Entando Documentation Documentation

Release 4.3.x

Simone

Nov 30, 2017

CONTENTS

Contents:

CHAPTER

ONE

ENTANDO CORE PROJECT

Our goal, in combination with the official https://www.entando.com website, is to provide users and developers an easy way to quickly get started the Entando platform. Throught this wiki, you can learn how to join the **Entando core**.

- Getting Started
- Build from source code
- How the Core works
- How to use the Core
- Bug Tracker
- FAQ

1.1 What is Entando?

Entando is the lightest, open source Digital Experience Platform (DXP) for **Modern Applications**.

Entando **harmonizes** customer experience across the omnichannel applying the techniques of **modern software practices** to enterprise applications.

Enterprises modernizing their application development processes and platforms or building new generation applications also require modern patterns for user interface design and need to replicate them across multiple applications throughout their enterprise in a process called UX convergence; Entando is focused at providing a toolset to meet the needs for these modern applications and UX convergence projects.

Entando then unifies the user experience across different applications, people, devices while reducing technical complexity, time-to-market and project development costs.

Entando can be used both to modernize UI/UX layers on top of existing applications or to build new applications aligned to UI/UX standards, across different industries and use cases. Major use cases are: modern web and mobile applications and new generation web applications.

Open Source license allows you to customize, extend and implement any change you wish to apply to our source code. Our core is available under the **LGPL v2.1** License and our components under the **MIT** License.

1.2 Capabilities

- **Robust and Reliable**: Based on Java technology (Java2EE, Spring, Struts), it's easy to integrate, testable, ready for continuous integration and delivery.
- **Modular and Extensible**: Services, core components, front-end layer and back-end layer are separate elements. Developers may easily work on new apps, extend core components, and create connectors to third-party systems.

- **Ease of Integration and Interoperability**: Entando's open standards-based architecture provides unbeaten integration capabilities. In addition, RESTFUL APIs allow to integrate with third parties. You can expose contents, apps, and integrated systems as a service via APIs.
- **Scalability**: Entando supports both multi-node clustering and replication.
- **Security**: Entando follows OWASP guidelines, encryption technologies (DES, MD5 and RSA), Oauth implementation, LDAP and Active Directory integration, and more. Identity Management modules are extensible with secure user and password policies management, pluggable authentication, log-in procedures. Entando integrates your existing security by acquiring user credentials on a customizable single-sign-on infrastructure based on CAS.
- **Accessibility**: Entando is compliant with international laws and standards in terms of accessibility for people with disabilities.
- **Usability**: Entando is easy to learn and to use.
- **User Experience**: Entando integrates technologies and tools used to provide interaction between a user and a set of applications, processes, content, services or other users.
- **UX Convergence and UI best practices**: by abstracting UI patterns, processes and components, Entando allows easier and quicker to develop UX proof code and to share it across different applications, people and devices.
- **Rich UI Support**: Entando supports user interface (UI) technologies that deliver rich experiences, including HTML5 and related technologies (e.g. CSS3 and JavaScript).

1.3 Main Features

Application Management Features * Centralized Administrative Console * Responsive, Mobile Rendering, Accessible Administration * Integrated Analytics tools * Social and Collaboration tools * Page Designer based on drag and drop feature * Page Preview * Widget-based UI * Host Dashboard

CMS Features

* Contents and digital assets management * Content classification * Content Workflow * Content Versioning * Content scheduling * Georeferenced contents * Authoring * Role-Group based access control * Search Support * Web Form Management * Multilingual support (i18n) * Multi-channel deployment * SEO urls

Framework Features * Components based * Cloud deployment * Multisites support * Templating engine * Identity Management support * Centralized Authentication and Authorization

CHAPTER

TWO

GETTING STARTED

Here what you need to start using Entando:

- Java JDK version >= 7
- Ensure which JDK's version you are using, running the following command at the command prompt:

```
java -version
```

- Maven version >= 2.2.1 (It's recommended to use Maven up to version v3.0.5)

And this if you are going to develop:

- Ant

With the JDK, Maven and Ant installed you are ready to go. Some detail on how to achieve that below.

