GuideLines H2o Build 2.0

Added by MARTINOTTI SIMONE, last edited by VATACHI ALEXANDRU on Jul 08, 2014

How to create a new project

Project structure is completely free and managed from development team.
Only the following files are required:

- build.properties
- build.xml
- mapping.xml

How to write a build.properties file

The file will contains all build configuration specific for a environment.
Each project will have this file in the project root folder and it will be substituted during the centralized build with a version specific for each environment.
The one committed in SVN can contains a template that each each developer can overwrite with his own configurations.

Required/available values

During the centralized build, only a restricted set of configurations will be available for compilation.
Here you can find the list of all the available values:

- build.commons
- java.home
- wls.home
- xmlbeans.home
- external.dependency.home
- ~~h2o.be.lib~~ *(Deprecated in Weblogic 12.1.2.0.0)*
- ~~h2o.fe.lib~~ *(Deprecated in Weblogic 12.1.2.0.0)*
- ~~h2o.in.lib~~ *(Deprecated in Weblogic 12.1.2.0.0)*
- ~~h2o.be.sys~~ *(Deprecated in Weblogic 12.1.2.0.0)*
- ~~h2o.fe.sys~~ *(Deprecated in Weblogic 12.1.2.0.0)*
- ~~h2o.in.sys~~ *(Deprecated in Weblogic 12.1.2.0.0)*
- ~~h2o.be.ejb~~ *(Deprecated in Weblogic 12.1.2.0.0)*
- ~~h2o.fe.ejb~~ *(Deprecated in Weblogic 12.1.2.0.0)*
- ~~h2o.in.ejb~~ *(Deprecated in Weblogic 12.1.2.0.0)*
- ~~h2o.fe.war~~ *(Deprecated in Weblogic 12.1.2.0.0)*
- ~~h2o.in.war~~ *(Deprecated in Weblogic 12.1.2.0.0)*
- ~~h2o.client.lib~~ *(Deprecated in Weblogic 12.1.2.0.0)*
- apps.batch.lib
- apps.intranet.lib
- apps.internet.lib
- apps.batch.ejb
- apps.intranet.ejb
- apps.internet.ejb
- apps.intranet.webapp
- apps.internet.webapp
- sys.common.lib
- test.common.lib
- pattern.home

Mount point

In order to have all the build dependencies available, the following Samba share has to be mounted in the developers machine:

```
//buildcvs/buildshare (Deprecated in Weblogic 12.1.2.0.0)
//build/public
```

Example

Here you can find an example of build.properties developers configuration:

```
mount.point=/shared # Folder path of BuildSystem shared filesystem mount
environment=TEST # PP and PRO are also available

java.home=/java-home # Path to your local Java home
build.commons=${mount.point}/build-commons
```

```
wls.home=${mount.point}/weblogic-home
external.dependency.home=${mount.point}/external-dependencies
xmlbeans.home=${mount.point}/xmlbeans-home
classpath.home=${mount.point}/compilation-classpath
pattern.folder=${mount.point}/pattern-home

# H2O library locations
h2o.be.lib=${classpath.home}/${environment}/apps/BEdomain/BEcluster/lib (Depre
h2o.fe.lib=${classpath.home}/${environment}/apps/FEdomain/FEcluster/lib (Depre
h2o.in.lib=${classpath.home}/${environment}/apps/FEdomain/INcluster/lib (Depre
h2o.be.sys=${classpath.home}/${environment}/system/BEdomain/BEcluster/lib (Dep
h2o.fe.sys=${classpath.home}/${environment}/system/FEdomain/FEcluster/lib (Dep
h2o.in.sys=${classpath.home}/${environment}/system/FEdomain/INcluster/lib (Dep
h2o.be.ejb=${classpath.home}/${environment}/apps/BEdomain/BEcluster/ejb (Depre
h2o.fe.ejb=${classpath.home}/${environment}/apps/FEdomain/FEcluster/ejb (Depre
h2o.in.ejb=${classpath.home}/${environment}/apps/FEdomain/INcluster/ejb (Depre
h2o.fe.war=${classpath.home}/${environment}/apps/FEdomain/FEcluster/war/h2o/FE
h2o.in.war=${classpath.home}/${environment}/apps/FEdomain/INcluster/war/h2o/IN
h2o.client.lib=${classpath.home}/${environment}/apps/ISEclient/lib (Deprecated

apps.batch.lib=${classpath.home}/${environment}/apps/batch/lib
apps.intranet.lib=${classpath.home}/${environment}/apps/intranet/lib
apps.internet.lib=${classpath.home}/${environment}/apps/internet/lib
apps.batch.ejb=${classpath.home}/${environment}/apps/batch/ejb
apps.intranet.ejb=${classpath.home}/${environment}/apps/intranet/ejb
apps.internet.ejb=${classpath.home}/${environment}/apps/internet/ejb
apps.intranet.webapp=${classpath.home}/${environment}/apps/intranet/webapp
apps.internet.webapp=${classpath.home}/${environment}/apps/internet/webapp
sys.common.lib=${classpath.home}/${environment}/system/common/lib
test.common.lib=${classpath.home}/${environment}/test/common/lib

# Pattern location
pattern.home=${pattern.folder}/${environment}
```

