

Installing JIRA on Tomcat 4.1.x

This section describes how to install JIRA on [Tomcat 4.1.x](#), a popular open-source server from the Apache Jakarta project.

JIRA 3 will work with Tomcat 4.1.27, 4.1.29 and above. It **will not fully work** with Tomcat 4.1.24 and versions earlier in the 4.1.x series (see [JIRA-4977](#)).

If you have not yet downloaded Tomcat, we recommend you bypass the Tomcat-JIRA config process altogether by downloading the [Standalone distribution](#), which comes preconfigured with Tomcat 5.5.9.

Tomcat can be downloaded the [Apache site](#), with earlier versions available from the [archives](#).

1. 1. Unpack JIRA

Unzip the JIRA WAR (Webapp ARchive) distribution. A new directory containing JIRA will be created, hereafter referred to as \$JIRA_HOME

2. 2. Configure JIRA

JIRA needs to be told what type of database you'll be using. The database is specified in \$JIRA_HOME/edit-webapp/WEB-INF/classes/entityengine.xml. Locate the **<datasource>** tag near the bottom, and change the **field-type-name** attribute value:

```
<datasource name="defaultDS"
  helper-class="org.ofbiz.core.entity.GenericHelperDAO"
  field-type-name="hsql"
  check-on-start="true"
  use-foreign-keys="false"
  use-foreign-key-indices="false"
  check-fks-on-start="false"
  check-fk-indices-on-start="false"
  add-missing-on-start="true">
  <jndi-jdbc jndi-server-name="default"
    jndi-name="java:comp/env/jdbc/JiraDS" />
</datasource>
```

Possible values include **cloudscape**, **db2**, **firebird**, **hsql**, **mckoidb**, **mysql**, **mssql**, **oracle**, **postgres**, **postgres72**, **sapdb**, and **sybase**

For PostgreSQL 7.3+ and DB2 you also need to set a **schema-name** attribute (see the [PostgreSQL](#) and [DB2](#) pages).

More details on JIRA's database access layer are available on the [EntityEngine configuration page](#).

3. 3. Build JIRA

Now build JIRA by typing **build** (Windows) or **./build.sh** (Unix) on the command line in the \$JIRA_HOME directory. This will produce the deployable WAR file in the \$JIRA_HOME/dist-tomcat directory.

Note:

The Tomcat developers, bless them, have bundled commons-logging.jar in common/lib/, causing [problems](#) for webapps like JIRA that also use commons-logging. For this reason, we generate a separate webapp for Tomcat (in the **dist-tomcat/** directory) that does not contain commons-logging.jar. Please be sure to deploy the correct webapp.

Warning:

If you want to copy the WAR file somewhere else, be sure to customise the path to it in **jira.xml** below. **Do not** copy this WAR file to Tomcat's `webapps/` (or `server/webapps/`) directory, as it would be auto-deployed there in an unconfigured state.

Note:

Following an upgrade, some users have encountered exceptions regarding velocity templates that cannot be found. Comments on issues are also not available - instead the warning "**Velocity template generation failed**" is displayed. A workaround to this issue is to manually extract the `.war` file.

4. 4. Update Tomcat Libraries

Tomcat does not come with some libraries required to run JIRA. To fix this, download [jira-jars-tomcat.zip](#) (1.2Mb), and copy the contained jars to Tomcat's `common/lib/` directory.

If using Tomcat 4.1.27 or earlier: JIRA requires commons-logging and commons-pool 1.1 or above. These are included in the [extra jars package](#). Tomcat 4.1.27 and earlier provide old versions of these libraries, which **you will need to remove**.

Warning:

Since Jira 3.1, Jira has been using a newer version of the `javax.mail` and `javax.activation` packages than are shipped with Tomcat 4.1.x. If you run Jira with the shipped jars in place you will encounter a known bug described [here](#) when trying to send email. The solution to the problem is to remove the `mail.jar` and `activation.jar` from the `TOMCAT_HOME/common/lib` directory and replace it with the jars that Jira ships with. Full instructions and details can be found [here](#).

5. 5. Configure Tomcat

A JIRA 'context' now needs to be set up in Tomcat. To do this:

1. Copy **dist-tomcat/tomcat-4/jira.xml** from the built JIRA distribution to your Tomcat's `webapps/` directory.
2. Customize the copied **jira.xml** as follows:

```
<Context path="/jira" docBase="path/to/atlassian-jira-3.6.war" debug="0">
  <Logger className="org.apache.catalina.logger.FileLogger"
    prefix="atlassian-jira." suffix=".log" timestamp="true"/>
  <Resource name="jdbc/JiraDS" auth="Container" type="javax.sql.DataSource"/>
  <ResourceParams name="jdbc/JiraDS">
    <parameter>
      <name>driverClassName</name>
      <value>org.hsqldb.jdbcDriver</value>
    </parameter>
    <parameter>
      <name>url</name>
      <value>jdbc:hsqldb:path/to/jira_database</value>
    </parameter>
    <parameter>
      <name>username</name>
      <value>sa</value>
    </parameter>
    <parameter>
      <name>password</name>
      <value></value>
    </parameter>

    <!-- NOTE: If NOT using hsqldb, delete the next two parameters -->
    <!-- Give unused connections 4 secs before eviction. -->
    <parameter>
      <name>minEvictableIdleTimeMillis</name>
      <value>4000</value>
    </parameter>
    <!-- Check for evictions every 5 secs. -->
    <parameter>
      <name>timeBetweenEvictionRunsMillis</name>
```

```
<value>5000</value>
</parameter>
<parameter>
  <name>factory</name>
  <value>org.apache.commons.dbcp.BasicDataSourceFactory</value>
</parameter>
</ResourceParams>

<Resource name="UserTransaction" auth="Container" type="javax.transaction.UserTransaction" />
<ResourceParams name="UserTransaction">
  <parameter>
    <name>factory</name>
    <value>org.objectweb.jotm.UserTransactionFactory</value>
  </parameter>
  <parameter>
    <name>jotm.timeout</name>
    <value>60</value>
  </parameter>
</ResourceParams>
</Context>
```

The paths (denoted as **path/to/**) will be correct by default, assuming you want to deploy the .war from the **dist-tomcat/** directory.

Note:

If you are *not* using hsqldb, make sure you comment out the `minEvictableIdleTimeMillis` and `timeBetweenEvictionRunsMillis` params, or JIRA will run slower than normal.

If you are installing in Windows, make sure that the paths you specify for the location of the WAR file and database are full paths with drive letters (e.g. `c:\yourdb\tomcatdb`). **N.B.** the last part of the path is the name of the database and is not a directory.

Note:

Currently JIRA's HTTP sessions may contain objects which cannot be serialized to disk. Tomcat tries to serialize existing sessions by default during shutdown. You can add the following configuration parameter between the `<Context>` tags to disable this:

```
<Manager className="org.apache.catalina.session.PersistentManager" saveOnRestart="false" />
```

Note:

So that user and group names in JIRA can contain internationalized characters the property `useBodyEncodingForURI` within the connector definition for your http protocol must be set to true. This is the default value for tomcat 4, so you do not need to modify any values, this is not the case for tomcat 5.0.x and 5.5.x. Please note that there is a bug in tomcat versions prior to 4.1.31 that ignores this value. If you want to use i18n characters in JIRA running in tomcat 4 please make sure you are using version 4.1.31 or higher.

5.1. Configure the database connection

The example context shown above uses the [HSQL](#) in-memory database. To use a different database, copy its JDBC driver jar to `common/lib/`, and change the context XML appropriately (see the [database doc](#)).

6. Start Tomcat

JIRA should now be ready to run in Tomcat. To start using JIRA, first start (or restart) the Tomcat server with Tomcat's `bin/startup.(sh|bat)` scripts, and point your browser to <http://localhost:8080/jira>

You should now see the [Setup Wizard](#), which will take you through the brief setup procedure.

7. Troubleshooting

It is easy to make a mistake in this process, and even more so if you are trying to connect to a

database other than hsqldb. First, check that you have followed the process described above:

- If you are using an external database (not hsqldb), have you set the **field-type-name** attribute in `$JIRA_HOME/edit-webapp/WEB-INF/classes/entityengine.xml`? ([step 1](#))
- Have you previously started JIRA with an incorrect **field-type-name** value? If so, the database schema would have been created incorrectly.
- If you have made changes to `$JIRA_HOME/edit-webapp/WEB-INF/classes/entityengine.xml` ([step 2](#)) and re-run the build script ([step 3](#)), but your changes are not being picked up, delete the Tomcat `webapps/jira` directory, then restart JIRA. It would seem that in some circumstances Tomcat does not correctly re-expand the web application.
- Have you copied the extra Tomcat jars ([step 4](#))? Check if you have `common/lib/objectweb-datasource-1.4.3.jar` present.
- If using an external database, did you copy the JDBC driver jar to `common/lib/` ([step 5](#))?
- Is the path to the `.war` file in `webapps/jira.xml` correct?
- Have you copied the `.war` file to Tomcat's `webapps/` directory? This is almost guaranteed to cause pain - please move it elsewhere, and delete any JIRA subdirectories created in `webapps/` from previous Tomcat starts.
- Have you configured JIRA centrally in `conf/server.xml` instead of in `webapps/jira.xml`? This is fine, but then be sure you *don't* also have a `webapps/jira.xml` present.
- The log files are usually vital to debugging problems. On Windows, these will appear in the console window that loads when running `startup.bat`, or in one of the log files in the `logs/` directory. On Linux/Unix, logs will appear in a log file in `logs/`, usually `logs/catalina.out`. Check the log file for errors after startup.

If you're stuck, please raise a [support request](#), and attach your logs, configuration files, plus anything else relevant, and we'll get back to you as soon as possible. If you have a general question, try the [jira-user mailing list](#) (which Atlassian staff monitor).