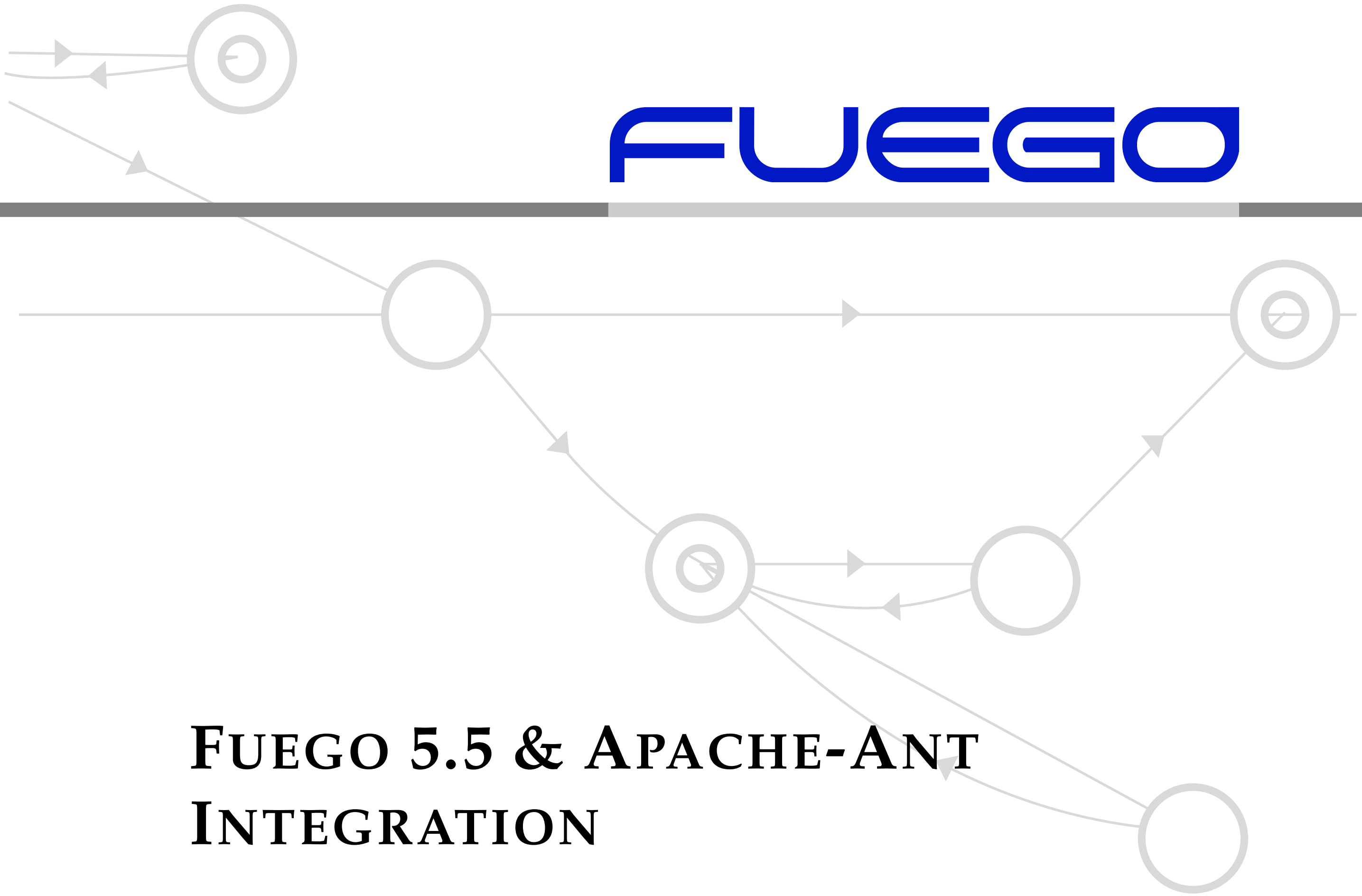

FUEGO 5.5 & APACHE-ANT INTEGRATION

May 23, 2005

Abstract

This document describes the integration between Fuego and Apache's Ant build tool.

Fuego provides an Ant library to automate some administrative tasks using Ant scripts. This includes the deployment of processes, managing Engine configurations, importing/exporting organizational data.

It is assumed that the reader is familiar with XML and the Fuego system. Basic knowledge of how Ant works is also required.

Contents

1 Introduction

Apache Ant[1] is a platform-independent (Java-based) build tool. It is useful for automating repetitive tasks and thus is well suited for implementing standardized build and deployment processes.

Ant scripts are called *build* files and are written in XML. Each build file defines one *project* with at least one *target*. A target contains a set of *tasks*, which are the actions that will be executed for that target.

The power of Ant lies on the fact that its XML syntax is designed to be extended. Ant provides a good set of standard tasks to achieve many common actions (like copying files and compiling source code), but new ones can be added.

Fuego provides a set of *tasks* that add Fuego-specific functionality to Ant scripts, including:

- Publishing and Deploying processes (see Task fuego:publish on page 24)
- Starting, Stoping and general management of Fuego Engines (see Task fuego:engine on page on page 16)
- Importing and exporting of configuration resources (see Task fuego:resources on page 28)
- Importing and exporting of Organizational data –participants, roles, org. units– (see Task fuego:directoryadmin on page 12)
- Creation and general management of Fuego Directories (see Task fuego:directoryschema on page 15)

This document is not a tutorial on Ant itself, but on the additional tasks provided by Fuego.

For more documentation, tutorials and general reference about Ant, please refer to its official website. It provides a manual and pointers to several online articles and published books about Ant.

2 Requirements

The Fuego Ant tasks were developed for Ant 1.6.1 or newer, and it relies on the *namespace* and *antlib* features included in Ant version 1.6.

