Setup a Java development environment for the Entando Platform

Software needed to run and develop the Entando Platform

Authors

Matteo Minnai Entando s.r.l. <m.minnail@entando.com> First draft

V0.9 Jan 2012

Legal Notice

Introduction

The purpose of this document is to provide a complete list of the software prerequisites needed to setup the proper environment to run and develop for the Entando Platform. This document does not show how to configure Entando.

Target Audience

This document is intended for **Java developers** who wish to setup a development environment or run an instance of the Entando Platform; we take as reference two opensource mainstream IDE, **Eclipse** and **Netbeans**, but the same concepts apply to others IDEs too.

As a general concept developers are not required to change the development tools of choice.

Web designers might still find the information here contained useful.

Prerequisites

No particular skills are required in order to properly setup an environment required by Entando

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 where to download and the instruction to install the various software, packages of your distribution will suit your needs, provided that you have an updated distribution.

Basic prerequisites

In order to successfully compile Entando you need

- **JDK** >= 1.6 Version 1.6 u34 can be downloaded from the Oracle website.
  Please make sure that the system variable JAVA_HOME is defined and point to the JDK base directory otherwise create it: consult the 3rd chapter for further information on how to create system variables.
- **Maven** 2.2.1 This version is mandatory, even if Maven 3.x aims to be backward compatible with 2.x.
  Maven binaries can be downloaded from the Maven website; installation instructions for both Windows and Linux are available in the same page.
  Please make sure that the system variables M2_HOME and M2 point respectively to the Maven installation directory and to the bin directory within it, otherwise create them both.

  Note for Windows users: even if the instructions target older Windows versions, the same concepts apply to both Windows Vista and 7.
- **Ant** >= 1.8.x It can be downloaded from the Ant website. A nice but optional addition to ANT is the **ant-contrib** package, which can be download from here. This package helps with the installation of Entando plugins.
  Please make sure that the system variable ANT_HOME points to the Ant installation directory, otherwise create it.
- Though Entando supports many Database Management Systems, for the purpose of the current document we have the following options:
  1. **PostgreSQL** >= 8.3.x. It can be downloaded form the PostgreSQL website.
  2. **MySQL Community Server** >= 5.x (Community edition download page)

  Important: the Entando Platform runs happily with all the mainstream databases; but it is *currently*

distributed with the scripts for PostgreSQL and MySQL only. The incoming release will greatly improve the installation of the platform with any DBMS of choice, through the smart use of Maven 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. Consult the 3rd chapter for information.

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 just following the simple guide in the Entando's documentation page on GitHub.

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, for the purpose of this document 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 document and are not shown here.

Tomcat

Tomcat can be downloaded from the Apache Tomcat website, version 6.x (install instructions) or 7.x (install instructions) are both fine.

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. Choose the correct driver for Postgresql or MySql.

Eclipse IDE

Eclipse IDE for Java EE Developers can be found here. Please note that as of November 2012, the latest release, Juno (4.2), is affected by performance issues which will be fixed only with the next release.

The reader might consider to use the Eclipse Indigo (3.7) version here.

Setting the eclipse.ini

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

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

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

Installing Maven plugins

Developers must install two additional plugins from the Eclipse Market**:**

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

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 different 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 were Tomcat was installed.

Press **Finish** to end 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 solely 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 an configuring the IDE, not Entando itself.

Netbeans IDE

The 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 test unnecessarily.

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

Configuring Tomcat

Access the Services pane from the *Window* → *Services* or by pressing *Ctrl-5.*

Right-click on the *Servers* → *Add Server.*

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

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

Click the **Finish** button.

From the Services pane 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 -Xmx1024m -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 Jasper and the Forum plugin are known to be memory and CPU demanding.

If a project requires a plugin which interacts with external servers like Pentaho, Jasper, 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 used (for example, Ubuntu is more resource demanding than Xubuntu).

Also the IDE used greatly influences the system performance: usually Netbeans requires less 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, please refer to the documentation accompanying the distribution.

To modify the *PATH* system variable is it 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 the variable)

To modify the *PATH* system variable is it sufficient 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

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