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

2.1 Setup Environment with OS X 10.9 or newer

- Download the Java JDK 7 from the oracle website 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
- Run brew doctor and fix any warning you get from it
- Install Maven and Ant:

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

2.2 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

2.3 Setup Environment with Windows

- Install Java JDK 7
- Download and install the .exe installer

`bash jdk-7u79-windows-x64` * Create the environment variable

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

- Install Maven

`bash apache-maven-3.3.3-bin` * Set the Maven environment variable

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

- Install Ant

  ```
  apache-ant-1.9.6-bin
  ```

- Set the Maven environment variable

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

- Update the environment variable `"PATH"` in order to execute the programs by shell:

Example:

`bash D:\programmiInstallati\apache-Maven-3.3.3\bin;%JAVA_HOME%\bin;%ANT_HOME%\bin;%SYSTEMR`

2.4 Production Server

If you need to configure a server follow the guide on How to Setup a Production Environment

CHAPTER

THREE

BUILD FROM SOURCE CODE

To download the latest source code:

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

```
mkdir ~/my_new_project
```

- Move to a directory of your choosing

```
cd ~/my_new_project
```

- `clone` in the following sequence `entando-core`, `entando-components`, `entando-archetypes`, `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 use `entando-ux-packages`:

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

- Install, in the following sequence, the `entando-core`, `entando-components`, `entando-archetypes` 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 applications samples, 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 entando-archetypes project.

CHAPTER

FOUR

SERVER INSTALLATION

4.1 Hardware Requirements

The installation of an Entando web application does not need heavy requirements of CPU and memory, however depending on the project configuration, the *development* environment could need heavy CPU and memory usage.

A tipical production server:

- 4 GB di RAM (or more depending on the traffic and users)
- Intel(R) Xeon(R) CPU E5-2680 v2 @ 2.80GHz
- 1 GB of free disk space

The previous hardware configuration has been used in many projects. The disk space is used mainly by the *Maven* dependencies while the installation phase, when in production the occupied space depends of the loaded resources (documents, images, files, etc).

4.2 Software Requirements

4.2.1 Application Server

The following software requirements are needed to compile and run Entando in a server

- Oracle JDK 1.7+ (Oracle website)
- Apache Maven 3+ (Maven website)
- Apache Ant 1.8.x + (Ant website)
- Apache Tomcat 6.x + *or* 7.x+ (Tomcat website)
- JDBC Driver for Tomcat (depending on the DBMS)

4.2.2 Web Server

Needed to serve the static resources and relieve the Tomcat's load. * Apache 2.2 (Apache website)

4.2.3 Database

Entando is ready to run with these DBMS, install accordingly the setup of your web application:

- PostgreSQL 8.4+ (PostgreSQL website)

- MySQL Community Server 5.6+ (Community download page)

4.3 Entando in Production, prepare the server

4.3.1 Install Tomcat

Get the installer from the Tomcat homepage or check if your system package manager provide a preconfigured Tomcat.

4.3.2 Configure the `trimSpaces` Tomcat directive

Open the `<TOMCAT_HOME>\con\web.xml` file and add the `init-param`element into the jsp servlet configuration:

```
<web-app ...>
    ...
    <servlet>
        <servlet-name>jsp</servlet-name>
        <servlet-class>org.apache.jasper.servlet.JspServlet</servlet-class>
        ...
        <!-- copy from here... -->
        <init-param>
            <param-name>trimSpaces</param-name>
            <param-value>true</param-value>
        </init-param>
        <!-- ...to here -->
        ...
    </servlet>
    ...
</web-app>
```

Tip: just find the keyword `trimSpaces` and you will find exactly where to place the init-param element

4.3.3 Install the JDBC driver

Download the JDBC driver and copy it into the `<TOMCAT_HOME>\libs` folder.

Remeber: the JDBC driver is a file with jar extension. If needed unpack the downloaded file and locate the proper jar file.

