Framework Configuration Guide

Table of Contents

Related Documents

- Entity Engine Configuration Guide
- Service Engine Configuration Guide

Introduction

This document describes the configuration of the Framework of the Open For Business Framework. It goes through each of the OFBiz Framework properties files to explain the available properties and their usage. The properties files used for the OFBiz applications have examples of the different options and are located in **`ofbiz/commonapp/etc/`**.

These files are resources on the classpath so they must be located somewhere on the classpath available to the Java application(s) for the proper operation of the OFBiz Framework. Because they are resources on the classpath they can be overridden locally in webapps by putting a changed copy of the desired file in the webapp/WEB-INF/classes directory.

The entityengine.xml file is not described in this document, it is described in the Entity Engine Configuration Guide as a separate document because of the large number of options in the file. See the Related Documents section for a link to the Entity Engine Configuration Guide. The same is true for the Service Engine Configuration Guide and the serviceengine.xml file, including a link in the Related Documents section above.

Properties File Extensions

The OFBiz UtilProperties class uses the OFBiz FlexibleProperties to read and parse properties files with a few extensions. The FlexibleProperties class is an extension of the standard Java Properties class.

The extensions included allow the value of a property to be substituted into the value of another property by simply using the ${property.name} syntax. Another extension is that if an "env." precedes a property.name inside the ${ and } delimiters, then the named property will be looked up in the system environment instead of in the current properties file.

For example the ofbiz.home system property is added using the -D parameter on the java command line and is used in the entityengine.xml and other properties files with **`${env.ofbiz.home}`**.

Note that these extensions are only currently available for properties files used in configuration.

cache.properties

The cache.properties file contains cache setting used by the UtilCache class. Default cache parameters can be passed in when a cache is created but, if cache parameters exist in this file for the given cache name, then they will be loaded from the cache.properties file.

There are three optional properties in this file for each named cache:
    `{cache-name}.maxSize` (whole number)
    `{cache-name}.expireTime` (whole number)
    `{cache-name}.useSoftReference` (true or false).
The maxSize property specifies the maximum number of entries in the cache, but does nothing with respect to the size of each cache entry. The expireTime property specifies how long in milliseconds each cache entry will be valid. If either value is 0 the feature will be disabled, i.e. unlimited cache size or no expire time.

The useSoftReference property is used to specify, true or false, whether soft references should be used for cache entries. Soft references help keep large caches from taking too much memory by allowing the garbage collector to clear these entries when more memory is needed.

If the values are not specified and are not passed to the UtilCache constructor, the values in the **`default.maxSize`**, **`default.expireTime`** and **`default.useSoftReference`** properties will be used.

debug.properties

The debug.properties file contains options to turn OFBiz debug threads on/off and also serves as the Log4J properties file (see jakarta.apache.org for documentation on Log4J).

In addition to the Log4J properties there are properties that turn OFBiz logging levels on and off. These are formatted like **`print.{level-name}`** where level-name is **`verbose`**, **`timing`**, **`info`**, **`important`**, **`warning`**, **`error`**, or **`fatal`**.

jndiservers.xml

This is a simple XML file that contains a number of **`jndi-server`** tags inside a single **`jndi-config`** tag. Each jndi-server tag corresponds to a named JNDI Server configuration that will be used in various places in the OFBiz Framework, especially in the Entity and Service Engines.

The **`jndi-server`** tag is used to configure JNDI servers that will be used for JTA objects, JDBC objects, JMS objects, and other JNDI resources used by the Entity Engine, the Service Engine, and other framework or application components.

