

A DIAGRAM APPROACH TO AUTOMATIC GENERATION OF JSP/SERVLET WEB APPLICATIONS

Kornkamol Jamroendararasame, Tetsuya Suzuki and Takehiro Tokuda
Department of Computer Science
Tokyo Institute of Technology
Tokyo 152-8552, Japan
email: {konkamol,tetsuya,tokuda}@tt.cs.titech.ac.jp

ABSTRACT

We defined diagrams called Web transition diagrams to represent overall behavior of Web applications. Using these diagrams, we can generate server program type Web applications such as CGI-based Web applications, and server page type Web applications such as ASP-based Web applications.

The purpose of this paper is to design Web transition diagrams to represent wider class of Web applications based on JavaServer Pages (JSP)/Servlet architecture. Then we present an automatic generation method of JSP/Servlet Web applications from these diagrams. We implement a software system called T-Web system which enables Web application designers to visually compose Web transition diagrams and generate Web applications from these diagrams without any manual programming. T-Web system can generate both HTML-based Web pages and XML-based Web pages for Web applications. Generated Web applications support standard level of security against attacks, the use of HTTP cookies, user authentication, and session management.

KEY WORDS

Web application, JSP, Java servlet

1 Introduction

Nowadays, the use of Web applications for data transaction has been immensely increasing. Because of the sudden growth of Web application technologies, the development of Web applications becomes more and more difficult not only for non-programmers but also for experienced programmers.

Even in the development of small size Web applications, we have to consider the consistency among application components, security against attacks and methods to protect Web users' personal information.

We defined Web transition diagrams to describe overall behavior of Web applications and an automatic generation method of Web applications from these diagrams. Then we implemented a CGI-based Web application generator [1, 2] and an ASP-based Web application generator [3] using a method called template method. However, these

generators have following restrictions.

- Newly created templates cannot be treated as predefined templates.
- They can generate either HTML-based Web pages [4] or XML-based Web pages [5], but not both.
- They cannot generate advanced structural Web pages.

In this paper, we propose the following enhancement.

1. Web transition diagrams that can represent wider class of Web applications based on JSP/Servlet architecture. Integration of server program type technology servlet [6] and server page type technology JSP [7] is possible.
2. A JSP/Servlet-based Web application generator which accepts any number of additional templates for Web page elements or programs. Generated Web applications support standard level of security against attacks, the use of HTTP cookies, user authentication, and session management.

We implement a software system called T-Web system. Using T-Web system, Web application designers including non-programmers can design JSP/Servlet-based Web applications in terms of Web transition diagrams and generate Web applications based on both HTML and XML pages from the complete diagrams.

The rest of this paper is organized as follows. In section 2, we explain Web transition diagrams. In section 3, we describe and illustrate our JSP/Servlet-based Web application generator. In section 4, we compare our approach with other approaches. In section 5, we give concluding remarks.

2 Web Transition Diagrams

This section presents the definition, notation and an example of Web transition diagrams. Web transition diagrams are directed graphs that can describe structures and overall behavior of general Web applications. They represent both the flow of data through Web application components (e.g. Web pages, programs, and databases) and hyperlink

relations between Web pages. Notation of Web transition diagram components is shown in fig.1. Basic components which always appear in Web transition diagrams are in the core class. Additional components which are sometimes necessary in Web transition diagrams are in the extended class. In this paper, we mainly define Web transition diagrams for JSP/Servlet-based Web applications. However, they can also represent Web applications based on other architectures such as CGI architecture.

