JooMDD: A Model-Driven Development Environment for Web Content Management System Extensions

Dennis Priefer
KITE - Kompetenzzentrum für Informationstechnologie, Technische Hochschule Mittelhessen, Germany
dennis.priefer@mni.thm.de

Peter Kneisel
KITE - Kompetenzzentrum für Informationstechnologie, Technische Hochschule Mittelhessen, Germany
kneisel@mni.thm.de

Gabriele Taentzer
Philipps-Universität Marburg, Germany
taentzer@mathematik.uni-marburg.de

ABSTRACT

Developing software extensions for Web Content Management Systems (WCMSs) like Joomla, WordPress, or Drupal can be a difficult and time consuming process. In this demo we present JooMDD, an environment for model-driven development of software extensions for the WCMS Joomla. JooMDD allows the rapid development of standardised software extensions requiring reduced technological knowledge of Joomla. This implies that even inexperienced developers are able to create their own functional WCMS extensions. This demonstrates that a model-driven approach is suitable for the domain of WCMSs.

A supporting video illustrating the main features and a demonstration of JooMDD can be found at: https://youtu.be/Uy_WBIjPldI.

CCS Concepts

•Software and its engineering → Application specific development environments; Software design techniques; Software prototyping;

Keywords

Model-Driven Development; Web Content Management Systems; Joomla

1. INTRODUCTION

In today's web, Open Source Web Content Management Systems, or so called "second-generation content management systems" [13] like Joomla [3], WordPress [7], and Drupal [1] play a significant role. Due to their providing core functionality, they are used as base platforms for dynamic web applications by experienced web designers and inexperienced users alike.

One of the biggest advantages of using a WCMS as a platform for dynamic websites is the functional extensibility through several standardised extension types. Through the use of extensible APIs, extensions can be implemented without changing the platform itself. Furthermore, through using already developed API functionality, written extension code can be reduced, which in turn allows a more rapid development. Experienced extension developers can use such mechanisms to increase the development speed. Inexperienced developers get less use from the API, because they don't know which functions are available and when to use them.

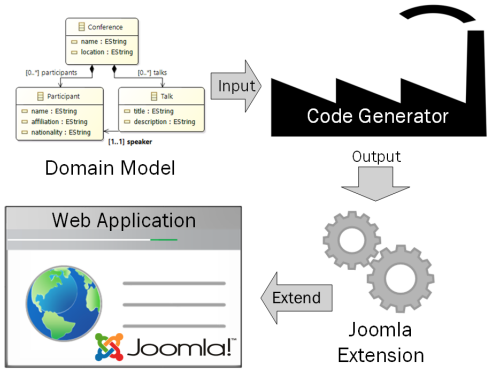


Figure 1: Model-Driven Tool Chain for the Development of Joomla Extensions

Due to this problem and the fact that most WCMS extensions, independent of their underlying platform, contain much redundant code, we propose a model-driven approach to develop WCMS extensions faster and more easily in comparison to manual programming. We assume that the creation of design models and the subsequent code generation increase the development speed independently of the developer's experience. To demonstrate the usefulness of our approach in the domain of WCMSs, we have developed an environment for the model-driven development of Joomla extensions. Figure 1 illustrates our model-driven approach. Generated extensions can be installed on existing Joomla-based websites to extend their functionality with features, which are not part of the core system itself. Another advantage of a model-driven approach is, that whenever the underlying platform is being updated to new versions, the models can be reused without changes. Only the generator has to be updated by the new features.

We selected Joomla as the target platform, because it is one of the most widely used WCMS [6], it provides a large core platform with various functions and can be used to create either simple or complex websites.