[1] http://ant.apache.org

2.1 Fuego Ant libraries

In order to make the Fuego tasks available to Ant, the Fuego antlib jar files need to be specified using the `-lib` option. For example:

```
On Unix:
ant -lib /usr/fuego5.5/enterprise/lib/fuegoenterprise-antlib.jar

On Windows:
ant -lib c:\fuego5.5\enterprise\lib\fuegoenterprise-antlib.jar
```

To avoid the need of using the `-lib` argument every time, it can be specified once using the `ANT_ARGS` environment variable:

```
On Unix:
ANT_ARGS="-lib /usr/fuego5.5/enterprise/lib/fuegoenterprise-antlib.jar"
export ANT_ARGS

On Windows:
set ANT_ARGS=-lib c:\fuego5.5\enterprise\lib\fuegoenterprise-antlib.jar
```

When working with the J2EE edition of Fuego, there is a second antlib that Fuego provides and must also be passed with the `-lib` option:

```
On Unix:
FUEGO_HOME=/usr/fuego5.5/j2ee
ANT_ARGS="-lib $FUEGO_HOME/lib/fuegoenterprise-antlib.jar:$FUEGO_HOME/j2ee/
    common/tools/ant/fuegoj2ee-antlib.jar"
export ANT_ARGS

On Windows:
set FUEGO_HOME=c:\fuego5.5\j2ee
set ANT_ARGS=-lib %FUEGO_HOME%\lib\fuegoenterprise-antlib.jar;%FUEGO_HOME%\
    j2ee\common\tools\ant\fuegoj2ee-antlib.jar
```

2.2 Fuego Directory drivers

Some of the Fuego Ant tasks need connectivity to the Fuego Directory.

If your Fuego Directory is stored on a relational database, Ant needs to load the required JDBC driver. A quick solution is to add the driver's .jar to the `-lib` option. Example:

```
On Unix:
ANT_ARGS="-lib $FUEGO_HOME/lib/fuegoenterprise-antlib.jar:$FUEGO_HOME/ext/
    ojdbc14.jar"
export ANT_ARGS

On Windows:
set ANT_ARGS=-lib %FUEGO_HOME%\lib\fuegoenterprise-antlib.jar;%FUEGO_HOME%$\
    backslash$ext$\backslash$ojdbc14.jar
```

2.3 Fuego Namespace declaration

Ant scripts need to include the Fuego antlibs in its project definition in order to use Fuego tasks:

```
<project name="FuegoExample"
         xmlns:fuego="antlib:fuego.tools.ant.enterprise"
         xmlns:fuego.j2ee="antlib:fuego.tools.ant.j2ee">
 ...
</project>
```

The previous example defines the `fuego` namespace for accessing the standard Fuego library of tasks and `fuego.j2ee` for those tasks specific to the J2EE edition of Fuego.

Every reference to a standard Fuego task will be prefixed by "`fuego:`", and the J2EE-specific ones will be referenced with the "`fuego.j2ee:`" prefix. Like in the following snippet that uses the `publish` and `buildear` tasks:

```
<project name="FuegoExample" xmlns:fuego="antlib:fuego.tools.ant.enterprise">
 ...
  <target name="publish">
    ...
    <!-- Publish a process -->
    <fuego:publish fpr="myproject.fpr">
      ...
    </fuego:publish>

    <fuego.j2ee:buildear ...
      ...

    </fuego.j2ee:buildear>
  </target>
 ...
</project>
```

2.4 Setting `fuego.basedir`

Finally, the last requirement for Fuego Ant tasks to work correctly is to define the `fuego.basedir` property inside the build script. This property must point to a valid Fuego Enterprise installation:

```
<project name="FuegoExample" xmlns:fuego="antlib:fuego.tools.ant.enterprise">

  <property name="fuego.basedir"
           value="/usr/fuego5.5/enterprise"/>
 ...
</project>
```

3 Using Fuego tasks

The following example represents a small but complete Ant script that uses Fuego tasks.

The example script publishes and deploys a Fuego process making use of:

- Task fuego:publish (see page 24)
- Task fuego:session (see page 29)
- Type fuego:passport (see page 23)

Fuego:publish is the task that allows for publishing and deploying processes. As any other task that needs access to a Fuego directory, it must be enclosed by a fuego:session task.

A fuego:passport basically defines the authentication information needed to access a particular Fuego directory. The fuego:session task accepts a passport reference to establish a session to the directory.

build.xml

```
<!-- This script publishes and deploys a Fuego Process. -->
<project name="FuegoExample" xmlns:fuego="antlib:fuego.tools.ant.enterprise">

  <!-- Include properties -->
  <property file="build.properties"/>

  <!-- Define a fuego passport -->
  <fuego:passport id="fuego.passport"
         directoryid="${fuego.directoryid}"
         participant="${fuego.participant}"
            password="${fuego.password}" />

  <target name="publish" description="Publish and deploy processes">

  <!-- Open a session to the Fuego directory -->
    <fuego:session
          passportref="fuego.passport"
              verbose="true"
          haltonerror="true" >

      <!-- Publish processes -->
      <fuego:publish fpr="${fuego.project}"
                  deploy="true"
                  engine="${fuego.engine}">

        <fuego:rolemap abstract="Employee" real="Role1"/>
        <fuego:rolemap abstract="Idea Evaluator" real="Role1"/>
      </fuego:publish>
    </fuego:session>
  </target>

</project>
```

The example includes properties from another file: `build.properties`. Keeping the values that are likely to change in a separate properties file is a good practice to follow, since it allows for easy parameterization of the script.

This is a `build.properties` file suitable for the above example:

build.properties

```
# Fuego Enterprise installation directory
fuego.basedir=/usr/local/fuego5.5/j2ee

# Fuego directory ID, user and password
fuego.directoryid=LocalOracle
fuego.participant=root
fuego.password=password

# Name of Fuego Engine to deploy process
fuego.engine=Standalone

# Project to deploy
fuego.project=/usr/local/fuego5.5/studio/samples/HelloWorld.fpr
```

The following sections describe each of the available Fuego tasks in more detail, including all the attributes they accept and examples of use.

4 fuego Library

4.1 Task fuego:database

This task allows the management of a database schema.

Using an schema definition in xml format, this task can create, drop or clean a database schema.

This task connects to the database using configuration of type SQL that must previously exist.

Example:

```
<fuego:database action="create"
         connectorName="oracleConnector"
                  file="/tmp/configurations.xml"
          userProperty="user"
      passwordProperty="password"/>
```

Parameters:

- **`haltonerror`** – boolean *Not required.* Defaults to true. Stop the Ant build process if an error occurs.
- **`userproperty`** – String *Required.* Sets the configuration's property key for the user name.
- **`file`** – File *Required.* Sets the location of an xml file containing the configuration's definitions.
- **`passwordproperty`** – String *Required.* Sets the configuration's property key for the password.
- **`verbose`** – boolean
- **`action`** – String [”create”, ”drop”, ”clean”]*Required.* Defines the operation to execute:
  - **create**: create a new database schema.
  - **drop**: drop an existing database schema.
  - **clean**: create a new database schema if it does not exist, or remove all the existing rows in all tables otherwise.
- **`schemafile`** – File *Required.* Sets the file location of the xml schema definition.
- **`user`** – String *Not required.* Defaults to null Sets the user to connect to the database.

