The Entando Model

Conceptual model of the Entando Platform

Authors

Eugenio Santoboni, AgileTec S.r.l. <e.santoboni@agiletec.it>	Author
Roberta Quaresima, Tzente S.r.l. <r.quaresima@tzente.it>	Author
Andrea Dessì, AgileTec S.r.l. <a.dessi@agiletec.it>	Author
Alessandra Fadda, AgileTec S.r.l. <a.fadda@agiletec.it>	Author
Matteo Minnai Entando s.r.l. <m.minnai@entano.com>	Translation and adaptation

v9.0 Jan 2013

Legal Notice

Introduction

This document describes the conceptual model of Entando.

Target Audience

This document is intended for developers who wish to explore the capabilities of the Entando Platform.

Table of Contents

Introduction to Entando

Entando is an integrated open source enterprise portal platform and content management system. It has been successfully adopted in many demanding public and private sector projects, which have high functional and legal requirements for accessibility, ease of use and mobility. Its agile, modern and user-centric platform is designed for building custom web portals quickly, which can be easily integrated with other systems and changed in line with end-user demands. It's open source technology means that modules or entire portals can be shared and reused throughout organizations.

This document briefly describes the elements that are the foundations of Entando platform:

- Content, Content Models and Categories
- Pages, Pages Tree and Page Models
- Resources
- Showlets
- Links

Please note that some of the elements above belongs to the *core* of the platform – e.g. The Pages Tree – others are specific to the Content Management System which is a *plugin* (whose code is **jacms**) bundled with the platform. For the purpose of this document we will treat them organically without making distinctions between core and plugin elements.

Another important aspect to consider is that Entando makes an extensive use of XML to represent the vast majority of the elements of the list above: some of them need to be inserted manually in the database (for example the definition of Showlets and Page Models) other are created transparently from the administration interface (e.g. Contents and Content Types). In this document we give a definition of each of these elements and when possible their XML structure as found in the database.

Contents

In Entando a **Content** is a set of information built using the so-called **Content Attributes**: a Content Attribute carries a primitive information of a given type. The organization of Content Attributes results in a structure called **Content Type**.

The “atomic” information carried by a Content Attribute might support multiple language or not, depending on the nature of the data itself; in the first case we have a *multilanguage* attribute, *monolanguage* otherwise.

It is important to note that a Content is a pure information: a content never carries information on how it is rendered to the external world: this is a task of **Content Models**.

In other words, a Content Type describes the pure information carried by a content, the Content Model describes how it is rendered: as a consequence every Content has only one Content Type, but it might have several Content models.

Content type

The **Content Type** groups together all contents with a similar structure that use the same set of editing operations in the administration area and the same Content Model in the portal (or front-end).

An example of content type can be a News, an Article, etc. Each content type is defined, in the system configuration, by set of attributes (id, description, etc.) and by the list of content attributes.

The content types available in the system must be declared and defined using XML in the table *sysconfig*, item *contentTypes*. The structure of the content type is the following:

```
<contenttypes>
 <contenttype typecode="____" typedescr="____" viewpage="____" listmodel="___"
 <attributes>
  <attribute name="____" attributetype="____" />
  ..............
 <list name="_____" attributetype="____" nestedtype="___" />
 ..............
 </attributes>
</contenttype>
<contenttype .......... >
.........
```

```
</contenttype>
........
</contenttypes>
```

For each Content Type are defined:

- *typecode*: a unique code that distinguishes the Content Type; it must be composed by 3 alphanumeric characters
- typedescr: a brief description that will appear during the creation of the Content in the Administration area
- viewpage: the code of the page prepared to display the complete content. This is also the default page for the so called on-the-fly publication.
- listmodel: code of the Content Model used to display the content as an item in a list
- defaultmodel: code of the default Content Model

listmodel and *default* model deserve an explanation: both are the codes (or id) of the Content Models used to render the *same* content in two different occasions: when the content appears in a list and when the content is fully detailed, respectively.

An example of the definition of a generic Content Type would be the following:

```
<contenttypes>
  <contenttype typecode="CNG" typedescr="Generic Content" viewpage="contentview"
    <attributes>
      <attribute name="Title" attributetype="Text" indexingtype="text"/>
      <attribute name="Abstract" attributetype="Longtext" indexingtype="text"/>
      <attribute name="TextBody" attributetype="Hypertext" indexingtype="text"/>
      <attribute name="Image_1" attributetype="Image" />
      <attribute name="Image_2" attributetype="Image" />
      <list name="RelatedLinks" attributetype="Monolist" nestedtype="Link"/>
      <list name="Attachments" attributetype="Monolist" nestedtype="Attach"/>
    </attributes>
  </contenttype>
. . .
</contenttypes>
```

All the Content Types generated in the administration interface result in the creation of a XML structure similar to the example above: contents differ only by the attributes defined in their types.

