GENERATION OF WEB APPLICATIONS FROM ANNOTATION-BASED DEFINITIONS

Kazuhiro Asami and Takehiro Tokuda
Department of Computer Science
Tokyo Institute of Technology
Meguro, Tokyo 152-8552, Japan
{asami, tokuda}@tt.cs.titech.ac.jp

Abstract Construction of Web applications usually requires much knowledge about protocols, programming languages and databases. We present a new method for generation of Web applications. We first construct HTML page templates for intended Web applications. Then we give annotations to these HTML page templates. Annotations are for session management, input data checking, database handling and communications with external programs. From HTML page templates with annotations, we automatically generate CGI-based Web applications.

Our method is simple but general enough to describe typical Web applications such as guest book systems, room booking systems, shopping cart systems, glossary systems and user registration systems. Without using detailed knowledge of Web programming, anybody who understands HTML, constraint expressions, SQL, SOAP and XSLT can easily develop Web applications.

Keywords: Web Applications, Annotations, Software Generators

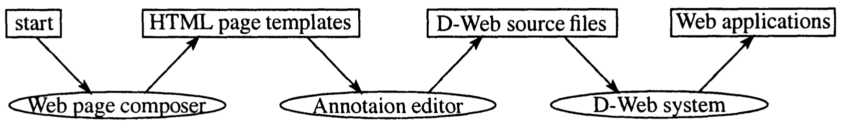
1. Introduction

Construction of Web applications, such as online shopping systems, database query systems, and reservation systems, usually requires detailed knowledge about HTTP, HTML, SQL, CGI, Servlet (1), PHP (2), JSP (1), and ASP (3) as well as procedural programming languages such as Perl, C, Java, and VB-Script. Also we must manually write extra codes for the check of input values and consistency of the session.

We present a new method for generation of Web applications as follows.

1 We first construct HTML page templates for intended Web applications. We give annotations to these HTML page templates. Annotations are

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35614-3_21



1 Create HTML page templates using ordinary Web page composers

2 (a) Annotate HTML page templates to define processing of intended Web applications using our annotation editor
(b) Convert annotated Web page templates to D-Web source files by our annotation editor automatically

3 Generate Web applications by D-Web system automatically

Figure 1. Flow of development of Web applications in our approach

for session management, input data checking, database handling, and communications with external programs.

2 Annotations for session management, input data checking, and database handling enable us to construct most typical Web applications we need everyday. Annotations for communications with external programs enable us to construct other types of Web applications which require existing programs handling complex logic.

3 From HTML page templates with annotations, we automatically generate CGI-based Web applications using a Web application generator called D-Web system (4). Extra codes for security and session management are also generated automatically.

Our flow of development of Web applications is shown in Figure 1.

Our method is simple but general enough to describe typical Web applications such as guest book systems, room booking systems, shopping cart systems, glossary systems, and user registration systems. Without using detailed knowledge about Web programming, anybody who understands HTML, constraint expressions, and SQL or who understands HTML, constraint expressions, SQL, SOAP (5), and XSLT (6) can easily construct these standard Web applications or advanced Web applications respectively.

The organisation of the rest of the paper is as follows. In Section 2, we describe our annotation-based definitions of Web applications. In Section 3, we describe a generation method of Web applications from our definitions. In Section 4, we compare our approach with related work. In Section 5, we give our conclusion.

2. Annotation-based definitions of Web applications

In our approach, HTML page templates are created using Web page composers and then the HTML page templates are annotated using a special annotation editor. We describe HTML page templates and annotations below.

HTML page templates. HTML page templates are HTML pages including special strings which are dynamically replaced by certain values. The HTML page templates are grammatically same as HTML pages. The HTML page templates can be created using ordinary Web page composers. Special strings are strings which begin and end with characters "$" or "#" like "*$variableName$*" and "*#fieldName#*". Special strings which begin and end with "$" are replaced by input values of form controls or developer-defined values. Special strings which begin and end with "#" are replaced by values of fields of tables in databases. Figure 2 shows examples of HTML page templates. This figure shows a user registration system. In this figure, the system is represented by a diagram whose nodes are HTML page templates and whose arrows are transitions between pages. In the page named Top of this system, users input names, PIN numbers, ZIP codes and so on. In the page named Complement, this system complements user's prefecture, city and town using an external program of ZIP code search and users complete their addresses. In the page named Register, this system register users' informations to a database table. If users' input strings are wrong, the page named Error is shown in their Web browsers.