- **`connectorname`** – String *Required.* Defines the name of the SQL database configuration.
- **`fuegobasedir`** – File
- **`loaderref`** – Reference
- **`password`** – String *Not required.* Defaults to null Sets the user's password to connect to the database.

4.2 Task fuego:directory

This task allows for exporting and importing the organizational information (roles, participants, OUs, etc).

Particular types of objects and their properties categories can be selected using the `include` nested elements. For each element type the categories selected to export can be selected using the `property` nested elements.

When the objects are imported to the directory, all categories contained in the file will be also imported and cannot be excluded.

When importing participants, a password file can be generated if desired. By default no password file is generated.

Export example:

```
...
 <!-- Open a session to the Fuego directory -->
   <fuego:session passportref="fuego.passport">

     <!-- Export participants and roles -->
     <fuego:directory file="${basedir}/export.xml"
                    action="export"/>

         <!-- include participants and export only the category "prefs"-->
         <include type="participants">
            <property name="prefs"/>
         </include>
         <!-- include roles and do not export categories -->
         <include type="roles"/>

     </fuego:directory>
 </fuego:session>
 ...
```

Import example:

```
...
 <!-- Open a session to the Fuego directory -->
   <fuego:session passportref="fuego.passport">
```

```
    <!-- Import participants (generating password file) and roles -->
    <fuego:directory file="${basedir}/export.xml"
                   action="import"
         generatePassFile="true"/>

        <!-- include participants -->
        <include type="participants"/>

        <!-- include roles-->
        <include type="roles"/>

    </fuego:directory>
</fuego:session>
...
```

Parameters:

- **`file`** – File *Required.*

  File used to export data if an export action is selected, or file containing the organization data to import if an import action is selected.

- **`generatepassfile`** – boolean *Not required.* Defaults to false Whether to generate a password file after importing participants.

- **`action`** – String [”export”, ”import”]*Required.* Whether to execute an export or an import.

  Possible values are [export,import]

- **`charset`** – String *Not required.* Defaults to default VM charset The charset name to use when the export file is written.

- **`haltonerror`** – boolean *Not required.* Defaults to false Whether to stop the ant build process if an error occurs during the execution of this task.

Parameters accepted as nested elements:

`<include>` – *(See Type fuego:directory.IncludeType on the next page)*

Parameters:

- **`type`** – String [”views”, ”referrals”, ”groups”, ”ous”, ”holidayrules”, ”presentations”, ”roles”, ”businessparameters”, ”variables”, ”calendarrules”, ”participants”]*Required.*

`<category>`

Parameters:

- **name** – String *Required.*

4.3 Type fuego:directory.IncludeType

Parameters:

- **type** – String [”views”, ”referrals”, ”groups”, ”ous”, ”holidayrules”, ”presentations”, ”roles”, ”businessparameters”, ”variables”, ”calendarrules”, ”participants”]*Required.* Type of object to include in the export/import.

Parameters accepted as nested elements:

<category>

Parameters:

- **name** – String *Required.*

4.4 Task fuego:directoryadmin

Manage the organization information (roles, participants, ous, etc).

Example:

```
<fuego:directoryadmin importFile="/tmp/fdiexport.xml">

    <deleteou name="ou1"/>
    <deleteou name="ou2"/>

    <createou name="ou3" description="ou1 description"/>
    <createou name="ou4"/>

    <deleterole name="role1"/>
    <deleterole name="role2"/>
    <deleterole name="roleParam1"/>

    <createrole name="role3" description="role1 description"/>
    <createrole name="role4" parametric="false"/>

    <createrole name="roleParam2"
         description="parametric role1 description"
          parametric="true">

          <param value="param1"/>
```

```
            <param value="param2"/>
        </createrole>

    </fuego:directoryadmin>
```

Parameters:

- **`importfile`** – File *Required.* File used to import the organization data.
- **`haltonerror`** – boolean *Not required.* Defaults to true. Stop the Ant build process if an error occurs.

Parameters accepted as nested elements:

`<deleterole>` – *(See Type fuego:directoryadmin.Role on the following page)* A role definition.

Parameters:

- **`name`** – String *Not required.*
- **`refid`** – Reference
- **`parametric`** – boolean *Not required.* Defaults to false.

`<imports>` (Of type FileSet)

Parameters:

- **`refid`** – Reference
- **`file`** – File
- **`dir`** – File
- **`defaultexcludes`** – boolean
- **`followsymlinks`** – boolean
- **`casesensitive`** – boolean
- **`excludes`** – String
- **`includesfile`** – File
- **`excludesfile`** – File
- **`includes`** – String

`<deleteou>`

Parameters:

- **`name`** – String *Required.*
- **`refid`** – Reference

`<createrole>` – *(See Type fuego:directoryadmin.Role on the current page)* A role definition.

Parameters:

- **`name`** – String *Not required.*
- **`refid`** – Reference
- **`parametric`** – boolean *Not required.* Defaults to false.

`<createou>`

Parameters:

- **`name`** – String *Required.*
- **`refid`** – Reference

4.5 Type fuego:directoryadmin.Role

A role definition.

Parameters:

- **`name`** – String *Not required.* Sets the name of the Organizational Role.
- **`refid`** – Reference
- **`parametric`** – boolean *Not required.* Defaults to false. Whether the role is parametric or not.

Parameters accepted as nested elements:

`<param>`

Parameters:

- **`refid`** – Reference
- **`value`** – String *Required.*

4.6 Task fuego:directoryschema

Manages the fuego directory schema.

Using a set of properties containing an schema definition, this task can create, drop, repair, recreate and clean a database schema.

Example:

```
<fuego:directoryschema action="create"
                             file="/tmp/oracle.schema"
             runtimeProperties="/tmp/conf/directory.properties"/>
```

Parameters:

- **`haltonerror`** – boolean *Not required.* Defaults to true. Stop the Ant build process if an error occurs.
- **`file`** – File *Required.* File name containing the schema properties.

  The properties contains the information to create, drop, clean and repair a Directory schema.
- **`action`** – String [”create”, ”drop”, ”repair”, ”recreate”, ”clean”]*Required.* Defines the operation to execute:
  – **create**: create a new Fuego Directory schema.
  – **drop**: drop an existing schema.
  – **recreate**: drop an existing schema and create it.
  – **repair**: repair an existing schema.
  – **clean**: create a new Fuego Directory schema if it does not exist, or remove all the existing objects otherwise.
- **`verbose`** – boolean
- **`fuegobasedir`** – File
- **`loaderref`** – Reference
- **`runtimeproperties`** – File *Not required.* Defaults to null. Location of the file where the runtime properties are stored.

