



# **Semantic-Driven Content Management**

**Software Requirements Specification**

**for**

**Google Sommer of Code 2008**

Updated on: 22.03.2008

# Inhaltsverzeichnis

<b>1. Introduction.....</b>	<b>3</b>
<b>1.1. About this document.....</b>	<b>3</b>
<b>1.2. Technicalities.....</b>	<b>3</b>
<b>2. Definition of objectives.....</b>	<b>4</b>
<b>2.1. Actual state and motivation.....</b>	<b>4</b>
<b>2.2. Must have criteria.....</b>	<b>4</b>
<b>2.3. Should have criteria.....</b>	<b>4</b>
<b>2.4. Functional constraints.....</b>	<b>5</b>
<b>3. Design goals.....</b>	<b>6</b>
<b>3.1. Maintainability.....</b>	<b>6</b>
<b>3.2. Extensibility.....</b>	<b>6</b>
<b>3.3. Reusability.....</b>	<b>6</b>
<b>3.4. Usability.....</b>	<b>6</b>
<b>3.5. Performance.....</b>	<b>6</b>
<b>4. Development aspects.....</b>	<b>7</b>
<b>4.1. Used tools.....</b>	<b>7</b>
<b>4.2. Communication with the mentor.....</b>	<b>7</b>

# 1 Introduction

## 1.1 About this document

This is a requirements specification for the project „Semantic-Driven Content Management“ within the participation of the PostNuke Application Framework in Google Summer of Code 2008.

This document does not contain specific details like use cases, service conditions or implementation aspects. It lists up the most important aspects, but it can be mostly seen as a rough guideline based on our own ideas. These are not obligatory, we are also looking forward to getting innovative input from the students.

## 1.2 Technicalities

- Project: Semantic-Driven Content Management
- Difficulty: hard
- Required experience: PHP, XML, XSL

This project will be mentored by {MENTOR NAME}. Please find more information about communication between mentors and students at chapter 4.2 on page 7.

## 2 Definition of objectives

### 2.1 Actual state and motivation

As the internet evolves new web clients are appearing continuously. Separating form and function becomes more important than ever – not only in a content management system itself, but also in the static content parts a website may produce.

Several clients use different technologies and standards. If one wants to support them all, a lot of templates have to be adapted for each project.

### 2.2 Must have criteria

PostNuke should become able to serve appropriate content for each type of user agent without manual efforts. Therefore this project is about creating a new approach of storing and managing information.

An editor should only define the structural/semantical information of his article which is being stored in XML for example. In addition a few "channels" are selected (which can be web, mobile, office aso) defining how the final rendering will happen and look like.

The primary goal is the ability to publish content for different client types at once. One possibility for the technical realisation is XSL processing converting XML content to other file types. A quite complex task is evaluating how the different channels are maintained. For example a merge functionality would be imaginable to synchronize changes.

### 2.3 Should have criteria

It would be nice if some additional channels could be provided for transforming content not only into different (X)HTML formats/templates, but also to file types like PDF or XLS.

XSL is very interesting and powerful, because for example it allows a clean and reliable PDF generation without broken paragraphs and other ugly effects.

Also serving arbitrary content in different feed formats automatically is a nice use case. Further special transformations are supposable for handling web services.

To improve the semantical power of this solution it could be pointed out how media is being handled in this environment relating inheritance of semantic content information.

## Chapter 2 - Definition of objectives

### 2.4 Functional constraints

None.

## 3 Design goals

### 3.1 Maintainability

The whole source code should follow the PostNuke coding standards and contain some comments for improved understandability.

Ideally some pattern are used during the design phase for reducing the amount of physical maintenance points.

### 3.2 Extensibility

Ideally the whole transformation functionality will be implemented in a class structure with special interfaces taking care for encapsulation and easy addition of extensions. For example including a new output channel should not affect the internal components.

### 3.3 Reusability

If this class structure can be kept quite independent of the data fields it processes, this would result in a greater reusability for other modules. Using interface types in operative code ensures exchangeability through polymorphism.

### 3.4 Usability

An easy and intuitive user guidance is necessary for acceptance on the part of the community.

If possible within the given time frame a small documentation could be written describing functionality and usage.

### 3.5 Performance

For using this project in productive areas caching mechanism might be needed to reduce the server load caused by the XSL transformations.

## 4 Development aspects

### 4.1 Used tools

There are no particular requirements. In general the following software components are needed for this project:

- Webserver (e.g. Apache)
- Database (e.g. MySQL, PostgreSQL, MS SQL or Oracle)
- PHP 5
- Latest version of PostNuke .8
- Programming editor (either a comprehensive IDE like Eclipse with PDT or a text editor with syntax highlighting)
- XSL Processor

### 4.2 Communication with the mentor

We created a mailing list where general things can be discussed between all mentors and students. The mentors are looking forward to some exciting projects.

If you have any further questions please contact the PostNuke Steering Committee at [sc@postnuke.com](mailto:sc@postnuke.com).