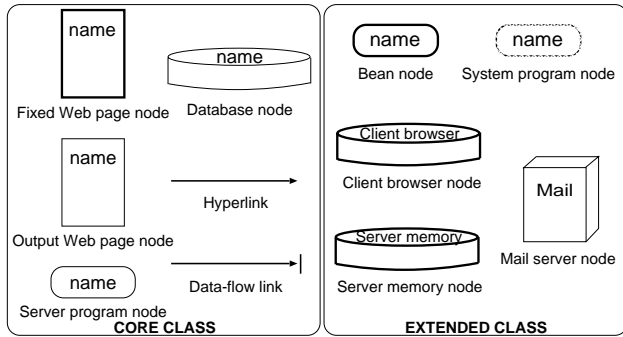


Figure 1. Nodes and links in Web transition diagrams

According to fig.1, a fixed Web page node represents a Web page stored in a file system and accessible by a certain URI. An output Web page node represents a Web page generated by a server program (a Java servlet in JSP/Servlet architecture). A server program node represents a Java servlet. A database node represents a database table. A hyperlink represents a hyperlink relation between Web pages. A data-flow link represents a flow of data between components. A bean node represents a Java bean. A system program node represents any system process such as frame contents presentation. A client browser node represents a client browser used to store HTTP cookies. A server memory node represents a server memory space used to store session objects. A mail server node represents a mail server for outgoing mails.

Fig.2 shows examples of relations between components in Web applications represented by Web transition diagrams. They can be described as follows.

1. **Multiple-frame Web page:** a representation of a multiple-frame Web page and corresponding frame contents
2. **Java servlet:** a representation of relations between a servlet and Web pages, a database table, a client browser, or a server memory space
3. **Java servlet-Mail server:** a representation of a servlet sending an email message to a mail server
4. **Java bean-Web page:** a representation of JSP codes setting or/and getting a bean property

An example of Web transition diagrams is shown in fig.3. The diagram describes overall behavior of a meeting

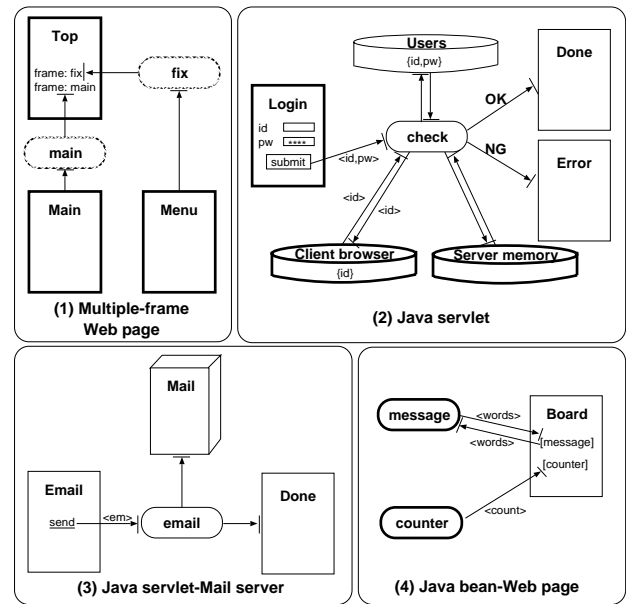


Figure 2. Examples of relations between Web application components represented by Web transition diagrams

room booking system. The system allows Web users to register as new members, login to the system to reserve or cancel their reservations, and request for their forgotten passwords.

3 Generation of JSP/Servlet Web Applications

We implement a software system, called T-Web system, to generate JSP/Servlet Web application components from predefined templates. We present its method in this section. Fig.4 shows the structure, processes, input and output of T-Web system.

T-Web system consists of two parts: a Web transition diagram editor and a Web application generator. Web application designers compose a Web transition diagram on the Web transition diagram editor. The Web application generator generates a Web application from the complete Web transition diagram and predefined templates without any procedural programming by the designers. The editor, the generator, templates and template method are described respectively in the following sections.

3.1 Web Transition Diagram Editor

According to the notation of Web transition diagrams, concepts used in the Web transition diagram editor (illustrated in fig. 5) are as follows.

1. Excluding system program nodes, each of nodes in Web transition diagrams realizes a set of parameters and its properties which is a union of a set of input