How to write a build.xml file

There are two different type of build.xml

Master build.xml

This file is placed in the project root folder and has to contains only the references to the other artifacts.
Content example:

```
<?xml version="1.0" encoding="UTF-8"?>

<project name="ProjectName">

        <property file="build.properties"/>
        <import file="${build.commons}/build-master.xml"/>

        <property name="project.name" value="ProjectName"/>
        <property name="project.version" value="1.0.0"/>
        <property name="project.home" value="${basedir}"/>

        <target name="module-definition">
                <ant dir="ArtifactName" target="${target}"/>
        </target>

</project>
```

- *ProjectName* and *ArtifactName* has to be substituted with the right values.
- *build.properties* and *build-master.xml* inclusion, on top project definition, is required in order to use the build.commons framework functionality.

Artifact build.xml

This file is placed inside the artifact folder and it is used to build the specific artifact.
Content example:

```
<project name="ArtifactName" default="compile" basedir=".">

        <import file="${build.commons}/build-lib.xml"/>

        <path id="classpath.compile">
                <pathelement location="LibraryLocation"/>
        </path>

</project>
```

- *build-<artifact_type>.xml* inclusion is required in order to use the build.commons framework functionality.
    - The following build scripts are available build.commons framework:
        - build-config.xml
        - ~~build-ejb20.xml~~ *(Deprecated in Weblogic 12.1.2.0.0)*
        - build-ejb.xml
        - build-lib.xml
        - build-master.xml
        - build-pdf.xml
        - build-web.xml
        - build-xsd.xml
- *classpath.compile* settings is required for classpath definition
- *classpath.test.extension* is available for unit-test classpath definition

How to build a project

Prepare the environment

In order to execute a project build, the following environment variable has to be exported:

- ***JAVA_HOME***
- ***ANT_HOME***

Execute Ant

Run the following command from the root folder of project (where main build.xml is avaiable):

```
ant <target>
```

The following target are available for Ant execution:

- *clean*
- *compile*
- *test*
- *package*
- ~~*coverage*~~ *(Deprecated in Weblogic 12.1.2.0.0)*
- ~~*site*~~ *(Deprecated in Weblogic 12.1.2.0.0)*
- *javadoc*

For example, during centralized projects build, the following command is executed:

```
ant clean package
```

How to build a mapping.xml

This file has to be inserted in the root folder of the project and it will be used by the BuildSystem to distribute an artifacts in a specific clusters.

