STARTER **GUIDE 4.2**

INTRODUCTION

This document is for those who wishes installing Entando v4.2.0.

TARGET AUDIENCE

This document is intended for both developers and experts in graphical interfaces for the Web.

Table of Contents

SOFTWARE **PREREQUISITES**

The Entando Platform requires the following software to be installed and operational. Unless otherwise specified, the version must be strictly observed.

The installation of the software might be more or less automated, assisted by an automated installer or not, depending on software and the target operating system: for this reason this guide does not make any assumption about the current operating system or about the presence of an automated installer. We rather suggest the final checks to be made to ensure everything is up and running.

Note for Linux users:
though this document shows the instructions to install different software, packages of your distribution will suit your needs, provided that you have an updated distribution.

BASIC **PREREQUISITES**

- JDK Oracle 1.7 + (Oracle website)
- Maven 3+ (Maven website)
- Ant 1.8.x + (Ant website)

Important:
Entando runs happily with all the mainstream databases, but is currently distributed for Derby, but you can switch to PostgreSQL, MySQL, SQLServer and to any other DBMS (also NoSQL as MongoDB).
The incoming release will greatly improve the installation of the platform with any DBMS of choice, through the smart use of bundles.

Note:
the scripts which restore the database use the user agile (with password agile) to establish a connection with the DBMS. After the installation of the software above is completed the JAVA_HOME, M2 and ANT_HOME must be included in the PATH environment variable. With this minimal setup evaluators who already know how Maven works are able to instantiate a new Entando project and to run it using the Jetty web server.

DEVELOPMENT **PREREQUISITES**

To develop new application services in Entando an IDE is obviously needed.
Entando is agnostic with respect to the IDE and to the servlet container, so developers don't have to change their tools of choice. However we refer to the Eclipse and NetBeans IDE and Apache Tomcat.
The goal is to make the IDE able to handle Maven projects and to deploy Entando based projects in an external instance of Tomcat (we strongly advice you against using built-in servers).

Warning:
the steps needed in order to configure Entando are beyond the scope of this chapter and are not shown here.

DATABASE

Sooner or later, developers will want to switch from the default Derby to another DBMS; for the purpose of the current document, PostgreSQL or MySQL will be sufficient, though Entando runs happily with all mainstream DBMSs.

- PostgreSQL >= 8.4+ can be downloaded from the PostgreSQL website
- MySQL Community Server >= 5.6+ (Community edition download page)
- Choose the correct driver for Postgresql or MySql.

SERVLET CONTAINER

Entando is agnostic with respect to the servlet container; any servlet container is supported as Jetty, Apache Tomcat (v6, v7, v8), JBOSS v5, Oracle WebLogic.

Regarding Tomcat:
Tomcat can be downloaded from the Apache Tomcat website.
Please make sure that the CATALINA_HOME system variable points to your local Tomcat installation directory, otherwise create it.
Open the web.xml in the Tomcat installation directory and add the following snippet within the <servlet> tag:

```
<init-param>
        <param-name>trimSpaces</param-name>
        <param-value>true</param-value>
        </init-param>
```

Optional:
edit the .../conf/tomcat-users.xml in the installation directory and add one administrator role for the server management.
Restart the server to update the configuration.
Remember to place the correct JDBC driver within the lib folder under Tomcat.

ECLIPSE IDE

The latest version of the Eclipse IDE for Java EE Developers can be downloaded here.

Setting the eclipse.ini

To prevent memory shortage problems the eclipse.ini located in the installation directory can be modified (or updated) to include the following lines:

```
-XX:MaxPermSize=512m
-Xms512m
-Xmx1024m
```

These values can be changed to fit the reader’s needs.