- PostgreSQL JDBC
- MySQL JDBC
- Derby JDBC

4.3.4 Install Apache2 httpd

Download Apache2 httpd for the webserver installation

- Windows: get apache_2.2.9-win32-x86-openssl-0.9.8h-r2.msi
- Ubuntu: `sudo apt-get install --yes apache2`

- Other systems: check if your system package manager provide a preconfigured Apache2 version 2.2 or check the Apache Http Server page

4.3.5 Tomcat Connectors `mod_jk`

Install and configure the `mod_jk` in order to work with Apache and Tomcat

- **Windows**: download version 1.2.40 and copy the mod-jk file into `apachedir\modules\` folder
- **Ubuntu**: `sudo apt-get install libapache2-mod-jk`
- **Other systems**: check if your system package manager provide a preconfigured package or check the official Apache Tomcat Connector page.

Setup `mod-jk` editing the `apachedir\conf\workers.properties` file, usually it should contain something similar to:

```
worker.list=ajp13
worker.myworker.type=ajp13
worker.myworker.host=localhost
worker.myworker.port=8009
```

4.3.6 Tomcat URI Encoding

Setup the URI Encoding for Tomcat, it allows to process the querystrings with the UTF-8 encoding.

Edit the file '`<TOMCAT_HOME>/conf/server.xml`, add the URIEncoding="UTF-8" attribute to the Connector with port 8080. See The example

```
...
<Server ...>
    ...
    <Service name="Catalina">
        ...

        <Connector port="8080" protocol="HTTP/1.1"
            connectionTimeout="20000"
            redirectPort="8443"
            URIEncoding="UTF-8"
        />

        ...
    </Service>
    ...
</Server>
...
```

4.3.7 Apache VirtualHost

Configure a *virtual host* adapting the pathes for your system.

Remember to change the ''APPNAME''' string with the name of your actual webapp.

```
<VirtualHost FQDN:80>
    ServerName FQDN
    DocumentRoot /var/www/FQDN
```

```
    ErrorLog /var/log/apache2/FQDN_error.log
    LogLevel warn
    CustomLog /var/log/apache2/FQDN_access.log combined

    # Use SetEnvIf to st no-jk when /home/ is encountered
    SetEnvIf Request_URI "/APPNAME/resources/cms/*" no-jk
    SetEnvIf Request_URI "/APPNAME/resources/plugins/*" no-jk

    Alias /APPNAME/resources/cms /var/lib/tomcat6/webapps/APPNAME/resources/cms
    Alias /APPNAME/resources/plugins /var/lib/tomcat6/webapps/APPNAME/resources/
↪plugins

    <Directory "/var/lib/tomcat6/webapps/APPNAME/resources/cms">
        Options -Indexes
        AllowOverride None
        Order allow,deny
        Allow from all
    </Directory>

    <Directory "/var/lib/tomcat6/webapps/APPNAME/resources/plugins">
        Options -Indexes
        AllowOverride None
        Order allow,deny
        Allow from all
    </Directory>

    JkMount /APPNAME/* ajp13_worker
    RedirectMatch ^/$ /APPNAME/
</VirtualHost>
```

CHAPTER

FIVE

HOW THE CORE WORKS

The **Entando Core**, underlying the platform, consists of three main components (engine, admin-console, and portal-ui) plus a set of components that solve specific business needs and a series of orchestration capabilities exposed by using robust APIs and consumed by each application.

- **engine** includes all the basic core features and services of the platform.
- **admin-console** includes basic engine and administration features.
- **portal-ui**: includes basic engine features, administration features and presentation features.

`entando-engine` is a Maven based project defining abstract specifications of the core objects model (Java Class) such as User, Page, Content, widget, and their implementation. It defines interfaces that can manage core services (see the core services list), for example user manager, group manager, content manager, and similar. It defines basic Content Management System (CMS) components, basic database resources, APIs services management. It also provides interfaces to manage integration with a wide range of applications, through REST APIs, RSS, Enterprise Service Bus, web services, and access to all features from a single point of entry. Furthermore, it defines interfaces for secure building and safe administration.

