

# Entity Engine

The Entity Engine from the [OFBiz project](#) is what JIRA uses to persist data to a database. You can find out more about why we chose the EE at the [bottom](#) of this page. See the [configuration overview](#) for a conceptual overview of what is being done here.

## 1. Configuring the Entity Engine for JIRA

The configuration of the Entity Engine is done through an XML file called `entityengine.xml`. This file is used to define parameters for persistence servers such as JDBC datasource parameters.

For JIRA, this file is located in the distribution at `edit-webapp/WEB-INF/classes/entityengine.xml`.

As outlined in the [overview](#), the settings which generally need to be configured are:

- **Transaction Factory** - see below
- **field type** - edit the `field-type-name` attribute of the `<datasource>` tag.
- **datasource location:**
  - edit the `jndi-name` attribute of the `<jndi-jdbc>` tag relevant to your database.
  - For certain schema-aware databases (**Postgres, DB2**), add a **schema-name** attribute specifying the schema the database uses. See note in `entityengine.xml`
  - For databases with table/column name length limits (eg. **DB2** on certain platforms), you may need to set a **constraint-name-clip-length** parameter. See note in `entityengine.xml`

### 1.1. Transaction Factory

By default the Entity Engine tries to obtain a JTA transaction factory from the application server using JNDI. This table shows the different values for different application servers:

**Orion, Tomcat 4.x, Tomcat 5.0, Jetty and Weblogic (see also the [Orion](#), [Resin](#), [Tomcat](#), [Jetty](#) and [Weblogic](#) guides)**

```
<transaction-factory class="org.ofbiz.core.entity.transaction.JNDIFactory">
  <user-transaction-jndi jndi-server-name="default"
    jndi-name="java:comp/UserTransaction"/>
  <transaction-manager-jndi jndi-server-name="default"
    jndi-name="java:comp/UserTransaction"/>
</transaction-factory>
```

**Tomcat 5.5 (see also the [Tomcat 5.5 install guide](#))**

```
<transaction-factory class="org.ofbiz.core.entity.transaction.JNDIFactory">
  <user-transaction-jndi jndi-server-name="default"
    jndi-name="java:comp/env/UserTransaction"/>
  <transaction-manager-jndi jndi-server-name="default"
    jndi-name="java:comp/env/UserTransaction"/>
</transaction-factory>
```

**Resin 3 (see also the [Resin 3 install guide](#))**

```
<transaction-factory
  class="org.ofbiz.core.entity.transaction.JNDIFactory">
  <user-transaction-jndi jndi-server-name="default"
    jndi-name="java:comp/UserTransaction" />
  <transaction-manager-jndi jndi-server-name="default"
    jndi-name="java:comp/TransactionManager" />
</transaction-factory>
```

**JBoss (see also the [JBoss 3.x](#) and [JBoss 4.x](#) guides)**

```
<transaction-factory class="org.ofbiz.core.entity.transaction.JNDIFactory">
  <user-transaction-jndi jndi-server-name="default"
    jndi-name="UserTransaction"/>
```

```
<transaction-manager-jndi jndi-server-name="default"
  jndi-name="java:/TransactionManager" />
</transaction-factory>
```

## 2. Altering the Entity Model

The Entity Model describes the table and column layout that JIRA uses in a database. It can be completely altered without changing any of the internal workings of JIRA.

The model provided should work with almost any database (care has been taken to ensure the column and table names are SQL compliant).

The entity model is configured through an XML file called `entitymodel.xml` (located in the distribution at `webapp/WEB-INF/classes/entitydefs/entitymodel.xml`). To edit this file, copy it to `edit-webapp/WEB-INF/classes/entitydefs/entitymodel.xml` and make changes there. When the WAR/EAR is built using `build.sh|bat`, the 'edit-webapp' version of the file will be used.

The format of the file should be fairly self explanatory - basically JIRA always refers to the `entity-name` and `field-name` attributes within the code. The `type` attribute of a `<field>` tag should always match the `type` attribute of a `<field-type-def>` tag in your `fieldtype-*.xml` files.

To change where entities and fields are persisted in your database, simply add (or edit) the attribute `table-name` (for entities) or `col-name` (for fields).

## 3. Why we chose the Entity Engine

We chose the EE over CMP or BMP entity beans because:

- it is more portable between application servers
- table schemas are automatically created and updated
- using the field type definitions, we can add support for new databases very quickly
- it is faster than most CMP implementations and has some nice caching features

This document deals with configuring the entity engine for JIRA (but should be applicable to most applications). For more details on the entity engine itself and it's inner workings, see:

### [OFBiz Entity Engine Guide](#)

describes the theory behind the entity engine, its architecture and usage patterns

### [OFBiz Entity Engine configuration guide](#)

describes all of the entity engine configuration options, whereas this document just describes configuring the entity engine for JIRA

### [API Docs](#)

the API docs for the `org.ofbiz.entity` package