedes.net/bbb/branch/aplic/bbb

-Dsvn.app.user=bbb

-Dsvn.app.password=bbb

-Dsvn.app.revision=HEAD

3. Targets para el trabajo con repositorios de config

Ver WLS11_NormasAlbergue.doc

Estos targets permiten descargar el código de la aplicación desde un repositorio de código fuente y organizar la estructura de ese código descargado según la estructura que necesita el servidor para poder contener y ejecutar la aplicación.

Para su ejecución hay que situarse en el directorio **/aplic/bbb/ant**.

Los targets disponibles son los siguientes:

3.1 checkoutConfig

Convierte la estructura de directorios de pc local de una aplicación a la estructura de servidor de desarrollo para aplicaciones. Los properties orígen con estructura de directorios de pc local una vez terminada la reestructuración permanece en el directorio /aplic/bbb/tmp.

Ejemplo:

ant checkoutConfig

-Dsvn.app.url=' svn://cvs.ejiedes.net/bbb/branch/config/ '

-Dsvn.app.user=''bbb

-Dsvn.app.password='bbb'

-Dsvn.app.revision='' (Opcional) Si no se aporta, se obtendra del build.xml

-Ddominio='' (Opcional) Si no se aporta, se obtendra del build.xml

-Denv='desarrollo' .

3.2 mappingConfig

Mapea el directorio config a la ruta correspondiente en el servidor /config/bbb/dominio/.

ant mappingConfig:

-Ddominio='' (Opcional) Si no se aporta, se obtendra del build.xml

-Denv='desarrollo' .

4. Targets de manejo de dependencias

Ver Normativa de desarrollo Maven y Archiva.doc

Ver WLS11_NormasAlbergue.doc

Los targets para el manejo de dependencias son necesarios para obtener las librerías que hacen que la aplicación que se quiere desplegar o la librería que se quiere compartir, compile correctamente y pueda ser empaquetada para su distribución.

Para que este manejo de dependencias se realice de forma correcta son necesarios ficheros que indiquen que dependencias existen, y un fichero que diga de dónde deben descargarse.

Los ficheros que informan de las dependencias cumplen con la estructura definida por Maven para los descriptores de proyectos. Estos ficheros son conocidos como POM Files (Project Object Model) y su nombre y su cantidad depende de si se quiere compilar una librería o una aplicación.

- Si se desea obtener las dependencias de una librería debe existir un fichero pom_lib.xml.
- Si se desea obtener las dependencias de una aplicación debe existir un fichero pom_app.xml para las librerías de la propia aplicación (ear), y un fichero pom_<nombreWar>.xml por cada uno de los módulos web que contenga la aplicación.

Estos ficheros de dependencias deben estar disponibles para el correcto funcionamiento de las tareas en la siguiente ruta: /aplic/bbb/tmp/deps.

Una vez que se cumplen las premisas, para ejecutar las tareas de dependencias hay que situarse en el directorio /aplic/bbb/ant.

Se han creado nuevas tareas ant que manejan las dependencias de los proyectos con los diferentes artefactos albergados en repositorios gestionados por el gestor de repositorios de Archiva. El gestor de repositorios Archiva proporciona una interfaz grñafica para realizar búsquedas en los repositorios y permite administrar repositorios maven privados (el usuario de dichas tareas tiene que crear previamente una estrutura de directorios necesaria en /entorno/aplic/bbb/.m2, ver Normativa de desarrollo Maven y Archiva.doc).

Los nuevos targets disponibles son los siguientes:

4.1 getDependenciesLibArchiva

Descarga las dependencias de la librería de los repositorios de Archiva y las deja en la ruta /aplic/bbb/src/bbbShLibClasses/lib

Ejemplo:

ant getDependenciesLib

4.2 getDependenciesAppArchiva

Descarga las dependencias de la aplicación de los repositorios de Archiva y de los módulos web incluidos en ella. Las dependencias de la aplicación las deja en la ruta /aplic/bbb/src/bbbEAR/APP-INF/lib y las dependencias de los módulos web los deja en la ruta /aplic/bbb/src/bbbEAR/wars/<nombreWar/WEB-INF/lib de cada módulo web.