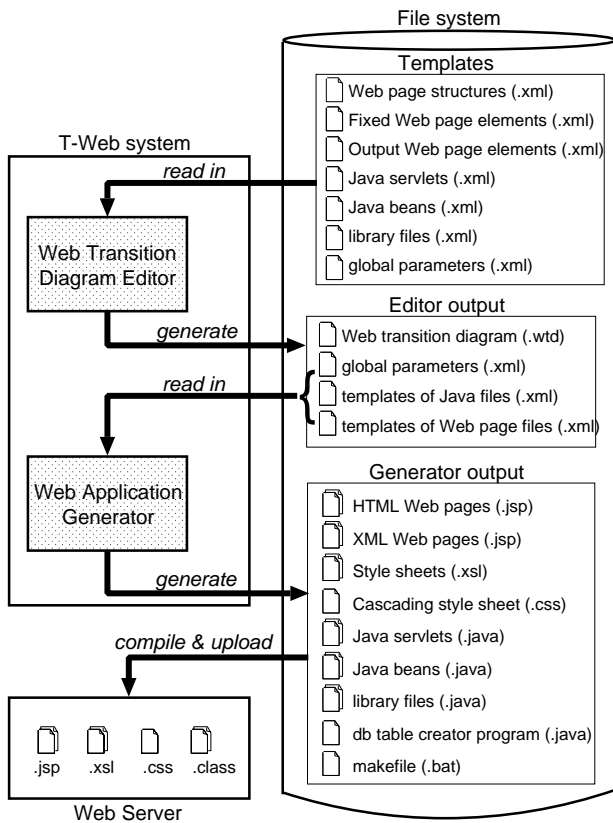


Figure 4. The structure and process of T-Web system

all nodes, specifying their names, and then specifying details of database nodes, bean nodes, fixed Web page nodes, server program nodes and output Web page nodes respectively. A Web transition diagram is completed when all nodes have names, necessary details and relationships with other nodes.

The output results of the editor are a Web transition diagram, a file containing global parameters, a file containing templates of all Java files and a file containing templates of all Web pages as shown in fig. 4.

3.2 Web Application Generator

As shown in fig. 4, Web application generator reads in a file containing global parameters, a file containing templates of all Java files and a file containing templates of all Web pages from the file system. To generate resulting files, the generator replaces values of template parameters into templates using following method. In templates, parameter names are distinguished by inserting them between the combination of slashes (/) and asterisks (*), namely `/*PARAMETER_NAME*/` or `/**PARAMETER_NAME**/`. A text from `'/*'` to `'*/'` or from `'/**'` to `'**/'` is replaced by a text created using the following rules.

1. Any character between `'/*'` and `'*/'` or between

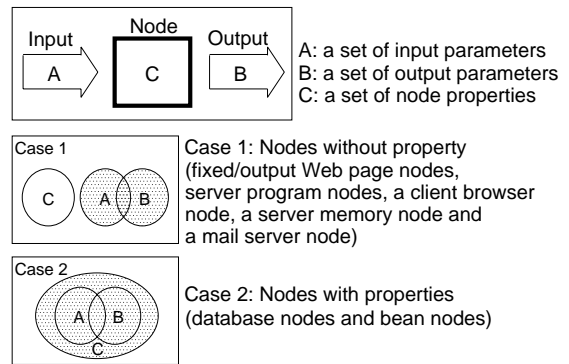


Figure 5. Concepts used in the Web transition diagram editor

`'/**'` and `'**/'` which is not a part of parameter name must be attached to each parameter value at its original position. (e.g. `/*JSPDIRECTORY/*'` becomes `'/shopping/'` if the value of the parameter JSPDIRECTORY is 'shopping'.)

2. If a parameter value does not exist, the whole text must be replaced by a null character.
3. If a parameter has multiple values, there are two possible cases.

(a) If the parameter name is enclosed by `'/*'` and `'*/'`, each value after attached by non-parameter characters must be separated by a comma. (e.g. `/*"COLUMNLABEL"*/'` becomes `'"Item ID", "Item Name", "Unit Price", "Order Number"'` if values of the parameter COLUMNLABEL are 'Item ID', 'Item Name', 'Unit Price' and 'Order Number'.)

(b) If the parameter name is enclosed by `'/**'` and `'**/'`, each value after attached by non-parameter characters must be separated by a new-line character. (e.g. `'/**String PARAM = " ";**/'` becomes `'String id = " "; String name = " "; String price = " ;'` if values of the parameter PARAM are 'id', 'name' and 'price'.)

Output files from the generator are as follows.