  This file is used to extract the runtime properties to establish a session to the Directory.

4.7 Task fuego:engine

This is task provides the ability to configure and manage Fuego Engines.

It can start/stop/create/drop and do general management of Engines.

Example:

```
<target name="start-engine" description="Start a Fuego Engine">

  <!-- Open a session to the Fuego directory -->
  <fuego:session
        passportref="fuego.passport"
            verbose="false"
        haltonerror="true" >

     <!-- Make sure the keystore is created (this is actually
             required only once per installation) -->
     <fuego:engine action="enable-location"
                   engine="${fuego.engine}">
         <fuego:admin participant="${fuego.participant}"
                         password="${fuego.password}"/>
     </fuego:engine>

     <!-- Start the Engine -->
     <fuego:engine action="start" engine="${fuego.engine}"/>

  </fuego:session>

</target>
```

Parameters:

- **`haltonerror`** – boolean *Not required.* Default is true Whether to stop the ant build process if an error occurs during the execution of this task.
- **`arguments`** – String *Not required.* If not set, the value specified via Web console will be used Optional additional arguments to the Fuego Engine. When specified, this arguments will override those on the Engine configuration (webconsole).
- **`connector`** – String *Required.* when creating a new Engine (that is, `action="-create"`) Name of the database resource configuration. This is the configuration to use for the Engine's back-end database.
- **`engineid`** – String *Required.* ID of the Fuego Engine to work on.
- **`action`** – String [”create”, ”delete”, ”create-database”, ”recreate-database”, ”drop-database”, ”import”, ”start”, ”stop”, ”status”, ”enable-location”, ”clean-database”, ”modify-properties”]*Required.* Specifies what action to execute:
  - **create:** Create a new Engine configuration.

- **delete:** Delete an existing Engine configuration.
- **create-database:** Create an Engine's back-end database structure.
- **recreatedatabase:** Drop and Create an Engine's back-end database structure.
- **drop-database:** Drop an Engine's back-end database structure.
- **import:**
- **start:** Start an Engine.
- **stop:** Stop a running Engine.
- **status:** Get status information about an Engine.
- **enable-location:** Create a local "keystore" file that is required for other engine actions. This is needed only once per-Fuego installation.
- **clean-database:** Clean the back-end database contents.

- **`runtimeconnector`** – String *Not required.* If omitted, the value of `connector` is used.

- **`type`** – String *Not required.* If not set, the default value is the type installed, or if there are more than one engine type installed, will fail. Specifies the type of engine to create. Is only required when **action** is "create".

- **`importprops`** – boolean

- **`homedir`** – File *Required.* when creating a new Engine configuration. Home directory for the new Engine. The new Engine's log directory will be `${homedir}/log`

- **`vmarguments`** – String *Not required.* If not set, the value specified via Web console will be used. Optional additional arguments to pass to the Java Virtual Machine. When specified, this arguments will override those on the Engine configuration (webconsole).

Parameters accepted as nested elements:

`<engprops>` – *(See Type fuego:engine.EngineProperties on the following page)*

This element is a placeholder for a set of Property elements.

Parameters:

`<admin>` Represents the credentials for an "Administrator" user.

If it's used to represent a Fuego Directory administrator, then `participant` should be used for the participant id. For connections to other systems (like DBMSs) the `user` attribute will be used.

This element is required when `action` equals `enable-location`, `create-database`, `clean-database` or `drop-database`.

Parameters:

- **`participant`** – String *Required.* When used to connect to a Fuego Directory
- **`user`** – String *Required.* When used for connections not to the Fuego Directory
- **`password`** – String *Required.*

4.8 Type fuego:engine.EngineProperties

This element is a placeholder for a set of Property elements.

Parameters:

Parameters accepted as nested elements:

`<property>`

This is the list of possible properties, grouped as shown in Web Console: the name, the description, the type, the default value, and restrictions:

- Log:
  - log.engineLogSeverity - Messages logged from Engine - String - Default: Warning - Values: Fatal,Severe,Warning,Info,Debug
  - log.methodLogSeverity - Messages logged from BP-Methods - String - Default: Info - Values: Fatal,Severe,Warning,Info,Debug
  - log.mailLogSeverity - Messages sent by mail - String - Default: None - Values: None,Warning,Severe
  - log.detailLevel - Log detail level - int - Default: 1 - Values between 1 and 10
  - log.maxLogSize - Maximum size of log file - int = Default: 2000 Kb - Values between 1000 and 1000000000
  - log.maxLogFiles - Maximum number of log files - int - Default: 5 - Values between 1 and 100
- Startup:
  - startup.autoStartable - Start automatically during Web Console initialization - boolean - Default: false
  - startup.arguments - Additional arguments used in startup - String
  - startup.javaArguments - Additional java arguments used in startup - String
- Memory:
  - memory.maxJvmHeapSize - Maximum JVM heap size - int - Default: 256 Mb - Values between 256 and 4096
  - memory.maxInstanceSize - Maximum instance size - int - Default: 16 Kb - Values between

  - memory.instancesCacheSize - Instances cache - int - Default: 5000
- Execution Threads:
  - execution.interactiveExecutionThreadsPoolSize - Maximum number of execution threads used for interactive executions - int - Default: 50 - Values between 10 and 200
  - execution.automaticExecutionThreadsPoolSize - Maximum number of execution threads used for automatic tasks - int - Default: 5 - Values between 1 and 200
  - execution.automaticExecutionThreadsPriority - Priority of Automatic Execution Threads - int - Default: 5 - Values between 1 and 10
  - execution.automaticItemsQueueSize - Automatic items queue size - int - Default: 1000 - Values between 100 and 100000
  - execution.retryTimes - Retry times - int - Default: 5 - Values between 0 and 100
  - execution.retryInterval - Retry interval - int - Default: 1800 - Values between 15 and 100000
- Timeouts:
  - timeouts.maxMethodTimeout - Maximum BP-Methods timeout - int - Default: 30 seconds
  - timeouts.itemExecutionTimeout - Interactive component timeout - int - Default: 720 minutes
- Debugger:
  - execution.traceComponents - Trace components - boolean - Default: false
- Disposer:
  - disposer.disposerLatency - Disposer latency - int - Default: 2 - Values between 0 and 100
  - disposer.instanceCaducity - Instance caducity - int - Default: 15 - Values between 1 and 100
  - disposer.disposeParticipants - Dispose Participants - boolean - Default: true
  - disposer.participantCaducity - Participant caducity - int - Default: 30 days - Values between 15 and 1000
  - disposer.disposerStartingTime - Disposer starting time - Time
  - disposer.archivingEnabled - Enable Archiving - boolean - Default: false
  - disposer.archivingConfiguration - Archiving Configuration - String
- IPC:
  - ipc.serviceEnabled - Enable IPC service - boolean - Default: false
  - ipc.servicePort - IPC service Port - int - Default: 54350
  - ipc.maxIncomingConnections - Maximum incoming connections - int - Default 5
- Socket Factory:
  - execution.enableSocketFactory - Enable socket factory - boolean - Default: false
- SNMP:
  - snmp.serviceEnabled - Enable SNMP service - boolean - Default: false
  - snmp.agentPort - SNMP agent port - int - Default: 161