Attribute Name	Required?	Description
name	Y	The name of the JNDI Server. Used in other tags in the 'jndi-server-name' attribute. This is the only required attribute. The others can be left out to use the default constructor of the InitialContext class, which will load the JNDI parameters from the jndi.properties file, which most app servers include.
context-provider-url	N	Example: rmi://127.0.0.1:1099
initial-context-factory	N	Example: com.sun.jndi.rmi.registry.RegistryContextFactory
url-pkg-prefixes	N	Example: java.naming.rmi.security.manager
security-principal	N	JNDI Username
security-credentials	N	JNDI Password

localdtds.properties

The localdtds.properties file contains DTD names and the name of the .dtd file on the classpath that can be used for that name. This is used to accomplish local loading of DTDs rather than always getting them over the internet when an XML file is loaded and validated. This file is used by the UtilXml class.

The properties in this file do not follow the standard name=value style properties because DTD names are not legal properties file names. So, they are specified using two attributes: **`name.{number}`** and **`value.{number}`**. When looking for the value for a given DTD name the **`name.1`** property is looked at first and then the number is incremented until the current number is not found. So, if you have 8 name/value pairs then your highest number for both **`name.{number}`** and **`value.{number}`** will be 8, with every number between 1 and 8 used. If, for example, the number 5 is skipped, then everything beyond 4 will be ignored.

An example of these properties: **`name.1=-//OFBiz//DTD Data Files//EN`** for the name and **`value.1=datafiles.dtd`** for the value. The **`datafiles.dtd`** resource will be searched for on the classpath.

security.properties

The security.properties file contains security related settings such as the minimum length for passwords. Currently the only property in the file is **`password.length.min`** which specifies the minimum number of characters in a UserLogin password.

serverstats.properties

The serverstats.properties file contains settings to control the gathering and persistence of server hit statistics. Statistics are gathered in two ways: logging each resource hit and logging a bin of resource hits where the bin may contain many hits that happen in a certain time period.

The **`stats.bin.length.millis`** property is used to specify the length of statistics bins and is a whole number milliseconds value.

To specify whether the bins for a certain resource should be persisted or not, use the **`stats.persist.{resource-name}.bin`** property where resource-name is **`REQUEST`** for Control Servlet requests, **`EVENT`** for Control Servlet Events, or **`VIEW`** for Control Servlet Views (JSPs mostly).

To specify whether each hit for a certain resource should be persisted or not use the **`stats.persist.{resource-name}.hit`** property where resource-name is **`REQUEST`** for Control Servlet requests, **`EVENT`** for Control Servlet Events, or **`VIEW`** for Control Servlet Views (JSPs mostly).

url.properties

The url.properties file is used to configure special server URLs for the cases where the URL of the current server is not sufficient. Examples of this include settings for automatically jumping to secure or non-secure areas of a site and referring to content that is on a different server or in a different server farm.

The first set of properties deals with creating links or redirecting to secure and non-secure areas of the site. This is primarily used by the <ofbiz:url> JSP tag and the ControlServlet to make sure that each request comes in how it should, either secure or not. Requests in the controller.xml file are marked as secure with an `https="true"` attribute on the `security` tag under a given `request-map` tag.

When requests are marked as secure they should go through the secure port. The ofbiz:url tag helps with this by creating an extended URL when necessary to go through the secure port. The ControlServlet also makes sure that when a secure request is required the request comes through a secure port, if not it will send a redirect to the same request through the secure port. The requests that are marked as not secure with `https="false"` behave just the opposite. This is done because it is more efficient and lighter on the server when the encryption and decryption do not have to be done.

This option of automatically going between the secure and non-secure ports can be turned on and off with the **`port.https.enabled`** property which should be either a **`Y or N`**.

To specify the secure (HTTPS) and non-secure (HTTP) ports and optionally host names to use with the ports, use the following properties: **`port.https`**, **`force.https.host`**, **`port.http`**, and **`force.http.host`**.

The next set of properties in the file is for static content URLs. This includes images and other resources that can be moved to another server or farm to lighten the load of the application servers and the load balancers in front of the application servers. This is an inexpensive and easy way to handle more load on a site.

There is one property for secure content: **`content.url.prefix.secure`**, and one for standard or non-secure content: **`content.url.prefix.standard`**.