Installing Maven plugins in Eclipse

3.x versions

Developers must install two additional plugins using the Eclipse Marketplace Client: installing plugins from the marketplace is the preferred method; a valid alternative is to use software sites valid at the time of writing and these instructions:

- **m2e** version >= 1.2.0 (info site)
- **m2e-wtp** >= 0.16.0 (info site)

Warning:
older versions of these plugins are known to have issues: only the latest versions fixed a severe bug which prevented a Maven project to be deployed correctly on the Tomcat instance.

4.2 (Juno)

- install the plugin Maven Integration for Eclipse WTP (Juno) from the Eclipse marketplace

4.3 (and newer) version

- The Java EE version is shipped with the Maven support, nothing to do!

When the installation process ends, choose to restart Eclipse.

At this point the project *Portalexample* from the Entando GitHub page can be downloaded and imported into the workspace: the icon of the project should contain a small “M” which distinguishes Maven projects from the others.

Configuring server options

Taking as reference the Eclipse Indigo (but newer versions should retain the same menu voices) *Window » Preferences then choose Server » Runtime Environment*.

Click on the **Add** button then choose the servlet container among those available. Let's choose Apache 6.0 or 7.0 then check off the **Create a new local server** box.

Press **Next**.

From this window click the **Browse**... button to select the directory where Tomcat was installed.

Press **Finish** to end the configuration of the server.

Configuring Maven options

From the Preferences window Maven » Installation. Again, we are not going to use the embedded 3.x Maven. Clicking on the **Add**... button then select the directory where Maven is installed.

Click on the **Apply** button.

Configuring Workspace options

From the *Preferences window select General » Workspace*. In the Text File encoding select the UTF-8 encoding.

Then from General » Content Types: in the Content Types window expand the Text menu: we are going to change the default encoding of the HTML, *Java Properties files*, *Java source files*, *Javascript*, *JSP* and *XML* groups.

All of these groups, with the sole exception of the Java Properties Files which explicitly require ISO- 8859-1, must be set to UTF-8 to live long and prosper.

The content type for each group (or individual file) is set in the **Default encoding** input field at the bottom of the window.

Final considerations for the Eclipse IDE

Even if the new release of the m2e* plugins works a lot better than the previous, developers might experience issues when deploying or running the project: inconsistent errors, partial deployments etc.

Usually it is necessary to clean the server work directory and refreshing the entire project after a fresh compile of the project.

Please also remember that the **procedure shown above is necessary but not sufficient to run an Entando project under Eclipse since we are only installing and configuring the IDE, not Entando itself**.

NETBEANS IDE

Netbeans IDE requires a lot less configuration work to be done and runs smoothly once configured.
Again we have to tell the IDE not to use the embedded Maven or Tomcat; we also specify the Ant installation directory.

Any version >= 7.1 can be download from this page.

Configuring Ant

Access to configuration windows from the Tools » Options menu.

On opening, click the **Miscellaneous** (Netbeans 7.1) or the **Java** (Netbeans 7.2) view:
select the tab Ant then click the **Browse Button** next to the Ant Home input field: point to your local installation directory of Ant.

The Option windows can be closed clicking on the **Ok** button.

Configuring Maven

From the same window opened previously, select the tab **Maven**.

In the Maven Home drop down menu choose browse... and select the Maven local installation directory.

Check off the box **Skip Tests for any build executions not directly related to test** to avoid executing tests unnecessarily.

The Option windows can be closed clicking on the **Ok** button.

Configuring Tomcat

Access the Services panel from the Window » Services or by pressing Ctrl-5.

Right-click on the Servers » Add Server.

From the Add Server Instance window select **Apache Tomcat** from the Server list and click **Next**.

Click on **Browse**... then select the directory where Tomcat is installed, and enter the Username and the Password of the Tomcat manager in the respective input boxes.

Click the **Finish** button.

From the Services panel right-click on the newly created Apache *Tomcat » Properties*: on the Server window click on the tab *Platform* and enter the following arguments in the VM Option input box:

```
-Xms512m -Xmx102 -XX:MaxPermSize=512m -XX:-DoEscapeAnalysis
```

Again, these values can be changed to fit the reader's needs.