  - snmp.managerPort - SNMP manager host - String - Default: $LOCALHOST$
  - snmp.managerHost - SNMP manager port - int - Default: 162
- Networking:
  - networking.mailServerName - Mail server name - String - Default: smtp
  - networking.administratorMail - Administrator mail - String - Default: ftadmin
  - networking.portalUrl - Web Work Portal URL - String - Default: http://$LOCALHOST$:9595/portal
  - networking.webServicesUrl - Web Services URL - String - Default: http://$LOCALHOST$:8585/fws/servlet/Deploy
- DB:
  - creationConfiguration - Creation configuration - String
  - runtimeConfiguration - Runtime configuration - String
- Directory:
  - directoryPollingInterval - Directory polling interval - int - Default: 1 - Values between 1 and 60
- Store Events:
  - storeEvents - Store Events - String - Default: DEPENDS - Values: DEPENDS,NEVER,ALWAYS
- PAPI:
  - papi.instanceRetrievalSize - Instance retrieval size - int - Default: 1000 - Values between 100 and 50000
  - papi.notifyThreadPriority - Notify thread priority - int - Default: 1 - Values between 1 and 10
  - papi.latencyBetweenNotifications - Latency between notifications - int - Default: 15 seconds - Values between 1 and 300

Parameters:

- **`name`** – String
- **`type`** – String
- **`value`** – String

4.9 Task fuego:instancesmanager

This task allows for exporting and importing process instances of a given Fuego Server.

All instances of all Active processes are exported. Deprecated instances are neither exported nor imported.

This task needs access to a Fuego Directory, so it must be contained in a `fuego:session` element.

Because process metadata is retrieved from the Directory, the processes must be deployed for this task to work.

When importing, a properties file may be specified for mapping instance variable names. This allows for instance variables to be imported with a name different from the original.

Example 1 (export instances in one file per Server table in the database):

```
...
<!-- Open a session to the Fuego directory -->
<fuego:session passportref="fuego.passport">

  <fuego:instancesmanager file="/tmp/export/"
                        action="export"
                      engineid="${export.engineid}"
                       oneFile="false"/>
</fuego:session>
...
```

Example 2 (export instances in one file):

```
...
<fuego:session passportref="fuego.passport">

  <fuego:instancesmanager file="/tmp/export/instances.xml"
                        action="export"
                      engineid="${export.engineid}"/>
</fuego:session>
...
```

Example 3 (import instances exported in example 1):

```
...
<fuego:session passportref="fuego.passport">

  <fuego:instancesmanager file="/tmp/export/"
                        action="import"
                      engineid="${import.engineid}"
                       oneFile="false"/>
</fuego:session>
...
```

Example 4 (import instances exported in example 2):

```
...
<fuego:session passportref="fuego.passport">

  <fuego:instancesmanager file="/tmp/export/instances.xml"
                        action="import"
                      engineid="${import.engineid}"/>
</fuego:session>
...
```

Parameters:

- **`fieldsmappingfile`** – File *Not required.* Defaults to no mapping file.

  Sets a file location to a properties file containing a mapping between old instance variable names and new ones.

- **`file`** – File *Required.*

  Sets the file name to export instances, if `oneFile` is selected, or a directory name, otherwise.

- **`engineid`** – String *Not required.* Sets the name of the Server.

- **`action`** – String [”import”, ”export”]*Required.* Whether to execute an export or an import.

  Possible values are [export,import]

- **`serverid`** – String *Required.* Sets the name of the Server.

- **`onefile`** – boolean *Not required.* Defaults to true.

  Whether to export/import instances to/from one file, or one file per Server table in database.

- **`counterstep`** – int *Not required.* Defaults to 100. Sets the number of steps in which the progress is shown.

4.10 Task fuego:parttrust

This task allows the management of participant trusts for a Directory.

Trusts can be added or removed for a participant, or for all participants in a directory. The `skip-auth` property can also be set.

Example 1 (adds a trust to a participant with `skip-auth`):

```
<fuego:parttrust action="add"
            participant="paul"
               skipAuth="true"
                  trust="dbFDIUser"/>
```

Example 2 (adds a trust to a participant without `skip-auth`):

```
<fuego:parttrust action="add"
            participant="paul"
               skipAuth="false"
                  trust="dbFDIUser"/>
```

Example 3 (adds a trust to all participants with `skip-auth`):

```
<fuego:parttrust action="add"
               skipAuth="true"
                  trust="dbFDIUser"/>
```

Example 4 (adds a trust to a participant without `skip-auth` for all database users):

```
<fuego:parttrust action="add"
            participant="paul"
               skipAuth="false"/>
```

Example 5 (removes a trust to a participant):

```
<fuego:parttrust action="add"
            participant="paul"
                  trust="dbFDIUser"/>
```

Parameters:

- **`participant`** – String *Not required.* Defaults to a global trust (all participants). Sets the id of the participant that will be added the trust.
- **`action`** – String [”add”, ”delete”]*Required.* Defines the operation to execute:
  - **add**: adds a participant trust.
  - **delete**: deletes a participant trust.
- **`skipauth`** – boolean *Not required.* Defaults to false

  Wheather the directory session skips the autentication for the participant if set, or skips authentication for all participants, otherwise.
- **`trust`** – String *Not required.* Defaults to null.

  Sets the directory provider’s user name that will impersonate a Fuego participant when she connects to the Directory.

  If a null value is set, a Fuego participant can connect to the Directory using any directory provider’s user name

---

4.11 Type fuego:passport

Definition of a directory passport used to connect to a directory service.

It includes information about a Fuego Directory plus credentials to establish a connection to it.