`entando admin-console` is a Maven based project including tools to manage administrative features of the core and WCMS functionalities. For instance, it provides a specific manager for administration area; it manages specific actions into admin console; it defines a specific REST APIs server, defines API OAuth servlets to protect Entando APIs. The admin-console component depends on engine component.

`entando portal-ui` is a Maven based project comprising all the features/tools to create interactive web UIs. It consists of Front-End servlets, functionality for security services to manage user access requests. The portal-ui component depends on admin-console and engine components.

Your first web application example, based on the three core components, can be easily created through the related archetype as you can try in the Getting Started page.

Through the entando portal-ui functionality, an end user uses interactive web UI to enjoy all the Entando services.

Through the entando admin-console functionality, an admin user can enjoy the administration console services.

A Developer User can extend any core components (engine, admin-console, portal-ui), not only to improve functionality of the core itself, but also to create new extensions of the Entando platform such as: **Plugins**, **Bundles** (see entando-components).

As shown in the picture below, an Entando basic application is composed by engine, admin-console, portal-ui core components.

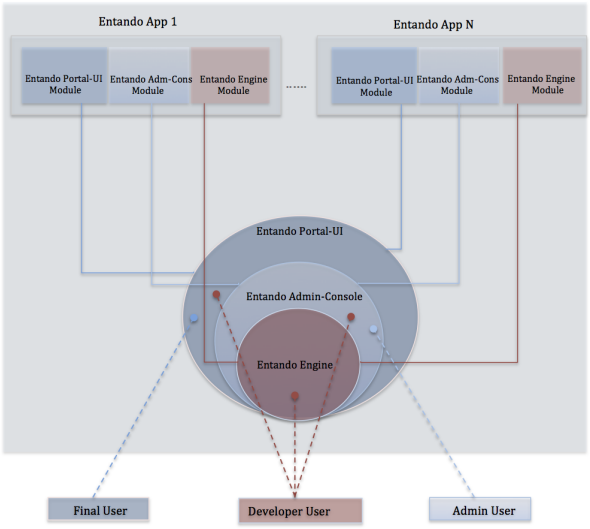

Fig. 5.1: Figure1: Entando Core Diagram
Legend: The picture shows the main components of Entando

5.1 Dependency of admin-console component with engine component

The admin-console component depends on engine component.

Example: Users' Administration

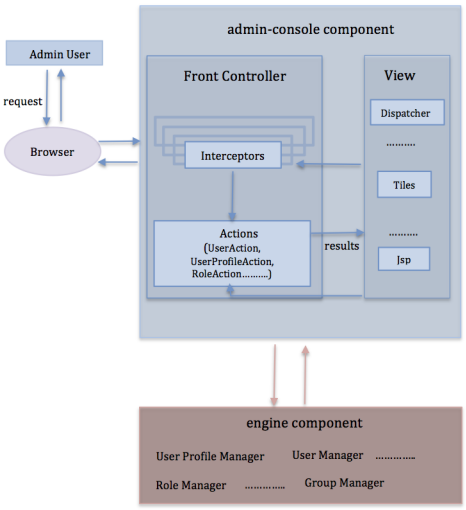

Fig. 5.2: Figure2: Entando Admin Engine

- An Admin User sends a request to the servlet about users administration (profile, role, authorization, etc.). A Dispatcher filter looks at the request and then determines N Actions (search User Action, User Profile Action, Role Action, User Auth Action etc.)

- The Front Controller of the admin-console module invokes M Interceptors, responsible of the request processing (authorization, validation, etc.). Interceptors are applied before and after the Actions execution.
- Actions are executed and the consequently Results are generated.
- A possible Result is the View creation (jsp, json, tiles, etc.), but more different kinds of results can be generated for each Action. The result is returned to the Admin User.
- To execute the Actions, Admin-console component uses appropriate Managers (User Manager, User Profile Manager, Role Manager, etc.) capable to manage, for example, the User object. Managers are included in the engine component.

The execution process of every Action provided by the admin-console is based on Apache Struts2 pattern. The admin-console delegates some responsibilities to the engine component, for example, some control tasks that are executed by the Interceptors classes defined in the determined Stack and some specific actions in the Model (FileSystem, data sources, etc.).

