INSTALLING ENTANDO APP BUILDER 5.0

1. PREREQUISITES

1.1. SPACE AND HARDWARE REQUIREMENTS

Entando is a built to be a very light framework requiring limited runtime resources. When extending Entando in a development setting, sizing for development environments that add to an Entando install is up to individual user requirements.

1.2. INSTALLED PREREQUISITES:

- Java 1.8+
- npm 5.6.0+ (for ReactJS front ends)
- Maven 3.0.5+ (including Maven Central)
- Ant 1.8.0+
- Git (only if you intend to build from source)

2. CREATING A NEW ENTANDO PROJECT

Set up an Entando application via a Maven archetype. The archetype creates the project and structure needed to develop and add to an application.

Steps:

1. Open your command line and navigate to the directory that will house your new project.
2. Run the following command:
   ```
   mvn archetype:generate -Dfilter=entando-archetype-webapp-generic
   ```
3. Select Option 1, displayed as follows:
   ```
   1: remote > org.entando.entando:entando-archetype-webapp-generic (Generic web app Archetype for Entando: an agile, modern and user-centric open source web app like platform.)
   ```
4. If prompted for a version, select 5.0.0.
5. Enter values for groupId, artifactId, version, and package. These values go into the Maven setup in the new project. To learn more about each value, see the Maven documentation here: https://maven.apache.org/guides/mini/guide-naming-conventions.html
6. Open the directory created by the Maven archetype in Step 2. The directory has the same name as the value you entered for artifactId in Step 5. This directory contains all of the structure necessary to start the Entando MApp Engine and execute the application. See Launching the MApp Engine section below for instructions on starting the newly created project.

3. LAUNCHING THE MAPP ENGINE

3.1. QUICK START USING DERBY AND JETTY

Once you have created an Entando Application, you can launch it using Jetty and an embedded Derby database.

To do so, run the following command inside your project:
`mvn clean jetty:run`

Once started the logs will print the following messages:
[INFO] Started SelectChannelConnector@0.0.0.0:8080
[INFO] Started Jetty Server
[INFO] Starting scanner at interval of 5 seconds

The logs will indicate that your application is running. Follow the steps in Launching the App Builder section to launch the App Builder.

3.2. BUILD THE MAPP ENGINE FROM SOURCE

Building the MApp Engine from source is for special cases such as applying hot fixes. The Quick Start method is sufficient for most uses.

To download the latest source code:

1. Open your terminal and create an empty directory for your project:
   `mkdir ~/my_new_project`
2. Move to the new directory
   `cd ~/my_new_project`
3. Clone the following repositories IN ORDER: entando-core, entando-components, entando-archetypes, entando-ux-packages projects:
   a. Entando-core:
      `git clone https://github.com/entando/entando-core`
   b. Entando-components:
      `git clone https://github.com/entando/entando-components`
   c. Entando-archetypes:
      `git clone https://github.com/entando/entando-archetypes`
4. Install, IN ORDER, the entando-core, entando-components, entando-archetypes projects:

   NOTE Each of the following steps takes several minutes to complete.

   a. Move to the entando-core directory and enter:
      `mvn clean install -DskipTests`
   b. Move to the entando-components directory and enter:
      `mvn clean install -DskipTests`
   c. Move to the entando-archetypes directory and enter:
      `mvn clean install -DskipTests`

5. Complete the download by following the steps from the Creating a New Entando Project section.

NOTE The command to use the artifacts you have installed locally with an additional switch on the archetype command is:
`mvn archetype:generate -Dfilter=entando-archetype-webapp-generic -DarchetypeCatalog=local`

4. SETTING UP A DATABASE (OPTIONAL)

Entando comes with a Derby database that handles small scale database functions. However, for more robust functionality such as running queries or creating a shared instance, you can configure a newly created Entando application to connect to a database as its backing store. You can change the default from Derby to your new database choice by opening the Filter Properties file in src/main/filters and entering the appropriate environment and database configuration.

To connect the MApp Engine to a database server:

1. In your database server, create a user for the application.
2. Create two databases. One database will hold the design, the other will hold the server instance. Give the user from Step 1 permission to create, read, and write. The default database names are:
   `<your-project-name>Port`
   `<your-project-name>Serv`

   NOTE These are the default names included in the default properties files. You can name the databases anything you like and can change to connect to your custom names in the configuration filters in src/main/filters inside the project.