Example:

```
<!-- Define a fuego "passport" -->
<fuego:passport id="fuego.passport"
       directoryid="${fuego.directoryid}"
       participant="${fuego.participant}"
          password="${fuego.password}" />
```

Parameters:

- **`participant`** – String *Required.* Unless `preset` is defined ID of the participant that will connect to the directory.
- **`directoryid`** – String *Required.* either directoryid or directoryurl must be defined The ID of the Directory as defined in the `directory.properties` file.
- **`password`** – String The password of the participant.
- **`directoryurl`** – String *Required.* either directoryid or directoryurl must be defined The URL of the Fuego Directory.
- **`properties`** – File Use a different `directory.properties` file instead of the default one located in the `conf/` directory.
- **`refid`** – Reference
- **`preset`** – String *Required.* Unless `participant` is defined. A preset in the `directory.properties` file that contains connection properties.

4.12 Task fuego:publish

This task allows to automatically publish and (optionally) deploy a complete Fuego project.

Particular processes within a project can be included or excluded from the publication by using the `include` and `exclude` nested elements.

It can handle the mapping of role/variables/resources, by either:

- Explicity defining them.
- Loading the projects' `publication.xml` definition file.

Example:

```
...
<!-- Open a session to the Fuego directory -->
  <fuego:session passportref="fuego.passport">

    <!-- Publish processes -->
    <fuego:publish fpr="${fuego.project}"
                   deploy="true"
                   engine="${fuego.engine}">

        <!-- Role mappings -->
        <fuego:rolemap abstract="Approver" real="CFO"/>
        <fuego:rolemap abstract="Requester" real="Consultant"/>

    </fuego:publish>
</fuego:session>
...
```

Parameters:

- **`importcustomviews`** – boolean *Not required.* Defaults to false Import custom portal Views and Presentations from the project (defined with Studio's PortalAdmin), taking into account the roles and variables mappings.
- **`haltonerror`** – boolean *Not required.* Defaults to false Whether to stop the ant build process if an error occurs during the execution of this task.
- **`fpr`** – File *Required.* Unless <`projects`> nested element is used. Fuego Project file to use for publishing/deploying.
- **`ou`** – String *Not required.* Organizational Unit where processes will be deployed. Only meaninful when `deploy` is true
- **`deploy`** – boolean *Not required.* Defaults to false Whether to deploy the process after publishing
- **`automapconfigs`** – boolean *Not required.* Defaults to false Automatically map External Resources configurations (as defined in the project design) to real Configurations with the same name (as defined in the Fuego Enterprise directory). Particularly useful when using the `importdata` option.
- **`keepgeneratedfiles`** – boolean *Not required.* Defaults to the user's studio preference Do not remove the .java and .class files generated during publication. It overrides the studio preference.
- **`remarks`** – String *Not required.* Optional commentary to add to the processes publication.

- **`importdata`** – boolean *Not required.* Defaults to false TODO I've tried, but it fails lo load my project :(Whether to import data from the project, as defined in Fuego Studio. This includes importing:
  - Holiday and Calendar rules
  - Organizational Units
  - Roles
  - Resource configurations
  - External Variables
- **`automapvars`** – boolean *Not required.* Defaults to false Automatically map external variable names (as defined in the project design) to an external variable id with the same name (as defined in the Fuego Enterprise directory). Particularly useful when using the `importdata` option.
- **`automaproles`** – boolean *Not required.* Defaults to false Automatically map abstract roles to real ones with the same name. Particularly useful when using the `importdata` option.
- **`viewsgenerationtype`** – String ["unified_inbox", "by_process", "by_process_and_activity"]*Not required.* Defaults to "unified_inbox" type Choose which type of views will be generated for the deployed processes.
- **`engine`** – String *Required.* when `deploy` is true Name of the Engine for deployment
- **`loadpublicationinfo`** – boolean *Not required.* Defaults to false Use the publication information defined in the `projectname.frp/publication.xml` file. Including:
  - Role mappings
  - Resource configuration mappings
  - External Variable mappings
- **`excludeall`** – boolean

Parameters accepted as nested elements:

`<varmap>` Represents a mapping of an external variable name (as defined in the project design) to an external variable (as defined in the Fuego Enterprise directory)

Parameters:

- **`externalid`** – String *Required.*
- **`name`** – String

`<include>` Declares a particular process to be included in the publication/deployment. If no `<include>` element is used, all the process in the project will be included.

Parameters:

- **`name`** – String *Required.*
- **`variation`** – String *Not required.* If not specified `"Default"` is assumed.

`<exclude>` Declares a particular process to be excluded from the publication/deployment.

Parameters:

- **`name`** – String *Required.*
- **`variation`** – String *Not required.* If not specified `"Default"` is assumed.

`<projects>` (Of type FileSet)

Parameters:

- **`refid`** – Reference
- **`file`** – File
- **`dir`** – File
- **`defaultexcludes`** – boolean
- **`followsymlinks`** – boolean
- **`casesensitive`** – boolean
- **`excludes`** – String
- **`includesfile`** – File
- **`excludesfile`** – File
- **`includes`** – String

`<rolemap>` Represents a mapping of an abstract Role (as defined in the project design) to real Role (as defined in the Directory)

Parameters:

- **`abstract`** – String *Required.*
- **`real`** – String *Not required.* If omitted, the same id specified in `abstract` role will be used (Useful when using the `importdata` option). See `automaprole`.

`<confmap>` Represents a mapping of an abstract Resource Configuration name (as defined in the project design) and a real Real Configuration (as defined in the Directory)

Parameters:

- **`local`** – String
- **`real`** – String

4.13 Task fuego:resources

Handles export/import of configuration resources.

Examples:

```
<target name="export-resources" description="Export Configuration Resources
    ">

  <!-- Open a session to the Fuego directory -->
  <fuego:session
          passportref="fuego.passport"
          haltonerror="true" >

     <fuego:resources action="export"
                      file="resources.exp" />

  </fuego:session>

</target>
```

```
<target name="import-resources" description="Import Configuration Resources
    ">

  <!-- Open a session to the Fuego directory -->
  <fuego:session
          passportref="fuego.passport"
          haltonerror="true" >

     <fuego:resources action="import"
                      force="true"
                      file="resources.exp" />

  </fuego:session>

</target>
```

Parameters:

- **`file`** – File *Required.* The file to export/import

- **`force`** – boolean *Not required.* by default, it will refuse to import a configuration if it already exists in the directory. Whether the `import` action will overwrite existing configurations.
- **`action`** – String [”import”, ”export”]*Required.* Defines the operation to execute on the configuration.
  - **import**: create the objects in the file in the directory
  - **export**: save the objects in the directory in the file
- **`haltonerror`** – boolean *Not required.* Default is true. Stop the Ant build process if an error occurs.

