

UML Profile for Analysis and Design of Jakarta Struts Framework Based Web Applications

Piotr Karwaczyński¹ and Łukasz Maciejewski²

¹ Division of Computer Science, Wrocław University of Technology, Poland
`piotr.karwaczynski@pwr.wroc.pl`

² Institute of Materials Science and Applied Mechanics, Wrocław University of Technology, Poland, `lukasz.maciejewski@pwr.wroc.pl`

Abstract. Jakarta Struts Framework is a mature open source solution for the development of Java-based web applications. In this paper the authors propose UML profile supporting such development process. The profile is adapted to five models describing different stages and viewpoints of analysis and design of Struts-based applications. As the profile and the range of chosen models proved their usefulness in real life scenarios, the fragments of commercial web application model are also presented.

1 Introduction

JSP scripting language, Java language and Jakarta Struts Framework are one of the most popular solutions for the web applications development. JSP and Java are Sun Microsystems products whereas open source Struts are part of the Jakarta Project of the Apache Software Foundation.

Tools mentioned above are very useful and speed up the process of web application development but don't offer any accompanying solutions for design and analysis. The authors, basing on their experience, point out two major deficiencies:

- lack of suitable notation that will meet web application development process,
- lack of tools for web applications visual modeling.

In general the notation proposed by Connalen [1] is the most suitable one for web application domain. In fact it does not express all the entities and abstractions required in a satisfactory way. The authors propose a specialized notation ready to use with Struts framework solution. The notation is based on UML [2] and is developed as an UML profile dedicated for Java, JSP and Struts web application modeling.

Among many tools available for the web development utilizing the internet technologies listed above it is worth to mention Struts Studio by Exadel and Eclipse (an open source IDE) with a set of appropriate plugins. Using this tools one may operate on a graphical representation of a web application in order to:

- generate a basic application structure and its configuration files,

- edit configuration files with tree-like configuration visualization,
- partially visualize flow controls.

Unfortunately the graphical representation is just a part of an application model and does not cover all the web developer need in analysis and design. The authors assume that the proposed set of models may be incorporated into existing development tools (i.e. Eclipse) as a specialized plug-in and utilized effectively.

2 Jakarta Struts Overview

Jakarta Struts are part of the Jakarta project held by Apache Software Foundation as well as Tomcat, Ant or Velocity. Since June 2001 Struts has been available as an open source under the Apache Software Licence. It means that Struts may be used for private and commercial application free of charge.

Jakarta Struts base on an architecture model commonly known as the Model 2. It was described in the Servlet/JSP Specification v.0.92. The model specifies rules of servlets and JSP pages utilization in one application. According to this model JSP pages are responsible for presentation part, while servlets control data access operations and navigation in the application.

Essential components of Jakarta Struts are:

- servlet `ActionServlet`,
- class `Action`,
- class `ActionForm` and accompanying `JavaBean` subclasses,
- class `ActionForward`.

`ActionServlet` controls flows in the application. Receives application server requests (Tomcat, JBoss, etc.) and specifies, basing on an URI, which the `Action` class is supposed to deal with the request. The `Action` class verifies input data and refers to a business layer to use existing data sources.

In order to service a request information on passed values is required. Such information is passed by the `ActionServlet` to appropriate `JavaBeans` that are subclasses of the `ActionForm` class. The servlet takes a decision which `ActionForm` to choose, as it was in the case of the `Action`, basing on an URI.

HTTP requests are passed by the `Action` to appropriate JSP pages. Paths to JSP pages are stored in the `ActionForward` class. While operations in a business layer are finished the `Action` class passes results to the servlet which finally generates a response (a web page) for the HTTP request. All necessary information required to service a request the servlet takes from an `ActionMapping` object that is related with a specified path or an URI.

Information about the `Action`, the `ActionForm`, the `ActionForward` and the `ActionMapping` is stored in a `struts-config.xml` file which is read by the `ActionServlet` at a start. Basing on the information from the configuration file, the servlet builds an object database which is then called by all of the Struts components.

Jakarta Struts Framework is build upon the *Model View Controller* paradigm. Delivering the controller layer it fills a gap existing in a domain of web applications based on Java technology. The main advantages of Jakarta Struts are:

- delivery of uniform design patterns,
- delivery of a source code and configuration files framework which significantly speeds up projects development, especially in initial stages,
- easiness to share work in teams while developing applications due to separation of a Java source code and a presentation part.

Detailed description of Jakarta Struts architecture may be found in [3].

3 Requirements Specification for Analysis and Design

The authors, experienced in the Struts-based web application development, distinguished five models that present simplified views of a system:

- Web Application Model,
- Use Case Model,
- Flow Model,
- Structure Model,
- Presentation Model.