5.2 Dependency of portal-ui with admin-console and engine components

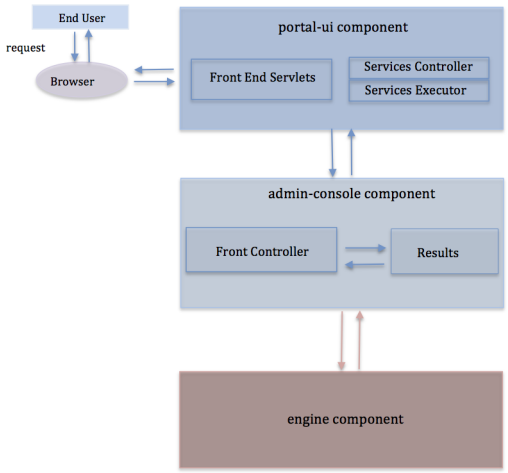

Fig. 5.3: Figure3: Portal UI component

The portal-ui component depends on admin-console component and engine component, as shown in the next picture.

Through the Front-End servlets and specific services controller and services executor, the user's access requests and services requests are managed.

5.3 entando-core services

The Entando's core provides several services:

- **API Rest Server**: managing resources catalogs served through REST APIs.
- **Authentication Provider**: verifying access permissions.
- **Configuration: managing** the core services and plugins configuration.

- **Cache**: allowing content caching for high performance.
- **Category**: managing the Category Tree.
- **Controller**: managing the execution of client's request.
- **Group**: managing user groups.
- **i18n**: providing strings based on the used local language.
- **Key Generator**: creating a sequence of pseudo-random characteristics.
- **Lang**: managing multi-language.
- **Notifier**: allowing events notification.
- **OAuth Consumer**: provides third-party applications for communication through APIs.
- **Page**: managing pages in a tree-like structure and handling the structure of the page itself.
- **Page Model**: managing multiple templates active on Entando web applications.
- **Authorization**: managing the support for authorization and access control services for users.
- **Role**: managing the different roles granted to users (user, admin, editor...).
- **Shortcut**: allowing easy access to information from homepage to administration area.
- **Widget**: allowing widget management.
- **Url**: managing URLs and creating full URLs to a application's page based on essential information.
- **User Manager**: allowing to manage registered users.
- **Installer Component Manager**: allowing the easy installation of plugins and bundles.
- **Database Management**: providing support to the installer manager, and allowing restore and backup of the components.
- **Content Management**: providing the functionalities to manage contents, as modeling of data types, data search, etc.
- **Resource Management**: providing various functionalities for resource management.
- **File Browser**: allowing to browse through directories the folders
- **Fragment**: allowing to manage GUI sections
- **Link Resolver Manager**: allowing to manage symbolic links
- **Action Logger**: allowing to track system actions
- **Storage**: allowing to manage system resources
- **User Profile**: allowing to manage user profile
- **Content Model Management**: allowing to manage content model
- **Renderer Manager**: allowing to check and manage contents in various situations, using content models, links resolving, caching etc.
- **Content Dispenser**: allowing to manage the content dispatcher
- **Template Engine**: allowing to customize dynamically all components of Entando page (widget, resources, files, etc.)

CHAPTER

SIX

HOW TO USE THE CORE

The Entando framework core is based on three main components: `entando-engine`, `entando admin-console`, `entando portal-ui`.

They are essential elements of the framework since provide the useful functionality to realize customized services.

Their high-level description and the interaction process is summarized in the How It Works page. More specifically, it shows how the *admin-console* component interacts with the *engine* component to expose various services and how the *portal-ui* interacts with *admin-console* and *engine* in order to provide the requested services to the final users.

In this page the procedure for using the three components useful to create a web application is described:

- How to use the main services exposed by the *engine* component;
- How to sue the main functionality provided by the *admin-console* component in order to create back-office interfaces;
- How to use the main functionality and tools provided by the *portal-ui* component to create the basic elements for front-end interfaces.