3. Uncomment the maven dependency for the database that you are using from the pom.xml file. For example for MySQL you would uncomment:

```
<dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>5.1.18</version>
</dependency>
```

4. Update the appropriate Filter Properties file in src/main/filters to use the configuration for the database properties. For example, on a macOS, you would update *filter-development-unix.properties*.
5. Set the user, database, and password for the values created in Steps 1 and 2.
6. Launch the application:
   `mvn jetty:run`

NOTE When launching with the mvn jetty:run command, Jetty Entando will automatically create the table structure required to run the application. This can be used to instantiate an empty database when the target deployment is an app server such as JBoss or Tomcat.

5. LAUNCHING THE APP BUILDER

This section provides several options for launching the App Builder which is used to manage a separately running Mapp Engine instance. The App Builder is completely stateless and relies on the Engine to store the application configuration. If you have not specified a Mapp Engine, the App Builder will default to providing mock data so that it can demonstrate the functionality.

5.1. BUILD FROM SOURCE

Prerequisites:

- Git
- npm
- node

5.1.1. CLONE AND SET UP

Enter the following commands in your command line to start the application in dev mode:

1. `git clone https://github.com/entando/app-builder.git`
2. `cd app-builder`
3. `npm install`
4. `npm start`

NOTE The npm install command installs npm dependencies inside the node_modules folder.

This will give you a running App Builder instance using mock data. The configuration section below covers how to connect your App Builder to a running MApp Engine.

5.1.2. CONFIGURATION

The application uses .env files to set up some environment variables.

Dev instances should use the .env.development.local file while production instances use .env.production.

Configurable properties

- USE_MOCKS (boolean, default: true): a boolean used to determine whether the API calls will be against a real Entando Core or if they are just being mocked internally.

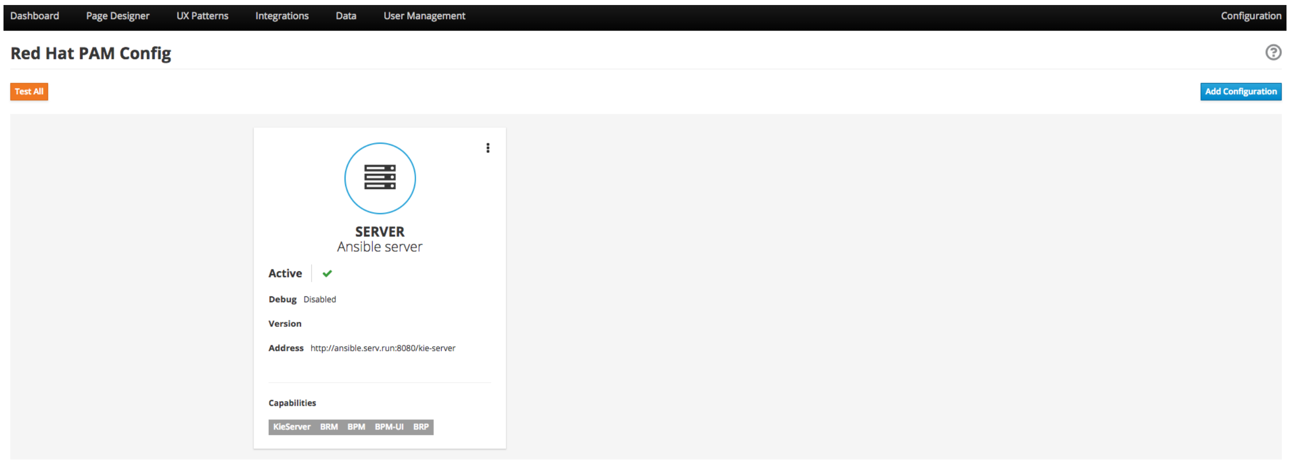
- DOMAIN (string, default: null): a string representing the domain name of the Entando Core instance. The protocol is optional and it is possible to specify a subdirectory of the domain. Trailing slashes are not valid and it only validates up to 3rd level domains.

To connect the MApp Engine:

1. Create a file at the root of the App Builder called .env.development.local.
2. Add
   `USE_MOCKS=false`
3. Add
   `DOMAIN=<url of running MApp Engine>`
4. Run `npm start` in your App Builder directory.

5.1.3. INTEGRATE THE BPM FORM

1. From the tabs along the top, choose Integrations > Red Hat BPM Config > Add Integration.



Red Hat PAM Config

SERVER
Ansible server

Active ✓

Debug Disabled

Version

Address http://ansible.serv.run:8080/kie-server

Capabilities

KieServer BRM BPM BPM-UI BRP

2. In the Connection section, add all server information. For example:
   - General Settings section:
     - **Active:** On
     - (Optional) **Debug:** On
   - Connection section:
     - **Name:** Ansible Server
     - **Host name:** ansible.serv.run
     - **Schema:** http
     - **Port:** 8080
     - **Webapp name:** kie-server
     - **Username:** username for the process server administrator
     - **Password:** password for the process server administrator
     - **Conn. Timeout (in millis):** 500
3. Click **Test Connection** to verify a valid the configuration, then click **Save**.