Within the next sections we describe how Joomla-based

ICSE '16 Companion, May 14 - 22, 2016, Austin, TX, USA

ISBN 978-1-4503-4205-6/16/05…$15.00

DOI: http://dx.doi.org/10.1145/2889160.2889176

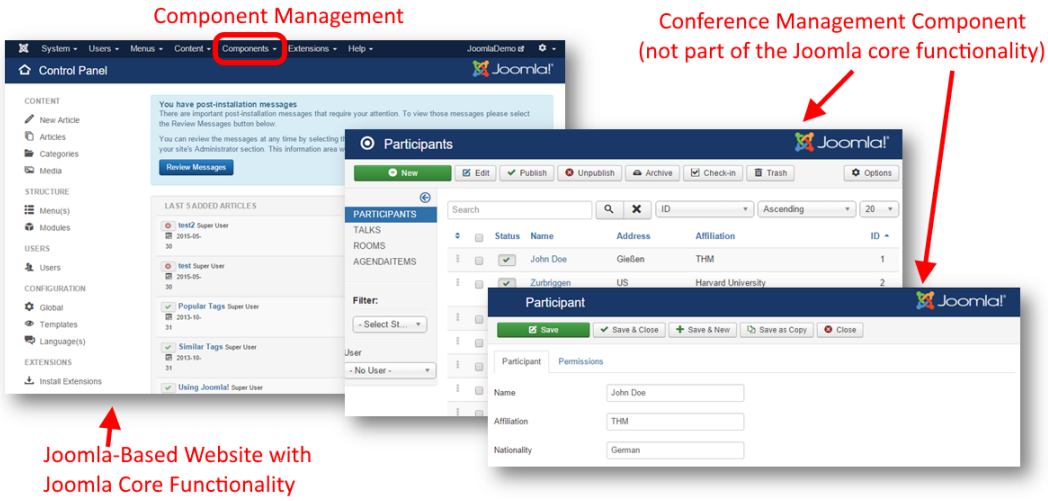


Figure 2: Joomla-Based Website and Component Views of a Conference Management (Joomla 3.x)

websites can be extended in a model-driven way, followed by a presentation of the supporting tools. Our approach is explained using the example of a conference management extension.

2. EXTENSION OF JOOMLA-BASED WEBSITES

To extend Joomla, developers may implement several extension types, which can be installed into a Joomla-based website. The most complex extension type is called ***Component***. This complexity is even reflected by the menu structure within the administration of a Joomla-based website. Examples for components can be the implementation of a web shop or, as in our example, a conference management with all the necessary functionality. The Joomla platform provides a wide range of core functionality, whereas components can provide specific features within a Joomla-based website using functionality of the API. This ensures a homogeneous look and feel for users of the website. Figure 2 illustrates a typical administration section of a standard Joomla website and shows some component views, which are not part of the Joomla core. In general, components have their own data model, represented in the form of database tables. These extend the basic application's database. Joomla highly recommends that components implement a Model View Controller (MVC) architecture on file and code base. Following this structure allows a more comprehensible way of using the API. However, this architecture also brings a certain overhead and schematically redundant code which requires more effort during the manual implementation process.

To tackle this problem, we present a model-driven approach to develop Joomla extensions illustrated using the example of a conference website. Proceeding from the assumption, that a conference management is not part of a standard Joomla application, the required functionality has to be implemented in the form of at least one Joomla component. In addition, using the environment, presented in this demo, Joomla extension developers are able to create even other extension types like *modules*, *plugins*, *libraries* and *templates* in a model-driven way.

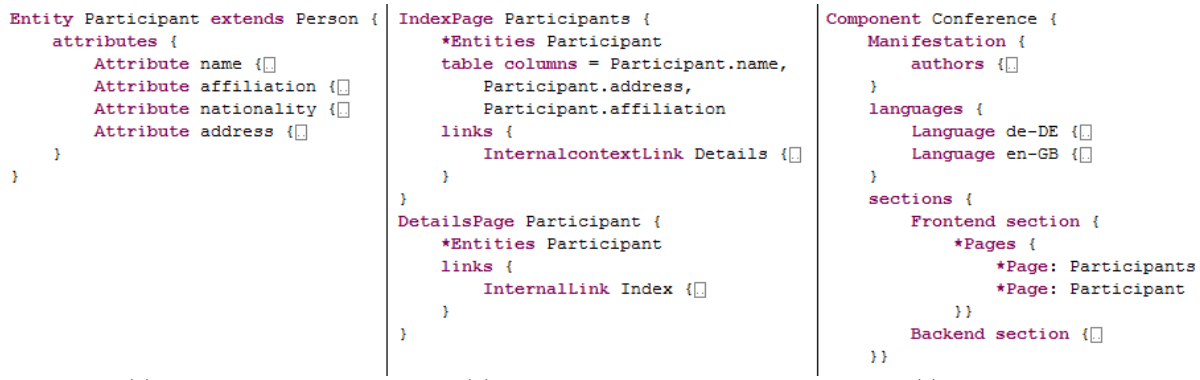
3. A DOMAIN-SPECIFIC LANGUAGE FOR CMS EXTENSIONS