3.1 Web Application Model

This is the most general model. It presents the fundamental division of the system on smaller subsystems. Every subsystem is related with a corresponding use case package that it realizes.

3.2 Use Case Model

The Use Case Model is useful while functional requirements of the system are formulated. Every use case is defined by an appropriate flow diagram.

3.3 Flow Model

This model describes control flows between web application components. It consists of every possible actions and JSP pages. Static aspect of each action is presented by a class diagram. The construction of every JSP page is described by presentation model.

3.4 Structure Model

The Structure Model presents all of the available actions that may be realized in an application and relations between this actions. Among the classes one may distinguish classes that realize operations over data, wrappers, classes that control forms and classes that realize control flows.

3.5 Presentation Model

The Presentation Model introduces construction of every JSP page. The page is presented as a template with specified template items that fill the template.

4 UML Profile for the Development of Jakarta Struts Framework Based Web Applications

4.1 UML Extension Mechanisms

There are some standardized mechanisms of UML allowing its extensibility. Due to them it is possible to adjust the set of UML modeling elements to the specific requirements of any modeled domain [2], [4].

The very basic mechanism of UML extensibility is a stereotype. It allows defining new modeling elements as subclasses of existing UML metaclasses, together with their metaattributes and new semantics. The elements newly introduced have to be consistent with the standard UML semantics. Their role is to precise UML semantics to the specific requirements. The stereotype is defined by its name, base class (UML metaclass), graphic icon (optional) and semantics.

In order to specify properties of modeled elements, tagged values are used. Tag definitions introduce new types for tagged values and are defined in the form of embraced in braces, comma-separated strings of *name:type* pairs. The concrete values of properties are written in the form of embraced in braces, comma-separated strings of *name=value* pairs, where a name is derived from the proper definition of tagged value. Another UML extension means are **constraints**, used to precise the semantics of modeled elements. They are mainly expressed in the specialized language, Object Constraint Language OCL, but also using any mathematical notation, natural or programming languages. The constraints have the form of logical expressions, which are true only when the model is well formed.

The modeling elements, defined for specific purposes, are usually grouped into **profiles**, i.e. stereotyped packages extending the UML metamodel. In the article the authors present the UML profile for the domain of analysis and design of Jakarta Struts Framework based web applications.

4.2 Summary of UML Profile Elements




The set of UML modeling elements, being used in the five models presented in the chapter 3, is summarized in a table 1. In the table, except for the name of an element, its base class and textual and (optionally) graphic³ notation are specified.

4.3 Modeling Elements Semantics

Web Application Model elements.

³ The graphic notation of elements is modeled on [1]

Table 1. The set of modeling elements, their base classes and notation

Element name	Base Class	Notation
WebApplicationModel	Model	package stereotyped <<webApplicationModel>>
WebApplication	System	package stereotyped <<webApplicationModel>>
WebModule	Subsystem	package stereotyped <<webModule>>
UseCaseModel	Model	package stereotyped <<useCaseModel>>
UseCasePackage	Package	package stereotyped <<useCasePackage>>
FlowModel	Model	package stereotyped <<flowModel>>
FlowPackage	Package	package stereotyped <<flowPackage>>
JSP	Class	class stereotyped <<jsp>> or an icon: 
Form	Class	class stereotyped <<form>> or an icon: 
Action	Class	class stereotyped <<action>> or an icon: 
StructureModel	Model	package stereotyped <<structureModel>>
StructurePackage	Package	package stereotyped <<structurePackage>>
PresentationModel	Model	package stereotyped <<presentationModel>>
PresentationPackage	Package	package stereotyped <<presentationPackage>>
Template	Class	class stereotyped <<template>>
TemplateItem	Class	class stereotyped <<templateItem>>

WebApplicationModel. The characteristic of common web applications is that they comprises the set of typical functional modules, like information module (public), intranet module (private), Content Management System (CMS), internet shop module, etc. The web application model presents the web application as the set of such functionally independent subsystems. It is the most general view of the application. The model comprises WebApplication package, which in turn comprises WebModule packages corresponding to application subsystems. The dependencies between WebModules and packages of corresponding use cases are indicated as relationships of refinement.

WebApplication. The highest level package in the web application model. It groups functionally independent modules of the web application.

WebModule. The package that represents the functionally complete module of web application. The set of functions it realizes is gathered in the corresponding use cases package. The dependency between WebModule and the corresponding use case package is indicated as relationship of refinement.