1. **HTML Web pages (.jsp)** are JSP pages in HTML format. One file is generated from each Web page node.
2. **XML Web pages (.jsp)** are JSP pages in XML format. One file is generated from each Web page node which contains database information.
3. **Style sheets (.xsl)** are style sheet files for Web pages in XML format. One file is generated for each JSP page in XML format.

4. A **cascading style sheet (.css)** is a cascading style sheet file. One file is generated for each Web application.
5. **Java servlets (.java)** are Java servlet files. One file is generated from each server program node.
6. **Java beans (.java)** are Java bean files. One file is generated from each bean node.
7. **Library files (.java)** are library files required for Web applications generated by T-Web system. One file is generated from each library file template.
8. A **database table creator program (.java)** is a source file of a Java program that can create all database tables used in the Web application.
9. A **makefile (.bat)** is a batch file containing commands to compile all Java files.

From output of the generator, the following steps must be done before the Web application can be launched.

1. The makefile must be run to compile all Java files.
2. The environment on the Web server must be set for JSP/Servlet-based Web applications.
3. Either the set of HTML-based Web pages or the set of XML-based Web pages must be uploaded to the appropriate directory on the Web server.
4. Java servlets, Java beans and library files must be uploaded to the appropriate directories on the Web server.
5. Database system must be connected to the Web server by network and database tables must be created. A database table creator program may be run to create all database tables automatically.

3.3 Templates and Template Method

As shown in the fig.4, templates must be predefined for the structure of Web pages, elements in Web pages, server programs (Java servlets), Java beans, library files and a database table creator program. All template files are written in XML format in which Web transition diagram editor can systematically read. When T-Web system is run, the editor firstly reads templates from the file system and provides lists of templates to be selected when designers compose Web transition diagrams. For each Web application component, designers are supposed to select an appropriate template from the lists and specify values to template parameters.

The set of predefined templates may not be expressive enough. The reason is that there might not be any template suitable for some processes, and providing a large number of templates may cause a difficulty of template selection. However, T-Web system has been designed for unlimited

number of templates. It accepts new templates for Web page elements, JSP codes, Java servlets and Java beans composed by programmers according to some simple rules of template parameter usage as mentioned in the previous section. Examples of templates predefined in T-Web system for Java servlets are as follows.

1. **insert if unique:** The servlet inserts a set of parameters into a database table only if all parameters are unique.
2. **insert if unavailable:** The servlet inserts a set of parameters into a database table only if similar parameter values do not exist in any record in the database.
3. **insert in any case:** The servlet inserts a set of parameters into a database table without any condition.
4. **insert or replace:** The servlet inserts a set of parameters into a database table or replace on existing records if they satisfy specified condition.
5. **match:** The servlet compares values of parameters with the existing values in the database table.
6. **show all info:** The servlet retrieved all available database information.
7. **show some info:** The servlet retrieved database information that satisfies specified condition.
8. **cancel:** The servlet cancels current values of database information that satisfies specified condition.
9. **update:** The servlet updates values of database information that satisfies specified condition to be current values of parameters.
10. **delete:** The servlet deletes the record that satisfy specified condition from the database table.
11. **sendmail:** The servlet sends an email message to the specified recipient through a mail server.

For Web page elements and JSP codes, templates for almost all of the basic elements and a number of JSP codes have been predefined such as templates for frames, hyperlinks, Java applets, images, forms, input controls, JSP codes to get current date, JSP codes to get access count, JSP codes to get database information, JSP codes to define a Java bean and JSP codes to set/get a bean property.

4 Comparisons

This section evaluates our Web transition diagrams and T-Web system by comparing them with other approaches.

We may observe that there are many types of diagrams used for design or construction process of Web applications. A site map is an example of diagrams whose purpose is obviously different from that of Web transition diagrams. Site maps are normally used to represent the

structure of Web application contents, while Web transition diagrams are used to represent the structure of Web applications in construction level. However, Web transition diagrams have a site map function such that they can represent hyperlink relations and brief Web page contents.

Using of UML class diagrams to represent ASP-based Web applications is another example [8]. UML class diagrams with extension for Web applications can represent client pages, server pages, interfaces and relationships between pages and interfaces. They are useful for low-level design process and construction process. On the other hand, our Web transition diagrams represent behavior of Web applications in higher level because they are used for automatic generation, not for manual construction.