Content Attributes

A Content Attribute (from now on: attribute) is the basic information which form a content. Each attribute is identified, within the Content Type, by a unique name defined during configuration .

For example, the content type Generic content of the Portal Example project project has the following attributes:

- *Title*: multilanguage text type
- *Abstract*: multilanguage textarea type
- *TextBody*: hypertext type
- *Image_1*: image type
- *Image_1*: image type
- *RelatedLinks*: monolist link type
- *Attachments*: monolist document/attach type

In addition, there are other content attributes (for example, attributes of type *Date*, *Number* etc.) useful to define custom content types.

The attributes available in the system to create Content Types are defined in specific system configuration files.

The XML declaration of an attribute has always this form:

```
<attribute|list|table name="ATTRIBUTECODE" attributetype="ATTRIBUTECODETYPE" />
```

please note that attributes *name* and *attributeType* are mandatory; there are other XML attributes available in addition to those above, which depend on the characteristic of the attribute itself (for example the attribute *Monolist*) and others which can be applied to every attribute.

For example the attribute *required* = [true | false] makes the attribute mandatory so that users are forced to specify the related value during content creation or edit; the default value is false.

The *searchable* = [true | false] attribute will result in the value of the attribute being replicated in a table used by the internal search engine for search operations.

Attributes can be subdivided in several ways depending on the characteristic of interest: they can be **simple** and **composite** or **basic** and **complex**: their description in detail is beyond the scope of the present document; for more information please consult the document Attributes Types for Contents.

Content Models

Entando has a formatting engine based on models used to present the content (or part of it) to users: in other words, given a Content (Type!) there is a Content Model for each way it can be presented to the web.

Contents Models are Velocity templates and, as for contents, they are easily managed in the administration Interface.

A possible example of a Content Model *for lists* follows:

```
<a href="$content.getContentLink()">$content.Title.text</a>
```

This Content Mode is used to represent the content in a list, for example as result of a search operation: this simple but powerful configuration creates a link; the link text is taken from the *Text* attribute used as the title, whose name assigned by the administrator is Title.

The link will redirect the user from a list to a page where the content will be fully displayed: in this case a possible Content Model would be the following:

```
<div class="news-2">
	<h2>$content.Title.text</h2>

	#if ( $content.Image1.getImagePath("3")!="" )
	<img src="$content.Image1.getImagePath("3")" alt="$content.Image1.text" />
	#end

	$content.Body.text

	#if ( $content.Image2.getImagePath("2")!="" )
	<img src="$content.Image2.getImagePath("2")" alt="$content.Image2.text" />
	#end

	#if ($content.Attach.size()>0)
	<h3>$i18n.getLabel("ATTACHMENTS")</h3>
	<ul>
	  #foreach ($item in $content.Attach )
	  <li><a href="$item.attachPath">$item.text</a></li>
	  #end
	</ul>
	#end
	#if ($content.Links.size()>0)
	<h3>$i18n.getLabel("LINKS")</h3>
```

```
      <ul>
        #foreach ($item in $content.Links)
        <li><a href="$item.destination">$item.text</a></li>
        #end
      </ul>
      #end
</div>
```

Pages

The page in Entando is like a blank sheet, uniquely defined by its code and title. Pages are arranged hierarchically in a tree, with no exceptions.

Pages in Entando consist of three main elements:

- **Model**
- **Property**
- **Showlet configuration**

In other words a *Page = Model + Property + Showlet Configuration*

Page models

The Page Model defines a characterization for the pages. Within an Entando Portal you can configure multiple page models.

Each page has a single model associated; the Model is characterized by two elements:

- **Scheme**: the division of the page in smaller area called frames.
- **Decoration**: the look and feel of the page

Showlet

Entando is a system that operates on a centralized set of services through components called Showlets.

Each showlet provides part of markup that must be incorporated in the final layout. The engine portal contains the logic control and aggregation of showlets and services. Showlets provide the ideal link between services and portal: in this regard showlets are best viewed as a simplification of portlet.

The Showlet is an object used to display information and functionality or services on the pages of the Portal: each frame of the page, as defined by the Page Model, can host a different showlet.

Note:

Showlets in Entando are similar to the portlet (JSR 168 standard), but are simpler and do not follow the standard: therefore they have a different name.

Showlet = JSP model + Logic + Configuration .

Resources

Resources usually corresponds to information on file, which you can attach to content. Resources can be of different types (eg images, documents, etc.). Each type of resource provides a repository and tools to manage it. A resource, in general, includes several files (eg the same image in different sizes, or a document in several languages) and also support attributes (eg file name, extension, etc..).

A resource is a set of objects on file that form a logical unit. The different files of a resource represent different "instances" or "views" or "versions", for example, different sizes or cuttings of a picture.