Table 2. Tagged values for WebApplication and WebModule packages

{ImplName : String}	WebApplication and WebModule packages may have some implementation-specific names. Such names should be in the form of strings without any whitespaces or non-ASCII letters. If the name is not stated explicitly, it is assumed that it is the same as the name of the appropriate package.
---------------------	--

Use Case Model Elements.

UseCaseModel. The use case model presents the functions of a web application available for its users. The elements of this model, their notation and semantics are invariable in relation to [2]. Additionally, in the model the relationships between use cases and packages from the flow model corresponding to them are indicated. It is suggested to make use of the rules of specifying use cases described thoroughly in [5].

UseCasePackage. The use case package groups use cases with respect to the web application module realizing them.

Flow Model Elements.

Table 3. Tagged values for the UseCasePackage

{ImplName : String}	UseCasePackage may have some implementation-specific name. Such name should be in the form of a string without any whitespaces or non-ASCII letters. If the name is not stated explicitly, it is assumed that it is the same as the name of the appropriate package.
---------------------	--

FlowModel. The flow model presents all possible navigation paths and, consequently, permissible flows of control between the sections of the web application. To achieve this, it makes use of the collaboration diagram.

For the sake of dynamic generation of part of WWW pages, it is required to present some specific activities, in Jakarta Struts Framework called **actions**, which support the generation process. Their main roles are to collect the necessary data from the model layer to be presented on the web page and to steer the flow of control. The actions are consistent with Jakarta Struts Framework sets of Java classes. Their structure is presented in the structure model.

In Struts-based web applications users navigate through JSP web pages (`<<jsp>>`). Such pages are generated on the basis of templates, described in the presentation model. Among JSP pages the pages containing web forms (`<<form>>`) are distinguished due to their specific treatment in Jakarta Struts Framework.

FlowPackage. The flow package presents the navigation paths that exist between web pages of the particular use case.

JSP. The class that represents a JSP web page in the collaboration diagram. As Jakarta Struts Framework allows to build JSP pages based on generic templates (*tiles* library), hence the presentation model is introduced (section 4.3.5). Such model makes use of notions of a template and a template item to describe the structure and contents of JSP pages.

Form. The class that represents a JSP web page containing a web form in the collaboration diagram. Such a web page builds on template in the same way as other JSP pages. JSP pages containing web forms are distinguished among all JSP pages due to their input role for the web application (they are gathering data from the users); except for them, all other web pages has output role (they are presenting data).

Action. The class that represents a fragment of web application business logic. It is responsible for the display of a proper JSP web page. Typically, actions gather data from the model layer and control flow of control accordingly to specific conditions (e.g. whether the user is authorized to access a particular web page or not). Their structure, in terms of Java classes, is presented in the structure model.

Table 4. Tagged values for the FlowPackage

{ImplName : String}	FlowPackage as well as JSP, Form and Action classes may have some implementation-specific names. Such names should be in the form of a string without any whitespaces or non-ASCII letters. If the name is not stated explicitly, it is assumed that it is the same as the name of the appropriate package or class.
---------------------	--

Structure Model Elements.

StructureModel. The structure model presents the static aspect of web application business logic, which is responsible for the display of a proper JSP web page. It demonstrates the relationships between Java classes constituting the action. At present there are no stereotypes introduced among the set of Java classes, even if they might be categorized as navigation controllers, data wrappers, data collectors, etc. However, their roles are described with specific prefixes added to their names.

StructurePackage. The structure package groups fragments of web application business logic, which are responsible for the display of proper JSP web pages. In other words, it comprises Java classes realizing the particular action.

Table 5. Tagged values for the StructurePackage

{ImplName : String}	StructurePackage may have some implementation-specific name. Such name should be in the form of a string without any whitespaces or non-ASCII letters. If the name is not stated explicitly, it is assumed that it is the same as the name of the appropriate package or class.
---------------------	---

Presentation Model Elements.

PresentationModel. The presentation model presents the inner structure of JSP pages using templates. This approach is supported by the tiles library, part of Jakarta Struts Framework. JSP pages usually comprises few conceptual blocks (template items), like header, footer, body, menu, etc. Some of them are invariable for all web pages of the web application (eg. a footer). Templates of web pages define where and which of template items should be placed in order to compile a complete web page. It allows to avoid unnecessary redundancy of code and makes the JSP code more manageable.

PresentationPackage. The presentation package groups templates and template items, which together compile a complete web pages.

Template. The class that represents the template of a JSP page in the presentation diagram. In Jakarta Struts Framework, the template is a part of configuration file. It states precisely where and which of template items should be placed for a particular web page. Additionally, there is always a generic JSP file, usually the same for all templates of web pages from the same web module, defining the structure of such web pages.