To ensure the universality of our approach, our Domain-Specific Language (DSL) should allow the creation of models as abstract as possible. At the same time the models must be as concrete as needed to ensure a code generation as complete as possible. We defined a DSL for WCMS extensions named **eJSL**. This language is based on the Simple Web Application Language (SWAL) [10], which separates web applications into a data model and a hypertext model for defining the page flow of a website. We adapted this approach to model WCMS extensions. Our DSL contains three main parts: ***Entities*** (data modelling), ***Pages*** (page flow), and ***Extensions***. Figure 3 shows how to apply this DSL to model a simple conference component. The illustration shows an entity section (Figure 3a), which represents the data entity *Participant* and a page section (Figure 3b) for a *list view*. This list view represents all the existing participants with a link to a *detail page*, which shows the details for one participant. Furthermore, an exemplary model definition of a Joomla *component* is shown in Figure 3c. This model contains references to the previously defined pages, and includes them as component views.

The entity and page parts can be used independently of the underlying platform, whereas the extension part of our modelling language is currently tailored to Joomla extensions, specifically to the different extension types characteristic to Joomla. These types do not necessarily apply to other WCMSs. We plan to generalise the extension part, and enable it to be used for the creation of platform-independent extension models.

4. DEVELOPMENT ENVIRONMENT FOR JOOMLA EXTENSIONS

According to [11], PHPStorm [4] by JetBrains is the most popular development environment in the web domain, in which our target users develop. This IDE is based on the IntelliJ IDEA [2], a professional and widely distributed Java IDE. This IDE is easy to use and provides a fast familiarisation. To support this trend, we created an IntelliJ plug-in with Xtext [8]. This allows the rapid development of a text-



(a) Entity (b) Index- and Details Page (c) Joomla Component

Figure 3: Excerpt of a Simple Conference Model Based on the DSL (eJSL) for WCMS Extensions

based editor for our DSL. Used together with our Joomla extension-specific generator, which is written in Xtend [8], we get a working environment. In creating the plug-in for IntelliJ, we get to use JetBrains' core environment and port these plug-ins to its base environments such as PHPStorm. Figure 4 illustrates the textual editor of JooMDD within the IntelliJ IDEA. This editor supports code completion, model validation, and quick-fixes.

Currently, our generator creates code for Joomla 3.x extensions. These can be found within the `src-gen` folder of the eJSL project, wherein the extension models are created. The generator structure itself follows a modular architecture. This allows its straightforward enlargement to generate code for other Joomla versions or even for other WCMSs, without starting from scratch. Generated extensions can be installed directly into a Joomla-based website. They follow the Joomla standards ensuring a homogeneous look and feel for Joomla users (c.f. Figure 2).

In addition to the IntelliJ plug-ins, JooMDD also has textual and visual editors for the Eclipse IDE. The use of the textual editor and the behaviour of the built-in code generator is identical to their counterparts for IntelliJ. Other tools are being planned, among them is the creation of a web-based model editor which allows the creation of eJSL models.

5. PRELIMINARY EVALUATION

A model-driven approach must allow the development of simple, such as the example in this paper, as well as complex extensions. Such extensions are currently installed within the department websites of our university, providing features for the academic sector. One example is the staff management extension family ***THM Groups***[1]. Recently, JooMDD was used to implement a functional version of the *THM Groups* component. Currently, we are planning the development of a pre-course management extension with JooMDD.