6.1 entando-engine

It is imported as dependency from the archetype and exposes services which are accessed in two ways:

1) Spring Dependency Injection

Configuring the xml of the bean we are creating in order to use the entando-engine services

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:context="http://www.springframework.org/schema/context"
        xmlns:jee="http://www.springframework.org/schema/jee"
        xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.
↪springframework.org/schema/beans/spring-beans-4.0.xsd
                http://www.springframework.org/schema/context http://www.
↪springframework.org/schema/context/spring-context-4.0.xsd
                http://www.springframework.org/schema/jee http://www.springframework.
↪org/schema/jee/spring-jee-4.0.xsd" >

    <bean id="myBean" class="com.my.package.Class" parent="abstractService">
        <property name="userProfileManager" ref="UserProfileManager" />
    </bean>
</beans>
```

For further details on how to use the dependency injection see the Spring documentation website.

2) Application Context

Access the bean of the manager using `ApsWebApplicationUtils.getBean`

```
import com.agiletec.aps.util.ApsWebApplicationUtils;

//...

IAuthorizationManager entandoAuthManager = (IAuthorizationManager)␣
↪ApsWebApplicationUtils.getBean(SystemConstants.AUTHORIZATION_SERVICE, this.
↪pageContext);

//do something with entandoAuthManager...
```

6.2 Entando admin-console

The developer will use the services exposed from the entando admin-console for managing objects and data for the web application's administrators (back-office)

- struts2 infrastructure (interceptors stack, helpers, base actions...)
- back-office templating system
- front-end templating system
- back-office tag libraries
- activity stream
- back-office user shortcuts

Entando admin-console provides all those functionalities needed to manage the applications (widgets, page models and pages) which portal-ui component will use, keep reading the Entando portal-ui.

6.3 Entando portal-ui

It provides the tools needed to create the front-end part of the web application. The three main elements are:

- Page
- Widget
- Content

6.3.1 Pages

Users access the web application through the Entando Pages which are organized in a tree. Portal-ui gives the tools to create, edit and delete the pages and the page tree.

Pages are created using models. Page Models define the structure of the page and tell the system what are the configurable slot available for that page and provide also a template for the GUI Page Models can be managed drom the back-office.

Example of a Page Tree:

- Home (Page Model Two Columns)
  - About Us (Page Model Two Columns)
    - Contact Us (Page Model Three Columns Green)
  - Buy (Page Model Two Columns)
- Pricing (Page Model Three Columns Green)
  - Shipment (Page Model Three Columns Red)

Page Models Xml The xml describes what configurable slots will be supported by the page Example of an XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<frames>
    <frame pos="0">
        <descr>Header</descr>
    </frame>
    <frame pos="1">
        <descr>Body</descr>
    </frame>
    <frame pos="2">
        <descr>Footer</descr>
    </frame>
</frames>
```

There will be possible to add any number of <frame> elements with the only constraint that the pos attributes are numbered correctly. The <descr> element contains the description visible in the back-office, it will show up to the administrator when performing actions with the pages.

Page Models Template They are the template UI of the page and it's written in Freemarker language and with the Entando front-end tag library will be possible to recall the positions content.

Here a template example working with the previous example xml:

```
<#assign wp=JspTaglibs["/aps-core"]>
<!DOCTYPE html>
<html lang="<@wp.info key=”currentLang” />">
    <head>
        <title><@wp.currentPage param="title" /></title>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <link rel="stylesheet" type="text/css" href="<@wp.cssUrl />pagemodes/mycss.css
↪">
    </head>
    <body>
        <div id="header"><@wp.show frame="0" /></div>
        <div id="body"><@wp.show frame="1" /></div>
        <div id="footer"><@wp.show frame="2" /></div>
    </body>
</html>
```

6.3.2 Widgets

Widgets are the bricks that make up the web application. Since version 4.2 is possible to create new widget from the web back-office.