HARDWARE PREREQUISITES FOR DEVELOPMENT ENVIRONMENT

An Entando portal by itself is neither CPU intensive nor memory demanding; however, depending on the project configuration, the development environment software might become aggressive on the CPU and memory.

For example, installing the Jaspersoft and the Forum plugin is known to be memory and CPU demanding.

If a project requires a plugin which interacts with external servers like Pentaho, Jaspersoft, SugarCRM etc., we strongly advise developers against installing those servers in the same machine where the project is developed, especially on older hardware.

The following hardware configuration is suitable for a wide range of projects:

- 4 GB of RAM
- Dual core CPU ~ 2.2 Ghz
- ~ 1Gb of free space on disk (the vast majority of the disk space is for Maven dependencies!)

It is worth noting that performances vary depending on the operating system used and, for Linux systems, also by the distribution (for example, Ubuntu is more resource demanding than Xubuntu).

Also, the IDE used greatly influences system performance: usually Netbeans requires fewer resources than Eclipse, even if Eclipse performs better with large projects.

Please note that **we are not endorsing an IDE over another**: as stated before Entando is IDE agnostic and the final decision must be made by developers.

SYSTEM **VARIABLES**

System variables are used by software scripts to evaluate the actual installation directory of a certain software element. Since we do not assume the presence of an installer we are going to show briefly how to create a system variable.

LINUX

In Linux, a global environment variable is always created with a command similar to the following:

```
export ANT_HOME=/home/entando/opt/apache-ant-1.8.4
```

This command alone is not persistent because on shutdown it will be lost; to make it permanent on the majority of Linux distributions it is sufficient to include this command at the end of the local user profile script file, namely ~/.bashrc or ~/.profile depending on the distribution of choice.

Other distributions might have different profile scripts, so please refer to the documentation accompanying the distribution.

To modify the PATH system variable, it is sufficient to do the following:

```
# define shortcuts to the executable directories
export ANT=$ANT_HOME/bin
export CATALINA=$CATALINA_HOME/bin
# finally modify the PATH variable, keeping its old value
export PATH=$PATH:$JAVA_HOME:$M2:$CATALINA:$ANT
```

WINDOWS

Though this example is aimed at Windows 7, the same concepts apply for older versions.

Open the System window from the Control Panel » System.

Click on the tab Advanced system settings in the left pane (Windows 7 only)

On the System Properties windows click on the Advanced tab, then on the **Environment Variables**... button.
In the Environment Variable window add the new variables in the System Variables section.

Click the **New** button (or the **Edit**... button) to create (or add) a system variable: in the New System Variable window insert the name and the value (the installation directory path of the program associated to theNvariable)
To modify the PATH system variable you only have to follow these steps:

- locate the **Path** variable in the System variables section of the Environment Variables window.
- edit it and append the following string (please note the leading semicolons!)

```
;%M2%;%JAVA_HOME%;%CATALINA_HOME%\bin;
```

ANT **TASKS**

Ant carries out a number of automated tasks to make the life of system administrators and developers easier; we are not going to show them all, but only those relevant to developers and system administrators in everyday life.
To list them, open a terminal in the root directory of the project and execute the following command.

- ant -p

Ant shows all the targets (tasks) available

```
entando@gemini /tmp/myNewPortal
$ ant -p

Buildfile: /tmp/myNewPortal/build.xml
myNewPortal - Build file
Main targets:

MySQL-db-backup          MySQL db backup
MySQL-db-create          MySQL db create
MySQL-db-drop            MySQL db drop
MySQL-db-full-update     MySQL db update
MySQL-db-restore         MySQL db restore
PG-db-backup-tar         PostgreSQL db backup in TAR format
PG-db-full-update-tar    PostgreSQL db update from tar file
Site                     Generate the Site
Test-MySQL-db-init       Use it to restore databases for tests - MySQL
Test-PG-db-backup        Test - PG db backup in TAR format
Test-PG-db-export-SQL    Test - PG db backup in plain SQL
Test-PG-db-init          Use it to restore databases for tests - PostgreSQL
WAR-build                Build the WAR
WAR-build-standalone     Build the standalone executable WAR
WAR-deploy               Copy the WAR to Tomcat
clean                    Clean the project
compile                  Compile the source
init                     Initialize the project
Default target: WAR-build
```