Ejemplo:

ant getDependenciesApp

4.3 publishPublicLib

Publica una librería en el repositorio de librerías compartidas de EJIE *repoAplicPublicas*.(repositorio de Archiva). Esta tarea si no se le pasa parámetros publicará la librería generada por la tarea

createShLibJar.y necesitará tener configurado correctamente el fichero pom_lib.xml en /aplic/bbb/tmp/deps.

ant publishPublicLib

En el caso de que se pretenda subir otro artefacto generado, como pudieran ser clientes generados de EJBs, etc se suministrarán parámetros adicionales:

- pomfile: fichero pom del artefacto
- jarfile: artefacto jar a publicar.

ant publishPublicLib -Dpomfile=/aplic/bbb/tmp/deps/pom_stubs.xml -Djarfile=../bbbStubsEJBs.jar

4.4 publishPrivateLib

Publica una librería en el repositorio privado de la factoría de software (Archiva). Esta tarea si no se le pasa parámetros publicará la librería generada por la tarea createShLibJar.y necesitará tener configurado correctamente el fichero pom_lib.xml en /aplic/bbb/tmp/deps.

ant publishPrivateLib

En el caso de que se pretenda subir otro artefacto generado, como pudieran ser clientes generados de EJBs, etc se suministrarán parámetros adicionales:

pomfile: fichero pom del artefacto

jarfile: artefacto jar a publicar.

ant publishPrivateLib -Dpomfile=/aplic/bbb/tmp/deps/pom_stubs.xml -Djarfile=../bbbStubsEJBs.jar

Para los que estaban haciendo uso de las tareas anteriores se han mantenido por compatibilidad pero se recomiendan usar las nuevas. La antiguas son:

4.5 getDependenciesLib

Descarga las dependencias de la librería y las deja en la ruta /aplic/bbb/src/bbbShLibClasses/lib

Ejemplo:

ant getDependenciesLib

4.6 getDependenciesApp

Descarga las dependencias de la aplicación y de los módulos web incluidos en ella. Las dependencias de la aplicación las deja en la ruta /aplic/bbb/src/bbbEAR/APP-INF/lib y las dependencias de los módulos web los deja en la ruta /aplic/bbb/src/bbbEAR/wars/<nombreWar/WEB-INF/lib de cada módulo web.

Ejemplo:

ant getDependenciesApp

4.7 publishDependency

Sube al repositorio de librerías una librería ya compilada y empaquetada. Este target solo sirve para las librerías y requiere previamente su compilación (compileLib) y empaquetado (createShLibJar). Para su correcta publicación debe estar correctamente configurado el fichero pom_lib.xml en /aplic/bbb/tmp/deps.

Ejemplo:

ant publishDependency

5. Targets de compilación y preparación

Ver WLS11_NormasAlbergue.doc

Los targets de compilación y preparación obtienen el código compilado del código fuente que se ha obtenido al ejecutar las tareas de creación de entornos. Este código compilado se puede preparar, según el tipo de desarrollo que sea, en un fichero jar de librería o en un fichero ear de aplicación.

Para su ejecución hay que situarse en el directorio **`/aplic/bbb/ant`**.

Los targets disponibles son los siguientes:

5.1 compileLib

Compila la librería que se encuentra en el directorio /aplic/bbb/src y deposita los ficheros compilados en el directorio /aplic/bbb/bin. Las librerías que son necesarias para realizar la compilación las obtiene de las siguientes rutas:

- El propio directorio de librerías de la librería que se desea compilar /aplic/bbb/src/bbbShLibClasses/lib.
- Las librerías que aporta XL-Nets.
- Las librerías que aporta Weblogic 11.

Ejemplo:

ant compileLib

5.2 createShLibJar

Crea el empaquetado jar de la librería (bbbShLibClasses.jar) dejándolo en el directorio /aplic/bbb/dist. Esta tarea incluye la ejecución previa de la tarea compileLib.

Ejemplo:

ant createShLibJar

5.3 compileApp

Compila la aplicación que se encuentra en el directorio /aplic/bbb/src y deposita los ficheros compilados en el directorio /aplic/bbb/bin. Las librerías que son necesarias para realizar la compilación las obtiene de las siguientes rutas:

- El propio directorio de librerías de la librería que se desea compilar /aplic/bbb/src/bbbEAR/APP-INF/lib.
- Las librerías que aporta XL-Nets.
- Las librerías que aporta Weblogic 11.

Ejemplo:

ant compileApp

5.4 createWar

Crea el empaquetado war de cada una de las aplicaciones web que contenga la estructura de directorios dejándolos en el directorio /aplic/bbb/dist. Las librerías que son necesarias para realizar la compilación las obtiene de las siguientes rutas:

- El propio directorio de librerías de la aplicación web que se desa compilar: /aplic/bbb/src/bbbEAR/wars/<nombreWar>/WEB-INF/lib
- El directorio de las librerías que se incluyen en la aplicación matriz: /aplic/bbb/src/bbbEAR/APP-INF/lib.
- Las librerías que aporta Weblogic 11.