Developers should know in order to add new widget that a new record in the widgetcatalog table of the port database must be created.

```
INSERT INTO widgetcatalog ( code,titles,parameters,plugincode,parenttypecode,
↪defaultconfig,locked,maingroup )
VALUES (
    'mywidget',
    '<?xml version="1.0" encoding="UTF-8"?>
            <properties>
            <property key="en">System Messages</property>
            <property key="it">Messaggi di Sistema</property>
    </properties>',
    '<config>
        <parameter name="myParamName">
            My Param Description
        </parameter>
        <action name="configSimpleParameter"/>
    </config>',
    NULL, NULL, NULL, 1, NULL);
```

Since version 4.2 widgets are rendered using the *Entando Fragments*, which are a piece of Freemarker template giving html as output, html that will be concatenated in Pages. This means that for every widget there is a Fragment making its GUI.

These Entando Fragments can be managed form the back-office. Here an example of the gui code for a widget returning the value of a parameter:

```
<#assign wp=JspTaglibs["/aps-core"]>
Stored value for myParamName is: <@wp.currentWidget param="config" configParam=
↪"myParamName" />
```

6.3.3 Contents

Within the default distribution of Entando *portal-ui* is embedded the CMS plugin, it allows the management of the contents.

From the back-office is possible to manage the content archive and with dedicated widgets is possible to publish lists or single contents.

Content from the CMS are grouped by Content Types. Think of each Content Type as the proprotype describing the content structure. For instance, the application can have the Content Type News, Announcement, Event, Contact... according to the needs more Content Types, this allows to:

- content archive cataloged homogeneously
- easier editing because every field is strictly related to that type
- ui customization with specific templates for each type

More in detail, Content Types structure the contents using basic data type such as texts, numbers, hypertexts, dates and few others.

Example, Content Type News:

- *Title* -> multilingual text
- *Publication Date* -> date
- *Subtitle* -> multilingual long text
- *Body* -> multilingual hypertext

- *Main Picture* -> image
- *Secondary Picture* -> image

With a so defined structure will be easy to introduce the validation check (mandatory fields, length) needed and also will be possible to publish widget with automatic generated lists filtering and sorting using those attributes (i.e. sort by title ascending, then date descending and with date greater than today. . .).

CHAPTER

SEVEN

TECHNICAL SPECIFICATIONS

7.1 Deployment Compatiblity

Operating Systems	**Servlet Containers**	**Application Servers**	**Databases**	**Public & Private Cloud**
Linux (various distributions and versions)	Jetty	JBoss v5	PostgreSQL	Cloud based
Windows	Apache Tomcat (v6, v7, v8)	JBoss 7.5 (JBoss EAP V6.4)	MySQL	Deployable to the cloud (EC2, Heroku, Jelastic, CloudBees) and virtualized environments (VMware)
Mac OS X	JBOSS v5	Apache Tomcat	Oracle	
	Oracle WebLogic	Oracle WebLogic	SQL Server	
			noSQL DBs (MongoDB)	

7.2 Platform Details

Technologies Used	Web Services	**Architecture **	Performance and Scalability	**Scripting Language**	Security	Authentication Management
Java J2EE/EE	JSON	Multi-Layers	Clustering Configuration	Javascript (JQuery)	Pluggable Authentication	Open LDAP
Spring	XML	Components based	Database Replication	HTML5	Email Verification	Kerberos
Struts2	REST	Full support of RESTful APIs	Advanced Caching (Ehcache, Memcached , etc.)	CSS3	Granular Permissioning	SiteMinder (SSO)
AJAX	Spring HTTP	Dependency injection provides pluggable service implementations	Load Balancing Support	AngularJS	LDAP Authentication	CAS
Ehcache			Lucene-Solr Search Platform Support	Others	Session Management	Apache Directory Studio
Memcached			Performance monitoring support (Java profiling , etc.)			
Lucene						
Solr						
Velocity						
HTML5						
AngularJS						
Bootstrap3						
Freemarker						
JSP						
ORM-Lite						
Apache CXF						
JAXB						

CHAPTER

EIGHT

TECHNICAL NOTES

The **release 4.3.x** of the *entando-core* comes with a series of new features, improvements and bug fixes, mostly aimed at improving the user experience of the platform.