TemplateItem. The class that represents a fragment of JSP web page. It is used to fulfill templates. In practice, a template item is a piece of JSP code written in JSP file.

Table 6. Tagged values for the PresentationPackage

{ImplName : String}	PresentationPackage as well as Template and TemplateItem classes may have some implementation-specific names. Such names should be in the form of a string without any whitespaces or non-ASCII letters. If the name is not stated explicitly, it is assumed that it is the same as the name of the appropriate package or class.
---------------------	---

5 An Application Example

The elaborated extension will be presented on the basis of an analytic and a design documentation of a real web application. The application was realized by the authors by means of Jakarta Struts, Java and JSP.

Application examples of each of introduced models are given below. the authors start from the analysis stage specifying functional modules and their functions. The last presented stage is the design stage where relation between actions and web pages are given To make the presentation more clear the diagrams were simplified.

At the beginning of the analysis stage functional modules of the application were distinguished. There were:

- a public internet module – its content was available for all of interested users,
- a private intranet module – available only for logged in users,
- a content management module.

Each of the distinguished modules is described by a corresponding use case diagram. A simplified diagram demonstrates Fig. 2. Intranet functions (browsing meetings calendar and bulletin board) available for a user are presented. Realization of the functions is described by corresponding flows diagrams. Use case

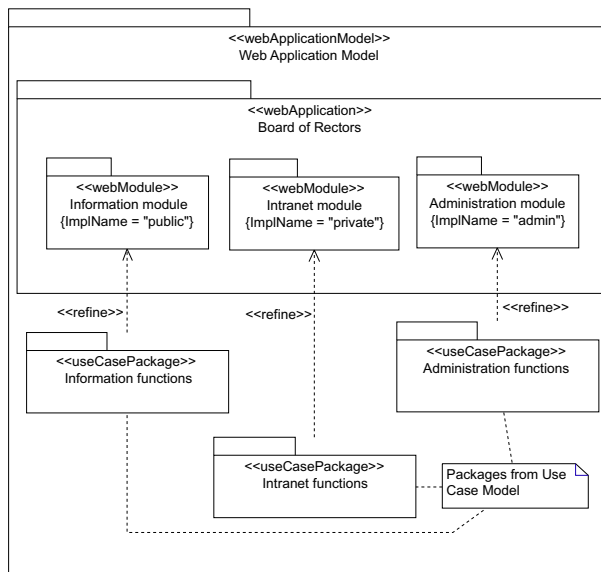


Fig. 1. Functional modules (subsystems) of the web application

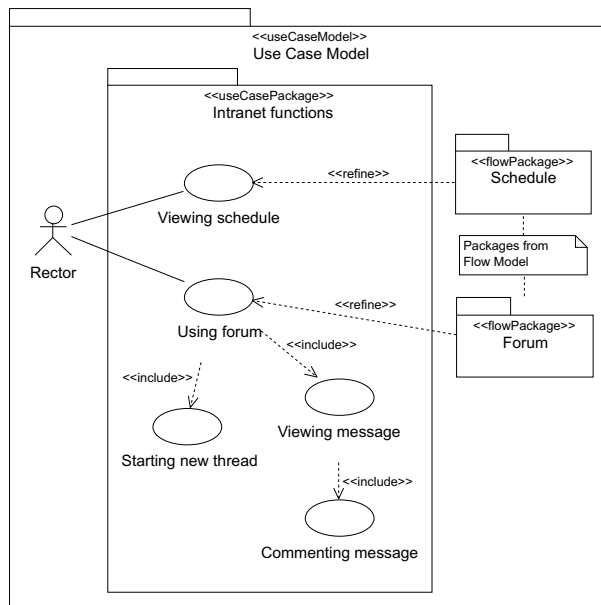


Fig. 2. A part of the use case diagram for intranet functions

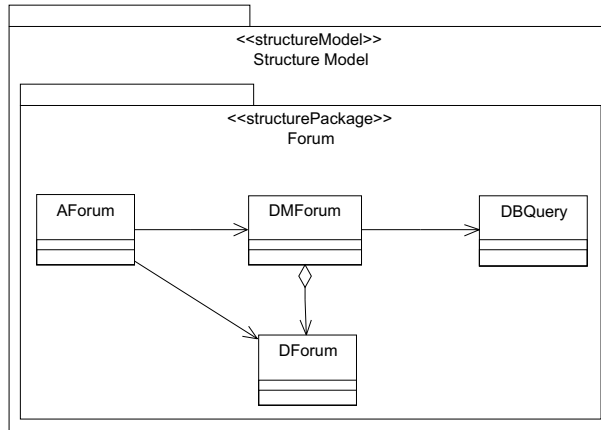


Fig. 3. A part of the structure diagram – an action that prepares JSP page of the bulletin board