4.14 Task fuego:session

Establishes a session with a Fuego Directory. The Tasks contained as nested elements are executed using this session.

An external Fuego Passport can be referenced for convenience.

All the Fuego Ant tasks that use a directory session must be a children of this task.

Example:

```
<!-- Define a fuego "passport" -->
<fuego:passport id="fuego.passport"
       directoryid="${fuego.directoryid}"
       participant="${fuego.participant}"
          password="${fuego.password}" />


<target name="publish">

  <!-- Open a session to the Fuego directory -->
  <fuego:session
        passportref="fuego.passport"
            verbose="true"
         properties="${fuego.basedir}/conf/directory.properties"
        haltonerror="true" >

       ... include other Fuego tasks here...

  </fuego:session>

</target>
```

Parameters:

- **`haltonerror`** – boolean *Not required.* Default is false Stop the build process if an error occurs.

- **`directorypreset`** – String
- **`verbose`** – boolean
- **`passportref`** – Reference *Required.* Unless `directoryid` or `directoryurl` are specified. Use a passport defined previously instead of defining all the properties in the session task
- **`directoryid`** – String *Required.* Unless `passportref` or `directoryurl` are specified. The id of the Fuego Directory to used, as defined in the `directory.-properties` file.
- **`properties`** – File *Not required.*

  Defaults to `${fuego.basedir}/conf/directory.properties`

  Use a different directory.properties file instead of the default one locate in the conf dir
- **`fuegobasedir`** – File
- **`loaderref`** – Reference
- **`participant`** – String *Required.* Unless `passportref` is specified. ID of the participant to use
- **`password`** – String *Required.* Unless `passportref` is specified. Password for the participant
- **`directoryfile`** – File
- **`preset`** – String *Not required.* Specifies a preset to use in the Directory Passport.

Parameters accepted as nested elements:

This Task is a Container (it accepts nested Tasks).

4.15 Task fuego:undeploy

This task allows to automatically undeploy a complete Fuego project.

Example:

```
...
<!-- Open a session to the Fuego directory -->
<fuego:session passportref="fuego.passport">

    <fuego:undeploy projectName="variationsTest"/>

</fuego:session>
...
```

Parameters:

- **`projectname`** – String *Required.* Project to be undeployed
- **`haltonerror`** – boolean *Not required.* Defaults to false Whether to stop the ant build process if an error occurs during the execution of this task.

4.16 Task fuego:unpublish

This task allows to automatically unpublish a complete Fuego project.

Example:

```
...
<!-- Open a session to the Fuego directory -->
<fuego:session passportref="fuego.passport">

    <fuego:unpublish projectName="variationsTest" major="2"/>

</fuego:session>
...
```

Parameters:

- **`major`** – int *Required.* Only when the minor version or the revision are set. Major version of the project to be unpublished. If no major version is specified, the whole project is unpublished.
- **`revision`** – int *Not required.* Defaults to being ignored. Revision of the project to be unpublished. If no revision is specified, every revision for the selected version will be unpublished.
- **`projectname`** – String *Required.* Project to be unpublished
- **`minor`** – int *Required.* Only when the revision is set. Minor version of the project to be unpublished. If no minor version is specified, every minor version for the selected major will be unpublished.

- **`haltonerror`** – boolean *Not required.* Defaults to false Whether to stop the ant build process if an error occurs during the execution of this task.

---

5 fuego.j2ee Library

5.1 Task fuego.j2ee:buildallprojects

This ant task generates an `ear` file for each deployed project in the engine.

Example:

```
<target name="build-all-projects" description="Build all projects ears">

    <fuego.j2ee:buildallprojects fuegobasedir="${fuego.basedir}"
        destDir="${destination.dir}"
        workdir="${fuego.workdir}"
        directoryfile="${fuego.directory.file}"
        engineid="${engine.name}"/>
</target>
```

Parameters:

- **`destdir`** – File *Required.* The destination directory of the generated archive files.
- **`directorypreset`** – String
- **`engineid`** – String *Required.* The Fuego Server identification. The server is used to obtain the Application Server Vendor and configure the EAR based on it.
- **`verbose`** – boolean
- **`fuegobasedir`** – File
- **`loaderref`** – Reference
- **`directoryfile`** – File
- **`workdir`** – File *Required.* The directory must exist and writable. A writable directory were temporary files are generated.

5.2 Task fuego.j2ee:buildear

Generates the `ear` files for a given engine or deployed project.

Example 1 Building an engine ear:

```
<target name="build-engine" description="Build engine ear">

    <fuego.j2ee:buildear fuegobasedir="${fuego.basedir}"
         destfile="${destination.file}" displayName="${displayName}"
                      workdir="${fuego.workdir}"
         directoryfile="${fuego.directory.file}"
         engineid="${engine.name}"/>
         includelibs="false"/>
         <fuego.j2ee:engine/>
</target>
```

Example 2 Building a project ear:

```
<target name="build-project" description="Build project ear">

    <fuego.j2ee:buildear fuegobasedir="${fuego.basedir}"
         destfile="${destination.file}" displayName="${displayName}"
                      workdir="${fuego.workdir}"
         directoryfile="${fuego.directory.file}"
         engineid="${engine.name}"/>
         includelibs="false"/>
         <fuego.j2ee:project name="${project.name}"/>
</target>
```

Parameters:

- **`includelibs`** – boolean *Not required.* Defaults to False Include the required libs for the ear inside the ear itself.
- **`directorypreset`** – String
- **`destfile`** – File *Required.* The file name of the destination archive file.
- **`engineid`** – String *Required.* The Fuego Server identification. The server is used to obtain the Application Server Vendor and configure the EAR based on it.
- **`verbose`** – boolean
- **`displayname`** – String *Not required.* The description of the ear application.
- **`fuegobasedir`** – File
- **`loaderref`** – Reference

- **`templateset`** – String
- **`directoryfile`** – File
- **`workdir`** – File *Required.* The directory must exist and writable. A writable directory were temporary files are generated.

Parameters accepted as nested elements:

`<engine>` Used to indicate the creation of the engine ear

Parameters:

`<project>` Represents a deployed project

Parameters:

- **`name`** – String *Required.*

5.3 Task fuego.j2ee:buildportal

Task to build the Portal Web Application as a war file (or its expanded version).