Ejemplo:

ant createWar

5.5 createJar

Crea el empaquetado jar (bbbEarClasses.jar) de los fuentes java incluidos directamente en la aplicación matriz. Esta tarea se utiliza internamente por la creación de empaquetados EAR y no sirve para empaquetar librerías, para las cuales se utiliza la tarea createShLibJar. El empaquetado jar generado lo deposita en la ruta /aplic/bbb/dist.

Ejemplo:

ant createJar

5.6 createEjb

Crea el empaquetado de todos y cada uno de los proyectos EJB que estén incluidos en la ruta /aplic/bbb/bin/bbbEAR/ejbs/<nombre EJB>/ de la aplicación matriz. Los empaquetados de cada uno de los ejbs los deposita en la ruta /aplic/bbb/dist.

Ejemplo:

ant createEjb

5.7 createEar

Crea el empaquetado (bbbEAR.ear) de la aplicación matriz incluyendo las aplicaciones web, los EJBs y las clases que pueda incorporar la propia aplicación. Este empaquetado se deposita en la ruta /aplic/bbb/dist. Para ello ejecuta las tareas createWar, createJar y createEJB.

Ejemplo:

ant createEar

5.8 createWS

Modifica el fichero ear (bbbEAR.ear) generado en el directorio de distribución de la aplicación (p.e /aplic/bbb/dist), añadiéndole un webservice en un módulo war existente y declarado en el application.xml de la aplicación. Internamente se utiliza la tarea de weblogic jwsc.

Los requisitos para la utilización de esta tarea son:

- Disponer de un fichero ear válido en la ruta /aplic/bbb/dist
- Disponer de un fichero java con anotaciones de webservice en la ruta /aplic/bbb/src/bbbEAR/wars/<nombre WAR>/WEB-INF/classes
- Disponer de un descriptor web.xml en la ruta /aplic/bbb/src/bbbEAR/wars/<nombre WAR>/WEB-INF
- Generar un único webservice por contexto

Ejemplo:

ant createWS

-Dcontextpath=bbbJspStrutsWAR

-Djavafile=com/ejie/bbb/ws/Greet.java

-Dservice.name=Greeting

-Dservice.portname=Greetingport

5.9 sync

Sincroniza los ficheros de la carpeta /aplic/bbb/src en la carpeta que contiene la aplicación en modo exploded /aplic/bbb/dist/exploded. Sólo serán copiados y compilados los ficheros modificados desde la última compilación.

Ejemplo:

ant sync

5.10 all

Esta tarea compila y empaqueta una aplicación. Para ello ejecuta las tareas compileApp y createEar.

Ejemplo:

ant all

6. Targets de creación y eliminación de recursos

Ver WLS11_NormasAlbergue.doc

Estas tareas permiten crear y eliminar recursos JDBC y recursos JMS que sean necesarios para el correcto funcionamiento de las aplicaciones a desplegar. Estos target invocan a scripts WLST que llevarán a cabo estas acciones.

Para ello, estos recursos deben tener correctamente definidos los datos necesarios para su configuración en el fichero bbbResources.properties que se encuentra en la ruta /<entorno>/config/<dominio>/bbb/.

Para su ejecución hay que situarse en el directorio **/aplic/bbb/ant**.

Los targets disponibles son los siguientes:

6.1 createDataSource

Crea un DataSource dentro de un dominio de Weblogic 11.

La información del datasource que debe encontrarse en el fichero bbbResources.properties es la siguiente:

- jdbc_DataSourceName: el nombre del data source.
- jdbc_DataSourceJNDIName: el nombre JNDI del data source.
- jdbc_DataSourceDriver: el driver que se va a utilizar para llevar a cabo las operaciones que permiten una única fase para el manejo de la transaccionalidad.
- jdbc_DataSourceDriverXA: el driver que se va a utilizar para llevar a cabo las operaciones que requieren más de una fase para el manejo de la transaccionabilidad.
- jdbc_Tnsnames: el nombre que tiene la instancia de base de datos en el servidor.
- jdbc_DataBaseUsername: el nombre del usuario de base de datos para realizar el test una vez creado el data source.
- jdbc_DataBasePassword: la contraseña que tiene el usuario de base de datos definido en el punto anterior.

- jdbc_TestTableName: la tabla que va a servir para realizar la prueba.
- jdbc_GlobalTransactionsProtocol: OnePhaseCommit / TwoPhaseCommit / EmulateTwoPhaseCommit.

Ejemplo:

ant createDataSource -Ddominio=wl11_intra_apps_dpto

6.2 deleteDataSource

Borra un DataSource dentro de un dominio de Weblogic 11.