File format

mapping.xml is a standard XML file having the following structure:

```
<mapping>
    <destinations>
        <destination>
            <artifact/>
        </destination>
    </destinations>
</mapping>
```

Destination definition

Using *destination* element of *destinations*, you can specify which *artifacts* has to deployed in the configured cluster. **name** attribute of *destination* can have one of the following values:

- ~~backend (*BEcluster*)~~ *(Deprecated in Weblogic 12.1.2.0.0)*
- ~~backendbatch (*BHcluster*)~~ *(Deprecated in Weblogic 12.1.2.0.0)*
- ~~client (*APP*)~~ *(Deprecated in Weblogic 12.1.2.0.0)*
- ~~daemon (*Daemon*)~~ *(Deprecated in Weblogic 12.1.2.0.0)*
- ~~internet (*FEcluster*)~~ *(Deprecated in Weblogic 12.1.2.0.0)*
- ~~intranet (*INcluster*)~~ *(Deprecated in Weblogic 12.1.2.0.0)*

- ~~intranetbatch (*IHcluster*)~~ *(Deprecated in Weblogic 12.1.2.0.0)*
- ~~pdf (*PDFServer*)~~ *(Deprecated in Weblogic 12.1.2.0.0)*
- batch-ear
- intranet-ear
- internet-ear
- extra

For example, in your *mapping.xml*, you can define the following *destination*:

```
<mapping>
    <destinations>
        <destination name="internet">
            ...
        </destination>
        <destination name="intranet-ear">
            ...
        </destination>
    </destinations>
</mapping>
```

Artifact definition

Using *artifact* element of *destination*, you can specify the artifact type of your module.
name attribute define your module name, **type** can have one of the following values:

- lib
- ejb
- config
- system-config
- web
- static
- external
- pdf
- test

artifact mapping on *destination* is totally in charge of development teams and they have the complete control on artifacts distribution. Development teams has to distribute an artifact to a destination **only** if used/required.

Here you can find the complete map of supported *artifact* type for each *destination*:

- ~~backend~~ *(Deprecated in Weblogic 12.1.2.0.0)*
    - lib
    - ejb
    - external
    - test
    - system-config
    - config
- ~~backendbatch~~ *(Deprecated in Weblogic 12.1.2.0.0)*
    - system-config
    - config
    - ejb
    - lib
    - external

- ~~client~~ *(Deprecated in Weblogic 12.1.2.0.0)*
    - system-config
    - config
    - external
    - lib
- ~~daemon~~ *(Deprecated in Weblogic 12.1.2.0.0)*
    - system-config
    - config
    - external
    - lib
- ~~internet~~ *(Deprecated in Weblogic 12.1.2.0.0)*
    - web
    - static
    - lib
    - system-config
    - config
    - ejb
    - external
- ~~intranet~~ *(Deprecated in Weblogic 12.1.2.0.0)*
    - web
    - static
    - lib
    - system-config
    - config
    - ejb
    - external
- ~~intranetbatch~~ *(Deprecated in Weblogic 12.1.2.0.0)*
    - web
    - static
    - lib
    - system-config
    - config
    - ejb
    - external
- ~~pdf~~ *(Deprecated in Weblogic 12.1.2.0.0)*
    - pdf
- batch-ear
    - web
    - static
    - lib
    - system-config
    - config
    - ejb
    - external
- internet-ear
    - web
    - static
    - lib
    - system-config
    - config
    - ejb
    - external
- intranet-ear

  - web
  - static
  - lib
  - system-config
  - config
  - ejb
  - external
- extra
  - pdf
  - test

Remember that no all the *artifact* type can be deployed in a specific *destination*. This means that, for example, *test* cannot be released in *backend*.
For example, in your *mapping.xml*, you can define the following *destination*:

```
<mapping>
    <destinations>
        <destination name="backend">
            <artifact name="projectname-lib-core" type="lib"/>
            <artifact name="projectname-ejb-core" type="ejb"/>
        </destination>
        <destination name="intranet">
            <artifact name="projectname-web-intranet" type="web"/>
            <artifact name="projectname-ejb-core" type="ejb"/>
        </destination>
    </destinations>
</mapping>
```

Migrations

From Weblogic 10.3.3.0 to Weblogic 12.1.2.0.0

Classpath definition

In Weblogic 12, the classpath definition is changed due to the different cluster configuration.

