Weaving Web Applications with WebDSL (Demonstration)

Danny M. Groenewegen, Eelco Visser

Report TUD-SERG-2009-021
This paper is a pre-print of:


@InProceedings{GV09,
    author = {Danny M. Groenewegen and Eelco Visser},
    title = {Weaving Web Applications with WebDSL (Demonstration)},
    booktitle = {Companion to the 24th ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2009)},
    year = {2009},
    editor = {Gary T. Leavens},
    address = {Orlando, Florida, USA},
    month = {October},
    publisher = {ACM Press},
}

© copyright 2009, Software Engineering Research Group, Department of Software Technology, Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology. All rights reserved. No part of this series may be reproduced in any form or by any means without prior written permission of the publisher.
Weaving Web Applications with WebDSL (Demonstration)

Danny M. Groenewegen  
Software Engineering Research Group,  
Delft University of Technology, The Netherlands  
d.m.groenewegen@tudelft.nl

Eelco Visser  
Software Engineering Research Group,  
Delft University of Technology, The Netherlands  
visser@acm.org

Abstract

WebDSL is a domain-specific language for the development of web applications that integrates data models, user interface models, actions, validation, access control, and workflow. The compiler verifies the consistency of applications and generates complete implementations in Java or Python. We illustrate the key concepts of the language with a small web application.

Categories and Subject Descriptors D.2.3 [Software Engineering]: Coding Tools and Techniques; D.3.4 [Programming Languages]: Processors

General Terms Languages

Keywords domain-specific languages, web application model, data model, data binding, access control

1. Motivation

The implementation of web applications comprises many technical concerns, including data representation, querying, and modification, user input, data validation, user interface design, and navigation. These concerns are often addressed by separate languages. For example, in one configuration of the Java web programming platform we find the Java general purpose programming language, the SQL query language (or some dialect such as HQL), the JavaServer Faces (JSF) presentation language with the EL expression language for accessing data, the CSS stylesheet language, and other XML schemas for configuration such as page flow declarations.

While separation of concerns and 'choosing the right language for the job' are conceptually appealing, the amalgam of languages used in a single web application project are typically poorly integrated, with an adverse effect on productivity and software quality caused by boilerplate code, loose coupling, and a lack of static verification.

2. WebDSL

WebDSL [3] is a domain-specific language for the development of web applications that integrates data models, user interface models, actions, styling, access control [1], data validation, and workflow [2]. While these different concerns are supported by separate domain-specific sub-languages, the static semantics of the language verifies the consistency of the different concerns of an application model. The WebDSL compiler generates a complete implementation in Java or Python without the need to write further code in these languages.

3. Example: WebTasks

We illustrate the features of WebDSL with a small web application for managing tasks (WebTasks). During the demonstration we will build this application from scratch, while introducing the concepts of the language. The next page illustrates the main concepts with a fragment of the WebTasks application. Figure 1 shows screenshots of three types of pages from the WebTasks application. Figures 2 to 7 show the WebDSL code for these pages.

Acknowledgments

This research was supported by NWO/JACQUARD project 638.001.610, MoDSE: Model-Driven Software Evolution.

References


Figure 1. Screenshots of the WebTasks application.

Figure 2. Data model defines entities with properties. Entity declarations are mapped to a database schema and objects are automatically persisted to the database. Validation constraints (username) pose extra requirements on entities. A derived property (todo) is a transient property whose value is computed from other properties.

Figure 3. Page definition (Fig 1(a)) defines view of properties of the parameter objects of the page.

Figure 4. Data input forms (Fig 1(b)) provide automatic data binding of form fields to entity properties.

Figure 5. Page definition for user task list (Fig 1(c)) with navigation to archive page.

Figure 6. Template definitions define page fragments that can be reused in multiple page definitions.

Figure 7. Access control rules restrict access to pages, templates, or actions using Boolean constraints on the data model.