La información del datasource, igual que en la creación, debe encontrarse en el fichero bbbResources.properties y tiene que contener los mismos datos que en la creación, aunque no todos serán necesarios.

Ejemplo:

ant deleteDataSource -Ddominio=wl11_intra_apps_dpto

6.3 createJMS

Crea un servidor JMS con una Connection Factory y una Topic, y los asocia a un servidor manejado.

La información del datasource que debe encontrarse en el fichero bbbResources.properties es la siguiente:

- jdbc_DataSourceDriver_jms: el driver que se va a utilizar para llevar a cabo las operaciones con colas JMS.
- jdbc_Tnsnames_jms: el nombre que tiene la instancia de base de datos en el servidor.
- jdbc_DataBaseUsername: el nombre del usuario de base de datos para realizar el test una vez creado el data source.
- jdbc_DataBasePassword: la contraseña que tiene el usuario de base de datos definido en el punto anterior.
- jdbc_TestTableName: la tabla que va a servir para realizar la prueba.
- jdbc_GlobalTransactionsProtocol: OnePhaseCommit / TwoPhaseCommit / EmulateTwoPhaseCommit.
- num_queues: el número de colas JMS que se crean.
- queue_[1..n]: n depende del número de colas a crear. Estos campos describen cada cola.
- num_topics: el número de topics JMS que se crean.
- Topic_[1..n]: n depende del número de topics. Estos campos describen cada topic.

Ejemplo:

ant createJMS -Ddominio=wl11_intra_apps_dpto

6.4 deleteJMS

Elimina el servidor JMS del dominio de Weblogic 11.

La información del servidor JMS y de su topic, igual que en la creación, debe encontrarse en el fichero bbbResources.properties y tiene que contener los mismos datos que en la creación, aunque no todos serán necesarios.

ant deleteJMS -Ddominio=wl11_intra_apps_dpto

7. Targets de despliegue

Ver WLS11_NormasAlbergue.doc

Estas tareas permiten desplegar, redesplegar o eliminar una aplicación del servidor y dominio de destino elegidos.

Para ello es necesario que el fichero bbbConfig.properties que se encuentra en la ruta /<entorno>/config/<dominio>/bbb/ esté configurado correctamente con la información necesaria para contactar con el servidor de Weblogic 11.

La información contenida en el fichero bbbConfig.properties es la siguiente:

- WLS_URL: la url del servidor en el que se van a realizar las operaciones.
- WLS_TARGETS: target del servidor en el que se va a realizar la operación.
- wl_managed_node_1: el nodo manejado 1 del cluster.
- wl_managed_node_2: el nodo manejado 2 del cluster.
- wl_managed_node_shortname_1: el nombre corto del nodo manejado 1 del cluster.
- wl_managed_node_shortname_2: el nombre corto del nodo manejado 2 del cluster.

Es necesario trabajar con nombres cortos para los nodos en algunas configuraciones de weblogic, como en los objetos JMS. Esto es debido a que la longitud de los nombres normales supera el tamaño máximo permitido.

Para su ejecución hay que situarse en el directorio **/aplic/bbb/ant**.

Los targets disponibles son los siguientes:

7.1 deploy

Despliega la aplicación completa generada en la ruta /aplic/bbb/dist. El modo de despliegue es stage.

Ejemplo:

ant deploy -Ddominio=wl11_intra_apps_dpto

7.2 deployExploded

Despliega el contenido del directorio /aplic/bbb/dist/exploded en el servidor.

Ejemplo:

ant deployExploded -Ddominio=wl11_intra_apps_dpto

7.3 undeploy

Elimina el despliegue de la aplicación en el servidor de aplicaciones.

Ejemplo:

ant undeploy -Ddominio=wl11_intra_apps_dpto

7.4 redeploy

Vuelve a realizar el despliegue de una aplicación ya desplegada anteriormente.

Ejemplo:

ant redeploy -Ddominio=wl11_intra_apps_dpto

8. Targets de creación de un servicio a partir de una clase Java

Ver Normativa_desarrollo_ERPI_Servicios v1 1.doc

A partir de una clase con determinadas anotaciones provistas por JAX-WS, la normativa de desarrollo de servicios proporciona un script que emplea tareas ant propias de Weblogic para generar:

- El descriptor WSDL del servicio.
- Los esquemas XSD.
- Ficheros estándar de configuración de JAX-WS (web.xml, webservices.xml).
- Ficheros propietarios de configuración de weblogic (weblogic-webservices.xml, weblogic-webservices-policy.xml).
- Eventualmente, clases adicionales Java (sobre todo si se ha configurado parameterStyle=WRAPPED).