To evaluate the usefulness of our approach we have to compare the development of WCMS extensions in a model-driven way with the traditional manual development. More precisely we want to find out if a model-driven approach can quantifiably speed up and simplify the development and maintenance of WCMS extensions.

The conference example shows that development effort can be reduced considerably using our tools. The same applies to our more complex in-house component *THM Groups*. Table 1 illustrates the number of entities (E), pages (P), code lines of the extension model (*LoC (M)*), and the lines of generated code (*LoC (GC)*) for both components. We assume that the amount of manually written code without the use of JooMDD is at least equal to the amount of generated code. The DSL model abstracts from technical details and allows to concentrate on the extension itself. [12] conducted a research on the usability of DSLs with the result that the use of DSLs improves the development speed, code quality, and ease of learning.

Table 1: Model-Driven Joomla Components

Component	E	P	LoC (M)	LoC (GC)
Conference	4	8	220	5964
THM Groups	17	14	691	16194

Many Joomla developers have expressed interest in using JooMDD especially for the migration of their developed extensions, as proposed in [14]. This gives us the opportunity to provide our infrastructure directly to a large group of developers as part of a field study (in vivo). We plan to collect the findings of the study and hopefully empirically infer the usefulness of our approach.

In addition we plan a field study within a controllable academic environment (in vitro) with developers having different experience levels to evaluate our approach. A set of controlled experiments is planned to quantify the qualitative findings of the case studies.

The implemented and planned tools for different IDEs should prevent biases favouring any specific IDE from any collected data.

6. RELATED WORK

In accordance to [13], there is little recent research on modern web development practices, especially concerning the use of current WCMSs. Model-driven development is one such modern development practice. Using this approach, developers elevate application development to a more abstract level, which may save maintenance time and effort. This approach is important in today's research and has enjoyed much attention over the past decade.

General web approaches like [9] can be used to create complete websites, but are not suitable for our problem since they do not address WCMSs and the model-driven development of their extensions. [16], [17], and [15] deal with

[1]An example view can be found at [5].

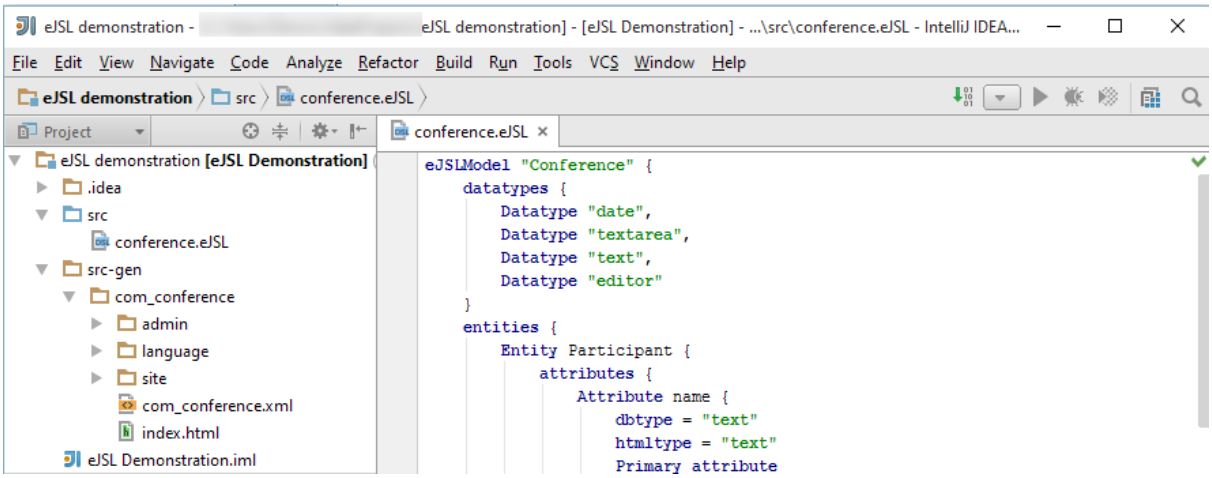


Figure 4: eJSL Editor within the IntelliJ IDEA

platform-independent meta models for the WCMS domain, whereas [15] also generates concrete WCMS-based web applications. However, all of these works overlook their models' extensibility, which necessitates further abstraction.