The most useful tasks by far are those used for database management, like MYSQL-db-* and PG-db-*

For example:

- **ant PG-db-backup-tar** will backup and bzip the tables in ../myNewPortal/src/main/db/tar (PostgreSQL DBMS)
- **ant MySQL-db-backup** will dump tables in SQL format in ../myNewPortal/src/main/db/mysql/ (MySQL DBMS)
- **ant PG-db-full-update-tar** restores the bzip backup in ../myNewPortal/src/main/db/tar (PostgreSQL)
- **ant MySQL-db-full-update** restores the backup in ./myNewPortal/src/main/db/mysql/ (MySQL)
- **ant MySQL-db-create** (MySQL) **ant PG-db-create** (PostgreSQL) will create the databases in the target DBMS
- **ant MySQL-db-drop** and **PG-db-drop** will drop tables (warning: system won't ask for confirmation!)

Since the 3.2 release Entando added **back-office functionality to backup the components of the system: the commands to dump and restore tables are now only meaningful to system administrators and developers should not use these commands anymore**.

- **ant WAR-build** will generate the WAR of the portal once the production filter has been prepared (the description of the filters is outside the scope of this document and is not treated here). It is worth noting that this task invokes Maven transparently with a number of parameters that would otherwise be difficult for developers to remember.

USEFUL MAVEN COMMANDS

Maven has a number of commands which come in handy during development.

- **mvn clean** cleans the 'target' directory (when present) where Maven assembles the web application. Derby DBMS places its tables inside the 'target' directory, so this command prevents the accidental deletion of these files.
- **mvn compile** compiles the project; if the IDE reports compilation errors but this command from terminal is completed successfully, then the project in the IDE needs to be cleaned and rebuilt.
- **mvn dependency:tree** creates a tree with all the dependencies of the project. Very useful to detect conflicting libraries or just to find exact versions of jar
- **mvn clean jetty:run run** the project using the Jetty web server Entando-

GETTING **STARTED**

In order to start Entando v4.2.0, you have to install:

- Java JDK version >= 7 (http://www.oracle.com/technetwork/java/javase/downloads/index.html)
- Maven version >= 2.2.1 (http://maven.apache.org/download.cgi) (It's recommended to use Maven up to version v3.0.5)

And this if you are going to develop:

- An t (http://ant.apache.org/bindownload.cgi)

With the JDK, Maven and Ant installed you are ready to go. Some detail on how to archieve that below.
Attention:
If you are using JDK = 8, you have to uncomment the following script in the pom.xml file of the plugins folder within the entando-components project and in the pom.xml file of the entando-core project:

```
<executions>
  <execution>
          <id>attach-javadocs</id>
          <goals>
                  <goal>jar</goal>
          </goals>
  <!-- uncomment this configuration if you want to use jdk 1.8 -->
          <!--
          <configuration>
                  <additionalparam>-Xdoclint:none</additionalparam>
          </configuration>
          -->
  </execution>
</executions>
```

Check your Installation

- Create an empty directory and generate a test web application, based on the Entando bootstrap archetype, typing bogus data when asked:

$ mvn archetype:generate -Dfilter=entando-archetype-portal-bootstrap

Note:
Every Maven project has its groupId: this is generally unique amongst an organization or a project; artifactId: this is generally the name that the project is known by; version: this is the version of the project. Maven will ask these to you in three steps. Finally, Maven will ask you the package structure for your code.