Currently there are many tools and approaches for Web application development proposed by software vendors or research projects [9]. Here we compare our T-Web system with examples of tools for non-programmers and tools for programmers. Visual Web page editors such as Microsoft's FrontPage [10] and Adobe's GoLive [11] are examples of tools for non-programmers which can construct the whole Web sites without manual programming. These tools are originally proposed for the construction of static Web sites. They are full of features for presentation logic but lack of features for data processing and data transactions with databases. Microsoft's Visual InterDev [12] is an example of tools for programmers which enables us to design and construct Web applications based on ASP architecture. This tool can construct data-intensive Web applications, but requires scripting language programming ability.

T-Web system for JSP/Servlets is a tool for non-programmers. It enables us to construct basic Web page elements and server programs dealing with databases in JSP/Servlet Web applications. It can generate server programs with codes for data processing and database transaction without any programming ability requirement. Visual Web page editors may be used to decorate the Web pages generated by T-Web system.

5 Concluding Remarks

We have presented the use of Web transition diagrams to represent general Web page elements and Web application components in JSP/Servlet-based Web applications. T-Web system has been implemented as a Web application generator for JSP/Servlets. From Web transition diagrams composed in Web transition diagram editor, Web application components based on both HTML and XML can be generated by the Web application generator using predefined templates. Resulting Web applications support standard level of Web security, the use of HTTP cookies, user authentication, and session management. Examples of Web applications that can be generated by T-Web system are booking systems, shopping systems, organizer systems and Webboards.

T-Web system accepts any additional templates for Web page elements, JSP codes, Java servlets and Java

beans newly composed by users. Methods and concepts used in this system may be possible in the generation of Web applications based on other architectures.

References

- [1] K. Jamroendararasame, T. Matsuzaki, T. Suzuki, T. Tokuda, Generation of secure Web applications from Web transition diagrams, Proc. of the IASTED International Symposia Applied Informatics, Innsbruck, Austria, 2001, 496-501.
- [2] K. Jamroendararasame, T. Matsuzaki, T. Suzuki, T. Tokuda, Two generators of secure Web-based transaction systems, Proc. 11th European-Japanese Conference on Information Modelling and Knowledge Bases, Maribor, Slovenia, 2001, 348-362.
- [3] M. Taguchi, T. Susuki, T. Tokuda, Generation of Server Page Type Web Applications from Diagrams, Proc. 12th European-Japanese Conference on Information Modelling and Knowledge Bases, Krippen, Swiss Saxony, 2002, 117-130.
- [4] W3C, HTML 4.01 Specification, <http://www.w3.org/TR/html401/>
- [5] W3C, Extensible Markup Language (XML), <http://www.w3.org/XML/>
- [6] M. Hall, *Core Servlets and JavaServer Pages* (New Jersey:Prentice-Hall,Inc., 2000).
- [7] D. Hougland, A. Tavistock, *Core JSP* (New Jersey: Prentice-Hall,Inc., 2001).
- [8] J. Collanen, *Building Web Application with UML* (USA: Addison-Wesly, 2000).
- [9] P. Fraternali, Tools and approaches for developing data-intensive Web applications:a survey, *ACM Computing Surveys*, 31(3), 1999, 227-263.
- [10] Microsoft Corporation, Microsoft Office - FrontPage Home, <http://www.microsoft.com/frontpage/>
- [11] Adobe Systems Incorporated, Adobe GoLive, <http://www.adobe.com/products/golive/>
- [12] Microsoft Corporation, Microsoft Visual Interdev Home Page, <http://msdn.microsoft.com/vinterdev/>
- [13] K. Jamroendararasame, T. Suzuki, T. Tokuda, JSP/Servlet-based Web application generator, Proc. 18th Japan Society for Software Science and Technology, Hakodate, Japan, 2001, 2C-1.
- [14] T. Tokuda, T. Suzuki, K. Jamroendararasame, S. Hayakawa, A family of Web diagrams approach to the design, construction and evaluation of Web applications, Proc. 12th European-Japanese Conference on Information Modelling and Knowledge Bases, Krippen, Swiss Saxony, 2002, 293-306.