4. Open the Page Tree by choosing the Page Designer tab > Page Tree.
5. Find your new page in the Page Tree and click its **More Options** button > Configure.
6. Search for the “BPM-form by Data Type” widget and drag it to the Central Bar Center frame.
7. Click the **Preview** button to verify the correct appearance.
8. Click **Publish**.

5.1.4. ADDITIONAL COMMANDS

The following are other, optional commands you can use to perform certain tasks. You must run these commands in the app_builder directory:

1. `npm run lint`
   Runs the linter. It fails if linting rules are not matched.
2. `npm run coverage`
   Runs unit tests. It fails if a unit test fails, or if the minimum coverage threshold is not met.
3. `npm run import-plugins`
   Compiles and imports Entando plugins.
4. `npm run build`
   Compiles the project and creates the build directory.
5. `npm run build-full`
   Runs npm run lint, npm run coverage, npm run import-plugins and npm run build.

5.2. USING DOCKER

5.2.1. PREREQUISITES:

- An installed and running Docker instance
- S2I command installed (https://github.com/openshift/source-to-image)
- An Entando App created using the steps in Creating a New Entando Project that can be run directly in Docker using the Entando S2I images. Images are available for Wildfly 11, JBoss EAP X, and Tomcat.

Using Full Stack Example Images

The project below includes scripts and example images that will boot up a full stack of Entando apps in Docker. https://github.com/entando/entando-ops/tree/master/Docker/Production/entando-full-stack

These projects are useful to quickly boot up an Entando environment in Docker. You can fork or clone them for extension and updates as needed as well. You can find scripts and image names in the repository as well.

Using a Base Image

This base image provides a base Dockerfile that you can extend to create and manage an Entando application. The base image downloads all of the Maven dependencies required for an application to run, meaning that the the initial run can take some time to complete. However, once it does complete, edits to a child Dockerfile will run quickly. The base example in the Readme uses a Jetty/Derby combination for execution.

Access the base image files at: https://github.com/entando/entando-ops/tree/master/Docker/Production/builder-images/docker-builder-5

Using S2I to Build Docker Images

To use the Docker S2I images, you must have already set up a database. See the Setting up a Database section for details.

By default, the app deployed in Docker will connect to a Postgres database to persist resources created using the App Builder and by the MApp Engine. In the app created from the archetype, update the properties in: <your application>/.s2i/environment to point to the user and databases created in the Setting up a Database section.

After configuring the database:

1. Pull in the Docker image using the following command:
   `docker pull entando_wildfly_s2i_5.0.0`
2. Build the image using S2I using the command to build and deploy a Docker app in Docker:
   `s2i build <path or URL of your project> entando_wildfly_s2i_5.0.0 <your image name>`
   Where:
   - **<path or URL of your project>** is the path to your project or a URL to access the code. The build for this project will be invoked and the resulting war file deployed to the app server in the image
   - **Entando_wildfly_s2i_5.0.0** is the name of the base S2I Docker image provided by Entando
   - **<your image name>** is the name for this Docker image

Run the Image Run the image using the following command:
`docker run -d -p 8080:8080 <your image name>`

The app may take some time to start. It is downloading Maven dependencies at startup. Your app will be available on http://localhost:8080/<your_project_name> after startup completes.

By default the image exposes port 8080 however you can change the local port mapped to that value by editing the first instance of 8080 in the Docker run command above.

5.3. USING OPENSHIFT

5.3.1. PREREQUISITES:

- Installed minishift environment

You can configure an Entando app to launch directly into OpenShift using a pre-configured archetype. Create an OpenShift-ready project using the following command:
`mvn archetype:generate -Dfilter=entando-archetype-webapp-openshift`

Using Docker

1. Pull in the Docker image:
   `docker pull entando/app-builder-5.0.0`
2. Run the image. Example Docker command:
   `docker run -it --rm -d -p 5000:5000 -e DOMAIN=http://localhost:8000/my-app appbuilder-5.0.0`
   Where:
   - **DOMAIN**= is the url of a running instance of the MApp Engine. The App Builder uses the REST APIs in the engine to create and manage the application.