El script de ant y las anotaciones recomendadas deben consultarse en el documento de Normativa_desarrollo_ERPI_Servicios v1 1.doc.

Dicho script se debe situar en la ruta "/aplic/bbb/bbbEjemploWAR/ant", y proporciona los siguientes target:

8.1 clean-build-service

Borra el directorio temporal que utiliza el script.

Ejemplo:

ant clean-build-service

8.2 build-service

Genera las clases java, WSDL, XSD, web.xml y descriptores propietarios de weblogic.

Ejemplo:

ant build-service

8.3 copy-generated-sources

Copia todos los recursos generados mediante la tarea build-service a las carpetas de fuentes adecuadas.

Ejemplo:

ant copy-generated-sources

8.4 default

Realiza la generación y la copia de recursos, utilizando los targets descritos anteriormente.

ant default

9. Targets de creación de un servicio a partir de un descriptor WSDL

Ver Normativa_desarrollo_ERPI_Servicios v1 1.doc

A partir de un descriptor WSDL creado adecuadamente, la normativa de desarrollo de servicios proporciona un script que permite generar las clases necesarias para implementar y desplegar un

servicio. Este script utiliza tareas de ant propias de Weblogic para generar las clases JAX-WS y recursos necesarios para crear el servicio.

El script se debe debe consultar en el documento Normativa_desarrollo_ERPI_Servicios_v1_1.doc. Según este documento, para un código de aplicación bbb, si el WAR en el que se desea desarrollar el servicio se denomina bbbEjemploWsdlWAR, el script se situaría en la carpeta "/aplic/bbb/bbbEjemploWsdlWAR/ant" y proporcionaría los siguientes target:

9.1 clean-service

Borra el directorio temporal que utiliza el script.

Ejemplo:

ant clean-service

9.2 generate-service

Crea la clase de implementación del servicio, la interfaz de la clase de implementación y las clases JAX-B necesarias para mapear los mensajes.

Ejemplo:

ant generate-service

9.3 copy-generated-sources

Copia todos los recursos generados mediante la tarea generate-service a las carpetas de fuentes adecuadas.

Ejemplo:

ant copy-generated-sources

9.4 build-service

Genera los ficheros de configuración estándar y propietarios de WebLogic para desplegar el servicio ("web.xml", "webservices.xml", "weblogic-webservices.xml", "weblogic-webservices-policy.xml") en el directorio temporal.

Ejemplo:

ant build-service

9.5 copy-build-service

Copia los ficheros de configuración a la carpeta "…/WebContent/WEB-INF" del proyecto. El fichero "web.xml" es copiado con un nombre distinto, para no sobrescribir la configuración ya establecida. Se deberán copiar las secciones de "servlet" y "servlet-mapping" del web.xml generado en el fichero "web.xml" de la aplicación para desplegar realmente el servicio web.

Ejemplo:

ant copy-build-service

9.6 default

Realiza la generación y la copia de recursos, utilizando los targets descritos anteriormente.

ant default

10. Targets de calidad (Dashboard-EJIE)

A partir de unas pruebas de calidad correctamente generadas, la oficina tecnica de calidad de Ejie define las siguientes tareas. Una primera tarea que permite compilar estas pruebas, y una serie de tareas ant que permiten lanzar y comprobar la calidad de la aplicación.

10.1.1. compileOTC

Tarea ant que compila el codigo fuente de calidad. Para ello deberan existir las clases java con su correspondiente estructura.

ant compileOTC

10.1.2. test-static

Ejecución del test de Findbugs.

ant test-static

10.1.3. test-unit

Ejecución de tests de junit para pruebas unitarias.

ant test-unit

10.1.4. test-integration

Ejecución de tests de junit para pruebas de integración.

ant test-integration

10.1.5. test-soapui-integration

Ejecución de test de webservices de integración.

ant test-soapui-integration

-DhostSoapUI=www.integracion.jakina.ejiedes.net

-Dotc.proyecto.nombre=w84b

-Dotc.proyecto.version=1.1

-Dotc.proyecto.entorno=Desarrollo

-Dotc.proyecto.perfil=NAC alto

10.1.6. test-system

Ejecución de tests de junit para pruebas de sistema.

ant test-system

10.1.7. test-soapui-system

Ejecución de test de webservices de sistema.

ant test-soapui-system

-DhostSoapUI=www.integracion.jakina.ejiedes.net

-Dotc.proyecto.nombre=w84b

-Dotc.proyecto.version=1.1

-Dotc.proyecto.entorno=Desarrollo

-Dotc.proyecto.perfil=NAC alto