

The Orange CMS future concept

- **Basic ideas**

1. Creating relationships between different types of content, e.g. blogs, guestbooks, e-mail forms, news pages, picture galleries
=> semantics
2. Defining individual structures for shared types of content data
=> templates
3. Organizing and linking different types of content data in one data source
=> database tables

- **Advanced techniques**

1. Abstract functions and instances
=> extendable modules
2. Creating a shared output structure to outline internal relationships of content data
=> common I/O language

Introduction

So this is the Orange CMS project. At first glance it looks like any other CMS. It enables users to easily add/edit/delete content data in an existing database. Any visitor then sees the result: one can find a lot of data posted by the users, surrounded by a nice layout template that gives the webpage a certain look.

So what now? There are many different kinds of sites on the web, some for presentation like business sites, some for large communities, some for sharing pictures etc.

But who manages content data? Somehow all data has something in common. Blogs and news pages e.g. are implemented almost in the same way, each post of a blog or news page contains an announcement or other type of text, any post has an author and a date. But what except for humanoid-readable data makes the difference?

A CMS won't need to be able to tell the difference. To be more specific, a CMS wouldn't even need to see any difference in handling between all these kinds of content data. There is only one kind of data in the Orange CMS view: Content data. Content data is designed for human visitors. It is not a CMS's job to explain the semantics, we have another level of application for that. But a CMS can greatly help to improve semantic web data by putting out data in a common way so it can be returned to other storages of data.

Example 1

Imagine visiting a small website. It consists of about 20 pages. We have a tree structured menu and pages look just fine like a newspaper or poster or whatever. Orange CMS now comes to tell the owner that any page of a certain type looks the same. One category in the menu may be recipes. 4 of the 20 pages are recipes, they all have the same structure. On top we find ingredients, in a further description we are told what to do with it to get a nice result, e.g. apple cake. Now the CMS will request two things: A general structure and design of the page, commonly implemented in HTML and CSS, and variables in the structure telling the CMS where to put which data as well as a source where the data is stored.

Conclusion 1

The user has to be able to easily generate templates because we have to understand that a user must not or would less appreciate to be confronted with hardcode. Thus we need an easy-to-use wysiwyg-editor which is just another main part of the CMS and a general language for data usage.

This is where the basic ideas of Orange CMS need to be implemented. A user interface will allow to view the existing structure of the whole website and to create new substructures. Then the CMS will have to give an overview of existing categories of pages so that the user may define a subcategory or a new category for his new page type, in our example cooking recipes.

Example 2

Now the content data, ingredients and instructions, will have to be structured. In XML, it might look like this:

```
<recipe name="apple cake">
  <ingredients>
    <ingredient name="apples" number="5" />
    <ingredient name="eggs" number="2" />
    ...
  </ingredients>
  <instructions set="Wash the apples, cut them into eighths. Open the eggs, put them into a bowl.
  Etc." />
</instructions>
</recipe>
```

Ingredients and instructions are on the same level of our schema. One contains subinstances, ingredients have different names, instructions are plain text. Additionally we will have to know that a cooking recipe consists of only one ingredients header and one instructions body, but may contain several names of ingredients. With such a given structure we could now organize the data we are given, we can reuse a basic template looking like this:

```
<recipe reusable="no" name="|value_recipe_name|">
<ingredients>
<ingredient reusable="yes" name="|value_ingredient_name|"
number="value_ingredient_number" />
</ingredients>
<instructions reusable="no" set="|value_instructions_set|" />
</recipe>
```

The user is now allowed to put his data into a database. But the CMS itself has to give the database structure. There will be exactly two tables, determined by the number of different levels of non-reusable namespaces: recipe and ingredients. These are the two elements of different levels in our structure that values are assigned to. Recipe will be the main table, giving each recipe a unique recipeid and name and a set of instructions. For referential integrity, the recipeid will be our primary key. The ingredients table reuses recipeid as a foreign key and will be used to combine a recipe with certain ingredients. At this stage, the user may choose whether to have more or less referential integrity. He could as well choose to only have one table recipe with a structure like the following:

```
<recipe reusable="no" name="|recipe_name|">
<ingredients reusable="no" set="|value_ingredients|" />
<instructions reusable="no" set="|value_instructions|" />
</recipe>
```

Conclusion 2

So how far may a user be allowed to go? How much referential integrity will be needed? Orange CMS requires this to be open so the user may go as far as he would like. He may even create hundreds of tables for one page type. The CMS manages namespaces and data connection, nothing else.

To ensure functionality, the CMS will have to come with abstract functions and a common input/output language which shall be specified just as introduced. A tag must have one or more properties (e.g. a recipe may have a name), it may contain other tags and it must tell whether to be reusable or not.

to be continued...

P.S.: This is my vision of the Orange CMS. This feature is not yet implemented.

written by Daniel Maslowski, 2007/07/22