Example:

```
<target name="build-portal" description="Build portal war">
    <fuego.j2ee:buildportal fuegobasedir="${fuego.basedir}"
         destfile="${destination.file}"
         workdir="${fuego.workdir}"
         directoryfile="${fuego.directory.file}"
         engineid="${engine.name}"
         uri="/fuego/portal"/>
</target>
```

Parameters:

- **`destdir`** – File
- **`uri`** – String
- **`directorypreset`** – String
- **`destfile`** – File
- **`engineid`** – String
- **`verbose`** – boolean

- **`fuegobasedir`** – File
- **`loaderref`** – Reference
- **`templateset`** – String
- **`directoryfile`** – File
- **`workdir`** – File

5.4 Task fuego.j2ee:buildportaladmin

Task to build the Portal Administration Web Application as a war file (or its expanded version).

Example :

```
<target name="build-portaladmin" description="Build portal admin war">
    <fuego.j2ee:buildportaladmin fuegobasedir="${fuego.basedir}"
         destfile="${destination.file}"
         workdir="${fuego.workdir}"
         directoryfile="${fuego.directory.file}"
         engineid="${engine.name}"
         uri="/fuego/portaladmin"/>
</target>
```

Parameters:

- **`destdir`** – File
- **`uri`** – String
- **`directorypreset`** – String
- **`destfile`** – File
- **`engineid`** – String
- **`verbose`** – boolean
- **`fuegobasedir`** – File
- **`loaderref`** – Reference
- **`templateset`** – String
- **`directoryfile`** – File
- **`workdir`** – File

5.5 Task fuego.j2ee:deploy

Deploys `ear` files to the App Server.

The deployment is done via the Fuego J2EE Deployer application, which must be deployed and available for this task to work.

Refer to the Fuego J2EE Deployer documentation for details.

Example:

```
<target name="deploy-ear" description="Deploy ear">
    <fuego.j2ee:deploy
           earFile="${ear.file}"
           targetName="${server.name}"
           url="http://host:port/fuego/deployer/servlet/worker"/>
</target>
```

Parameters:

- **`targetname`** – String *Required.* The name of the server or cluster were the operation will be performed.
- **`url`** – String
- **`startafterdeploying`** – boolean *Not required.* Defaults to False Start the applications after being deployed.
- **`verbose`** – boolean
- **`earspath`** – String *Required.* cannot be used together with **earfile** Specify multiple ear files located in this path.
- **`fuegobasedir`** – File
- **`loaderref`** – Reference
- **`cluster`** – boolean *Not required.* Defaults False If the target WebSphere server is clustered or not.
- **`failonerror`** – boolean
- **`earfile`** – String *Required.* cannot be used together with **earspath** Specify the ear file to be deployed

5.6 Task fuego.j2ee:showAppStatus

This task shows an application status in a J2EE Application Server.

It relies on the Fuego J2EE Deployer application, which must be deployed and available for this task to work.

Example:

```
<target name="show-status" description="Show application status">
    <fuego.j2ee:showAppStatus applicationName="${application.name}"
        url="http://host:port/fuego/deployer/servlet/worker"/>
</target>
```

Parameters:

- **`url`** – String
- **`verbose`** – boolean
- **`earspath`** – String *Required.* cannot be used together with **earfile** or **applicationName** Specify multiple applications based on local ear files located in this path.
- **`applicationname`** – String *Required.* cannot be used together with **earspath** or **earFile** Specify the application to process using the name defined for it in the Application Server. This name is tipically shown in the Application Server administrative console.
- **`fuegobasedir`** – File
- **`loaderref`** – Reference
- **`failonerror`** – boolean
- **`earfile`** – String *Required.* cannot be used together with **earspath** or **applicationName** Specify the application to process based on a local ear file.

5.7 Task fuego.j2ee:startApp

Starts an application in a J2EE Application Server.

It relies on the Fuego J2EE Deployer application, which must be deployed and available for this task to work.

Example:

```
<target name="start-application" description="Start application">
    <fuego.j2ee:startApp applicationName="${application.name}"
        url="http://host:port/fuego/deployer/servlet/worker"/>
</target>
```

Parameters:

- **`url`** – String
- **`verbose`** – boolean
- **`earspath`** – String *Required.* cannot be used together with **earfile** or **applicationName** Specify multiple applications based on local ear files located in this path.
- **`applicationname`** – String *Required.* cannot be used together with **earspath** or **earFile** Specify the application to process using the name defined for it in the Application Server. This name is tipically shown in the Application Server administrative console.
- **`fuegobasedir`** – File
- **`loaderref`** – Reference
- **`failonerror`** – boolean
- **`earfile`** – String *Required.* cannot be used together with **earspath** or **applicationName** Specify the application to process based on a local ear file.

5.8 Task fuego.j2ee:stopApp

Stops an application in a J2EE Application Server.

It relies on the Fuego J2EE Deployer application, which must be deployed and available for this task to work.

Example:

```
<target name="stop-application" description="Stop application">
    <fuego.j2ee:stopApp applicationName="${application.name}"
        url="http://host:port/fuego/deployer/servlet/worker"/>
</target>
```

Parameters:

- **`url`** – String
- **`verbose`** – boolean
- **`earspath`** – String *Required.* cannot be used together with **earfile** or **applicationName** Specify multiple applications based on local ear files located in this path.
- **`applicationname`** – String *Required.* cannot be used together with **earspath** or **earFile** Specify the application to process using the name defined for it in the Application Server. This name is tipically shown in the Application Server administrative console.
- **`fuegobasedir`** – File
- **`loaderref`** – Reference
- **`failonerror`** – boolean
- **`earfile`** – String *Required.* cannot be used together with **earspath** or **applicationName** Specify the application to process based on a local ear file.

5.9 Task fuego.j2ee:undeploy

Undeploys an application from a J2EE Application Server.

It relies on the Fuego J2EE Deployer application, which must be deployed and available for this task to work.

Example:

```
<target name="undeploy-ear" description="Undeploy ear">
    <fuego.j2ee:undeploy earFile="${ear.file}"
        url="http://host:port/fuego/deployer/servlet/worker"/>
</target>
```

Parameters:

- **`url`** – String
- **`verbose`** – boolean
- **`earspath`** – String *Required.* cannot be used together with **earfile** or **applicationName** Specify multiple applications based on local ear files located in this path.

- `applicationname` – String *Required.* cannot be used together with **earspath** or **earFile** Specify the application to process using the name defined for it in the Application Server. This name is tipically shown in the Application Server administrative console.
- `fuegobasedir` – File
- `loaderref` – Reference
- `failonerror` – boolean
- `earfile` – String *Required.* cannot be used together with **earspath** or **applicationName** Specify the application to process based on a local ear file.