7. CONCLUSIONS

Web Content Management Systems are the leading platforms for dynamic web applications. Their functional extensibility is a powerful feature allowing developers to build individual software without changing the platform itself. Even though modern WCMSs provide extension APIs for a simplified implementation of extensions, this can still be a time-consuming and complex task for extension developers, even for experienced ones.

In this demo, we present JooMDD, an environment which allows the model-driven development of software extensions for the WCMS Joomla. This enables the rapid development of standardised Joomla extensions for Joomla version 3.x. In addition, the model-driven approach further simplifies development, which hopefully will help inexperienced developers to create their own extensions. The tools can be used within different development environments like IntelliJ IDEA and Eclipse. Further tools, including a web-based editor, are in progress. Furthermore, we plan to make the approach platform-independent, so that other WCMSs like WordPress or Drupal will benefit from it.

8. REFERENCES

[1] Drupal.org [online]: https://www.drupal.org/.

[2] IntelliJ IDEA [online]: https://www.jetbrains.com/idea/.

[3] Joomla!.org [online]: https://www.joomla.org/.

[4] PhpStorm [online]: https://www jetbrains.

[5] THM - MNI Website [online]: http://www.mni.thm.de/index.php/fachbereich/mitarbeiter.

[6] Usage Statistics and Market Share of Content Management Systems for Websites [online]: http://w3techs.com/technologies/overview/content_management/all.

[7] WordPress.org [online]: https://wordpress.org/.

[8] L. Bettini. *Implementing domain-specific languages with Xtext and Xtend: Learn how to implement a DSL with Xtext and Xtend using easy-to-understand examples and best practices*. Packt Publ, Birmingham, 2013.

[9] M. Brambilla. *Interaction flow modeling language: Model-driven UI engineering of web and mobile apps with IFML*. Morgan Kaufmann, Waltham, MA, 2015.

[10] M. Brambilla, J. Cabot, and M. Wimmer. *Model-driven software engineering in practice*. Morgan & Claypool, San Rafael, Calif., 2012.

[11] B. Skvorc. *Best PHP IDE in 2014 - Survey Results* [online]: http://www.sitepoint.com/best-php-ide-2014-survey-results/.

[12] J. Kärnä, J.-P. Tolvanen, and S. Kelly. *Evaluating the Use of Domain-Specific Modeling in Practice*. In *Proceedings of the 9th OOPSLA Workshop on Domain-Specific Modeling (DSM '09)*, volume B-108 in HSE-Print. Helsinki, 2009.

[13] M. C. Norrie et al. *The Forgotten Many? A Survey of Modern Web Development Practices*. In *Current Trends in Web Engineering*, volume 8541 of *Lecture Notes in Computer Science*, pages 290–307. Springer International Publishing, Cham, 2014.

[14] D. Priefer. *Model-driven development of content management systems based on Joomla*. In *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering : September 15-19, 2014, Västerås, Sweden*, pages 911–914. ACM, New York, 2014.

[15] J. d. S. Saraiva. *Development of CMS-based Web Applications with a Multi-Language Model-Driven Approach*. Dissertation, Universidade Técinica de Lisboa, Lisbon, Portugal, 2012.

[16] F. Trias. *Building CMS-based Web applications using a model-driven approach*. In *Sixth International Conference on Research Challenges in Information Science*, pages 1–6. IEEE, Piscataway, New Jersey, 2012.

[17] K. Vlaanderen, F. Valverde, and O. Pastor. *Model-Driven Web Engineering in the CMS Domain: A Preliminary Research Applying SME*. In *Enterprise Information Systems*, volume 19 of *Lecture Notes in Business Information Processing*, pages 226–237. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.