- Enter the newly created folder
- Launch Jetty

$ mvn clean jetty:run

- Open your browser at http://localhost:8080/ (ignore the 404 error)
- Click on the link you will find on that page

SETUP ENVIRONMENT WITH OS X 10.9 OR NEWER

- Download the Java JDK 7 from the oracle website (http://www.oracle.com/technetwork/java/javase/downloads/index.html) and install the provided package.
- Set the JAVA_HOME environment variable:

```
$ echo "export JAVA_HOME=\`/usr/libexec/java_home\`" | tee -a ~/.bash_profile
```

- Install XCode from the App Store
- Install Homebrew (http://brew.sh/)
- Run brew doctor and fix any warning you get from it
- Install Maven and Ant:

```
$ brew install maven
$ brew install ant
```

SETUP ENVIRONMENT WITH UBUNTU

- Install Maven and Ant:

```
$ sudo apt-get install maven ant
```

- Set the JAVA_HOME environment variable:

```
$ echo "JAVA_HOME=\"/usr/lib/jvm/default-java\"" | sudo tee -a /etc/environment
```

- Reboot

SETUP ENVIRONMENT WITH WINDOWS

- Install Java JDK 7 (http://www.oracle.com/technetwork/java/javase/downloads/jdk7- downloads-1880260.html)

  - Download and install the .exe installer

```
jdk-7u79-windows-x64
```

  - Create the environment variable

```
JAVA_HOME -> C:\Program Files\Java\jdk1.7.0_79
```

- Install Maven (http://maven.apache.org/download.cgi)

```
apache-maven-3.3.3-bin
```

  - Set the Maven environment variable

```
MVN_HOME -> D:\programmiInstallati\apache-maven-3.3.3
```

- Install Ant (http://ant.apache.org/bindownload.cgi) apache-ant-1.9.6-bin

  - Set the Maven environment variable

```
ANT_HOME -> path: D:\programmiInstallati\apache-ant-1.9.6
```

• Update the environment variable “PATH” in order to execute the programs by shell:

Example:

```
D:\programmiInstallati\apache-Maven-3.3.3\bin;%JAVA_HOME%\bin;%ANT_HOME%\
bin;%SYSTEMROOT%;
%SYSTEMROOT%\System32;
```

DOWNLOAD THE **LATEST SOURCE CODE**

To download the latest source code:

• Open your terminal
• Create an empty directory for your project

```
mkdir ~/my_new_project_portal
```

• Move to a directory of your choosing

```
cd ~/my_new_project_portal
```

• clone in the following sequence entando-core, entando-components, entandoarchetypes, entando-ux-packages projects:

```
git clone https://github.com/entando/entando-core
git clone https://github.com/entando/entando-components
git clone https://github.com/entando/entando-archetypes
```

• if you want to join with ready samples of Entando based applications, you can to use entandoux- packages:

```
git clone https://github.com/entando/entando-ux-packages
```

• Install, in the following sequence, the entando-core, entando-components, entandoarchetypes projects:

```
cd entando-core
mvn clean install -DskipTests
```

```
cd entando-components
mvn clean install -DskipTests
```

```
cd entando-archetypes
mvn clean install -DskipTests
```

• Create an empty directory and generate a test web application, for instance based on the Entando bootstrap archetype, and type bogus data when asked:

```
$ mvn archetype:generate -Dfilter=entando-archetype-portal-bootstrap
```

otherwise, if you want to use the ready samples applications (e.g. portalexample), you can run **entando ux-packages**:

```
cd entando-ux-packages
cd portalexample
mvn clean jetty:run
```

Finally:

- Open your browser at http://localhost:8080/ (ignore the 404 error)
- Click on the link you will find on that page

NOTE:
If you want to deepen how to use the Entando archetypes, visit the wiki of entandoarchetypes project.

Authors
Entando Dev Team <info@entando.com>
Second Public Release

V 2.0 – April 2016

entando

www.entando.com