8.1 Patternly Client Framework

A new designer for the Admin Console based on the **PatternFly** guidelines. The previous versions of the Entando were only based on Bootstrap framework.

8.2 New organization of Information Architecture

A re-design of the Information Architecture improves the usability of the platform for all users.

8.3 New Admin Console Dashboard

New admin console dashboard lets users visualize data that shows some back-stage information about the applications' contents and page progresses.

8.4 New Page Designer

New feature to create and configure pages for a web application, in a drag & drop way. * different way to create new pages * new graphical editor to customize the page layout * drag&drop feature to configure the widgets in the the page * grid view or list view in a vertical menu is provided for the widgets * new *page status* has been implemented: It indicates if the page is in a *draft*, *online*, *draft-online mode*

8.5 Struts 2 update

Fix of a bug related to a security problem found by Struts 2 and corrected in the current version of Struts 2.5.10.1.

CHAPTER

NINE

FAQS

In this section are available the answers to Entando Frequently Asked Questions (FAQs)

Q: Is Entando free?

A: Yes. It is free to use and modify under the terms of the LGPL v2.1 License for the Core and under the MIT License for the components

Q: Is Entando supported?

A: Yes. We support Entando in open source terms. You can report bugs and expect the same order of responsiveness that you get from any open source project.

Q: Where is the code?

A: The code is here

Q: Where can I get more help? A: Entando offers different support level and feature development. Visit us at our website for more information.

Q: I have a problem with the installation. Where do I go?

A: There is a Getting Started page in this wiki. If your problem is not solved, you may contact us via our website.

Q: How can I change the pages of an Entando web application?

A: Simply enter in the Admin Console of your application. You will be able to modify all the pages templates and contents.

Q: Does Entando allow to use external authentication systems?

A: Yes. In addition to internal users management system, there are several plugins to connect with external Users Management System such as LDAP, Active Directory, Kerberos, CAS, ...

Q: Does Entando allow the management of multilanguage content?

A: Yes. The platform allows the management of multilanguage content, with the possibility of adding or removing languages, and provides the opportunity of selecting the language in which you want to translate the contents.

Q: Does an internal search engine exist in Entando?

A: Yes. Entando provides indexing and searching features. The search engine integrated in the platform is Lucene. In some projects also third-party owners search engines have been integrated.

Q: Is there content versioning in Entando?

A: Yes. A specific plugin allows content versioning.

Q: How to stay tuned with the latest version?

A: Follow those steps * create a generic folder and go to the generic folder just created may I suggest entandoGitHub * Download from Github all required package in the following order: + in a terminal type `git clone https://github.com/entando/entando-core` + in a terminal type `git clone https://github.com/entando/entando-components` + in a terminal type `git clone https://github.com/entando/entando-archetypes` this will create three new folders: entando-core record, entando-components and entando-archetypes for each package be sure to use the branch Master, in each folder use `git checkout master` * check update in entando-core + go to the folder entando-core + `git pull` + `mvn clean install -DskipTests` (this will update your local Maven Repository) + go to the generic folder * check update in entando-components + go to the folder entando-components + `git pull` + `mvn clean install -DskipTests` (this will update your local Maven Repository) + go to the generic folder * check update in entando-archetype + go to the folder entando-archetype + `git pull` + `mvn clean install -DskipTests` (this will update your local Maven Repository) + go to the generic folder

Q: How can I install ImageMagick to create, edit, compose, or convert bitmap images?

A: To install ImageMagick on your OS, you have to:

- For Linux and Mac OSX:
- Install ImageMagick using your package manager
- Open the file *./myprojectexample/src/main/config/systemParams.properties* (After the first starting project and extract the archive)
- Locate the line `imagemagick.enabled=false` and set it to "true"
- For Linux and Windows:
- Install ImageMagick
- Open the file *./myprojectexample/src/main/config/systemParams.properties* (After the first starting project and extract the archive)
- Locate the line `imagemagick.enabled=false` set it to "true"
- Locate the line `imagemagick.windows=false` set it to "true"
- Locate the line `imagemagick.path=null` and specify the installation path of ImageMagick