- For classpath definition migration, you have to simply include the new definition by migrating your dependencies using this map:

10.3.3.0	**12.1.2.0.0**
h2o.be.lib	apps.intranet.lib
h2o.in.lib	apps.intranet.lib
h2o.fe.lib	apps.internet.lib
h2o.be.ejb	apps.intranet.ejb
h2o.in.ejb	apps.intranet.ejb

h2o.fe.ejb	apps.internet.ejb
h2o.client.lib	Dependencies not available anymore
h2o.be.sys	Dependencies available by default for all the project
h2o.in.sys	Dependencies available by default for all the project
h2o.fe.sys	Dependencies available by default for all the project

- If you are referring to a local project module, you have to change the definition using the following map:

10.3.3.0	12.1.2.0.0
/build	/target/build/compile

- If you are referring to an InternaLib/ExternalLib dependencies, you have to use the use the map in in attachment:
  - Migration_H2oBase.xls
- If you are referring to compilation pattern, you have to change the definition using the following map:

10.3.3.0	12.1.2.0.0
${pattern.home}/INwebapp	${pattern.home}/intranet
${pattern.home}/FEwebapp	${pattern.home}/internet

Remember to all the old definitions when the infrastructure migration will be concluded!

Example

- Actual definition:

```
<pathelement location="${project.home}/anagrafe-lib-core/build"/>
<pathelement location="${h2o.be.lib}/Contabile/contabile-lib-core-1.0.0.j
<pathelement location="${h2o.in.lib}/121Core7/121core7-lib-intranet-1.0.0
<pathelement location="${h2o.client.lib}/ReteFTF/reteftf-lib-client-1.0.0
<pathelement location="${h2o.be.lib}/InternalLib/commons-lang-2.3.jar"/>
<pathelement location="${h2o.be.sys}/crypting.jar"/>
```

- New definition:

```
<!-- Weblogic 10.3.3.0 -->
<pathelement location="${project.home}/anagrafe-lib-core/build"/>
<pathelement location="${h2o.be.lib}/Contabile/contabile-lib-core-1.0.0.j
<pathelement location="${h2o.in.lib}/121Core7/121core7-lib-intranet-1.0.0
<pathelement location="${h2o.client.lib}/ReteFTF/reteftf-lib-client-1.0.0
<pathelement location="${h2o.be.lib}/InternalLib/commons-lang-2.3.jar"/>
<pathelement location="${h2o.be.sys}/crypting.jar"/>

<!-- Weblogic 12.1.2.0.0 -->
<pathelement location="${project.home}/anagrafe-lib-core/target/build/com
<pathelement location="${apps.intranet.lib}/Contabile/contabile-lib-core-
<pathelement location="${apps.internet.lib}/121Core7/121core7-lib-interne
<pathelement location="${apps.intranet.lib}/H2oBase/commons-lang-2.3.jar"
<fileset file="${sys.common.lib}/realm/securitycore-lib-*.jar"/>
```

EJB build

Since EJB 1.1 has been deprecated, compilation script for this artifact has been in a single ANT script called **build-ejb.xml**.
This means that:

- **build-ejb20.xml** doesn't exist anymore and all the reference has to be migrated to the new one.
- If you are referring to **build-ejb.xml**, your classes will be compiled the EJB standard.

Example

- Actual definition:

```
<import file="${build.commons}/build-ejb20.xml"/>
```

- New definition:

```
<import file="${build.commons}/build-ejb.xml"/>
```

Build workflow

Build workflow has been aligned with Maven standards, this means that if you have overwritten some targets, the compatibility with new **build-commons** has be verified.

Example

```
<target name="test-core">
    <echo>test not needed</echo>
</target>
```

- This definition is not compatible with new system for 3 reasons:
  - **Not required:** Unit-test are not executed by default is dedicated folder is not available
  - **Wrong definition:** Core targets should never be ovrewritten
  - **Break build workflow:** New definition of test-core target depends on compile

Mapping xml

Mapping file definition is change and old entries will not be considered anymore in new system.
Artifact-cluster map has to be changed by adding the following entries:

- batch-ear
- intranet-ear
- internet-ear
- extra

Migrated project example

Here you can find a migrated project that can be used as an example:

```
https://svn.bansel.it/h2o/ElectronicSignature/branches/TestWeblogic12Migratior
```

Printed by Atlassian Confluence 3.4.9, the Enterprise Wiki.