The concept of "multiple files for the same resource" is reserved for the following cases:

- the end user is allowed to access a single resource-file per time;
- the context defines which resource-file have to be used.

Resources types:

- **Image**: multidimensional image (resource characterized by a number of instances/views composed by the original image and those obtained from resize).
- **Attach**: word document, pdf, txt or similar (resource characterized by a single instance).

The resource types are defined in XML format on the sysconfig table:

A resource type definition example is shown below

```
<resourcetypes>
  <resourcetype code="Image" folder="cms/images" class="..."
  <resourcetype code="Attach" folder="cms/documents" class="..."
</resourcetypes>
```

The XML associated to an image, automatically generated is the following:

```
<resource id="444" typecode="Image">
  <descr>descrizione</descr>
  <categories/>
  <instance>
    <size>0<size>
    <filename>fotografia_0.jpg</filename>
```

```
    <mimetype>image/jpeg</mimetype>
    <lenght>2.8Kb</lenght>
  </instance>
<instance>
  <size>1<size>
  <filename>fotografia_1.jpg</filename>
  <mimetype>image/jpeg</mimetype>
  <lenght>3.8Kb</lenght>
</instance>
</resource>
<resource id="448" typecode="Attach">
  <descr>descrizione</descr>
  <categories/>
  <instance>
    <size>0</size>
    <filename>schema.pdf</filename>
    <mimetype>application/pdf</mimetype>
    <lenght>440 Kb</lenght>
   </instance>
</resource>
```

Categories

With categories the administrator can classify arbitrarily information; this classification is useful to present aggregates of information to the final users.

A Category is a label that can be assigned to the information so to satisfy the need to aggregate them regardless of the type in a flexible manner. Each category has a *code* and a *title* that can be specified for every language configured in the system

Every Category is inserted in a hierarchical structure organized in a tree, namely **Categories Tree**. It's possible to create other categories and branches of nested Categories.

The ability to group together similar information by argument, makes possible to enlarge the concept of archive by creating sets of selected information .

Categorized contents

The most common use of the categories is to prepare Contents for automatic publishing: the Contents are automatically grouped by category and then shown in the portal pages.

A real use case is represented by a portal page that presents to the users all the Contents whose topic is "Sport". First a category "Sport" must be created if not available in the category tree; then you must associate the category to the target Contents. Finally you have to publish the Showlet configured to show all the contents whose topic is "Sport". The list is automatically refreshed whenever a new Content is labeled with the "Sport" category.

Categorized resources

As for contents, Resources can be grouped with the Categories . This is handy when the number of Resources grows (and so the Categories Tree): in the Resource archive is possible to search by categories to restrict the number of the results found.

User management

The access to information and the ability to use services and perform operations are mediated by the **Permissions**, the **Roles** and the **Groups** associated to the current user.

Users

A User in Entando represents an entity which has the ability to perform operations and access to the information.

The actions allowed to a certain user are given by the combination of the Roles and Groups.

Permissions

A *Permission* represents the authorization to perform a certain action in the portal.

Entando has a number of built-in permissions that restrict the access to the CMS functions: *Content Editing* and *Supervision of Contents*, *Operations on Pages*, *Operations on Resources* e *Operations on Categories* related to the management of information.

There are also a more generic Permissions like *Access to Administration Area* that affects the entire system; one of them, *All functions*, grants access to all the system services.

Note

It's important to understand that Permission are not arranged hierarchically. They only express the ability to perform a certain action.

Each Permission may appear in several Roles and new ones can be further added when the system grows with its functionality.

Single Permissions cannot be directly assigned to the users, but must be grouped in a Role.

Roles

A *Role* is a set of Permissions. You can associate several Roles to a User using the CMS administration interface.

Mixing together Permissions and Roles it is possible to finely tune the accesses and the interaction of the users with services and information.

Like for Permissions, the Roles are mutually independent and cannot be hierarchically arranged.

Groups

A *Group* is a container of Users.

Groups are used to restrict or grant the access to information and services independently of the Roles of the single Users.

Users may belong to several Groups; the access to the functionalities is performed on Group basis.

Groups play a prominent role in the Content Redaction and in the *Pages Tree*.

Plugin

A plugin is is a module that adds new functionalities to Entando. The application services are implemented as Plugins; the CMS itself is a plugin!

A Plugin can be classified as:

- **PurePlugin**
- **Modification**

Several Plugins may be installed in Entando; the way they interact is explained in the document "*Pattern for application services creation and integration*".

Plugins allow a faster development of new functionalities of the system; furthermore they are integrated seamlessly with the underlying system.

PurePlugin

The plugin introducing new functionalities without affecting in any manner the existing core functionalities is said to be *pure*: they are commonly called PurePlugin.

Modifications

The Plugins that extend or modify the existing core functionalities are defined Modification .