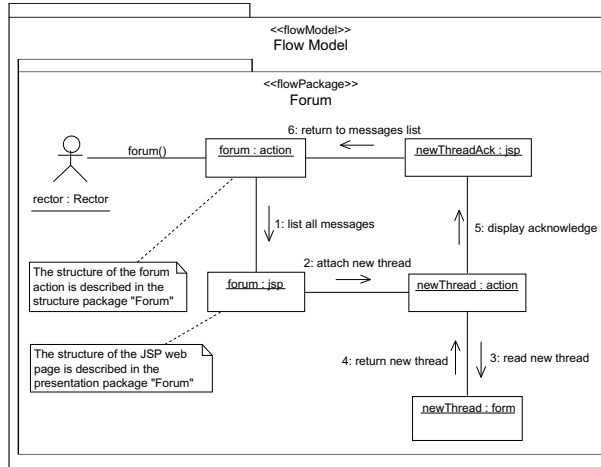


Fig. 4. A part of the flow diagram (bulletin board function)

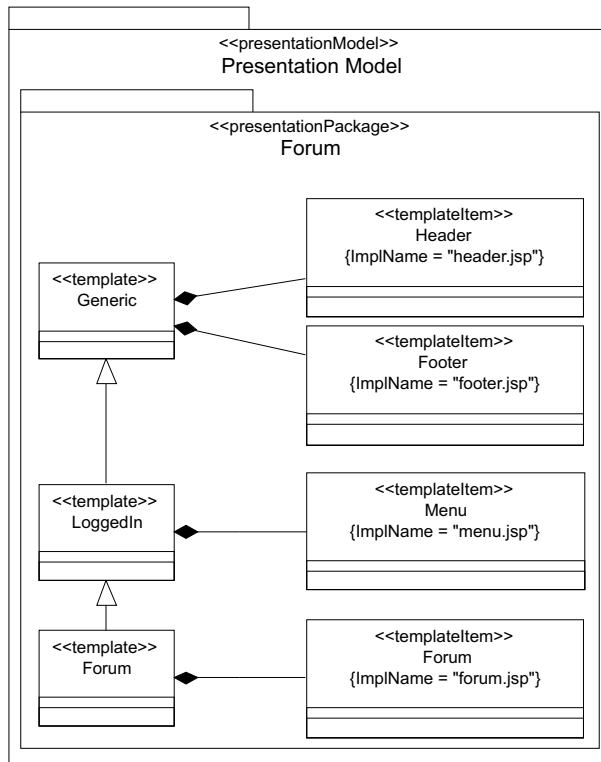


Fig. 5. A part of the presentation diagram (a page with the bulletin board posts list

realization is given by the collaboration diagram (Fig. 3). The diagram is highly simplified. It does not demonstrate, for example, a browsing action or sending post comments.

The structure of each action that appear on the flow diagram is described by the class diagram. Example for an action *bulletin board* is given on Fig. 4. A typical realization of the action is done by four classes: a navigation control class, data management class, a wrapper class and a helper one for data base querying. Prefixes A, DM and D are used to stress a role of the class. The authors abandoned specified stereotypes to simplify diagrams.

JSP pages are build upon templates basing on Tiles library that is a part of Jakarta Struts framework. Templates are in fact JSP files that include appropriate directives for JSP/HTML code enclosure. Templates may be filled gradually therefore the inheritance relation is presented on Fig. 5.

6 Summary

In the paper the UML profile supporting development process of Jakarta Struts Framework based web applications has been presented. Together with the profile, five models describing different stages and viewpoints of analysis and design of Struts-based applications have been introduced. In order to prove the usefulness of abovementioned approach, some fragments of a real life web application have been included. The future improvements to the profile and models are envisaged. The introduction of new models, presenting data-related aspects of a web application, like data flow model or data structure model, is very probable.

References

1. Conallen J.: Building Web Applications with UML. Second Edition. Addison Wesley, Oct 2002
2. OMG UML Specification Version 1.5. Mar, 2003.
3. Husted T., Dumoulin C., Franciscus G., Winterfeldt D.: Struts in Action Building web applications with the leading Java framework, Manning Publications, 2003
4. Booch G., Rumbaugh J., Jacobson I.:UML Przewodnik użytkownika, WNT, 2001
5. Cockburn A.: Jak pisać efektywne przypadki użycia, WNT, 2004