Annotations. In our approach, annotations are definitions of processing of Web applications. We introduce five types of annotations: session, input check, constraint, SQL, and SOAP. Table 1 shows the meanings of each annotation. To annotate HTML page templates, we use a special annotation editor. Our annotation editor is implemented as one of Web applications. To use our annotation editor, we upload HTML page templates to our annotation editor. Our annotation editor embeds hyperlinks in each HTML page template to Web pages where annotations are defined. We annotate HTML page templates by following the embedded hyperlinks and by inputting items of annotations. Figure 3a and Figure 3a show screen shots of our annotation editor. Figure 3a shows the HTML page template named Top in which hyperlinks are embedded. Figure 3a shows the Web page of an input check annotation for the input field of PIN numbers. For our user registration system, we use a number of annotations to define processing. In Top page, we have a session annotation, seven input check annotations for each input field and a constraint annotation whose expression is "$\$pin\$$ eq $\$reenterPin\$$" for checking sameness of PIN

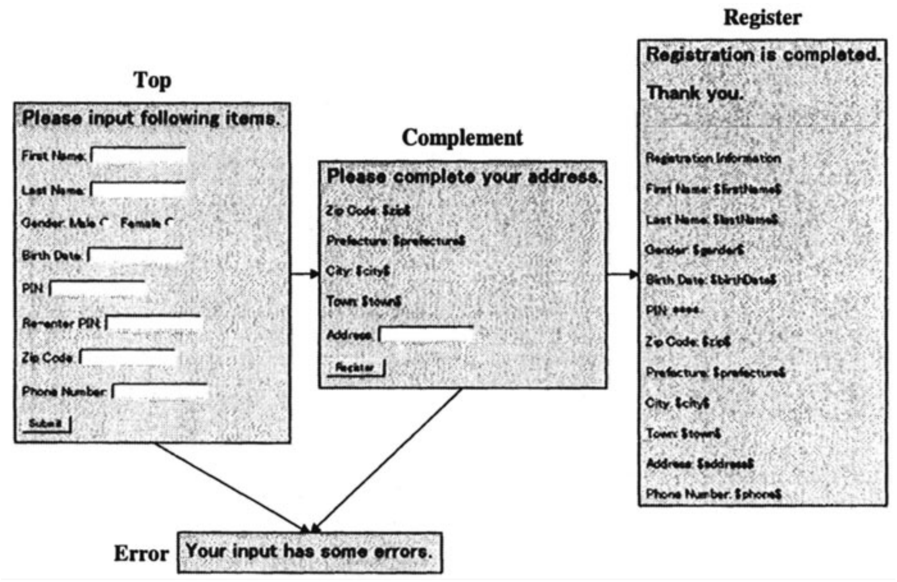
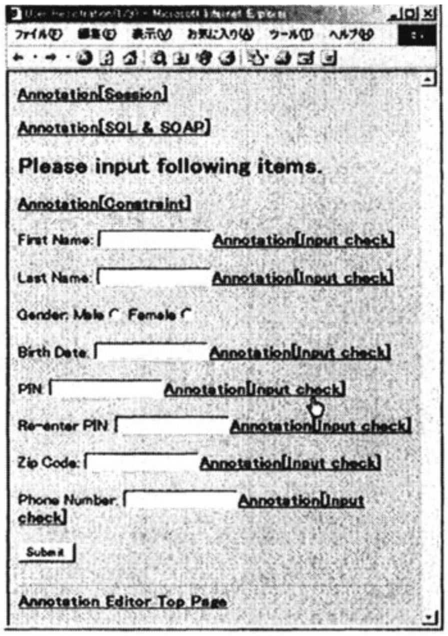


Figure 2. HTML page templates of a user registration system

Table 1. Meanings of annotations

Annotation	Meaning
Session	Session management.
Input check	Definitions of valid input strings sent from form controls using constraints on length of input strings and regular expressions including repeting operators such as +, * and ?, selecting operator(|) and character class operator([ ]).
Constraint	Relations among input values sent from form controls. Relations are represented by Boolean expressions including a number of arithmetic operators, logical operators, comparative operators and predicates regarding database tables.
SQL	Queries to database tables using SQL.
SOAP	Communications with external programs representing business logic using SOAP envelopes and transformations of results into HTML using XPATH (6) or XSLT.

and Re-enter PIN. In this case. In Complement page, we have a session annotation, an input check annotation and a communication annotation for search of ZIP codes. In Register page, we have a session annotation and an SQL annotation to register user informations to a database table using an INSERT statement of SQL. In Error page, we have a session annotation.


ファイル(F) 編集(E) 表示(V) お気に入り(A) ツール(T) ヘルプ(H)

Annotation[Session]

Annotation[SQL & SOAP]

Please input following items.

Annotation[Constraint]

First Name: Annotation[Input check]

Last Name: Annotation[Input check]

Gender: Male Female

Birth Date: Annotation[Input check]

PIN Annotation[Input check]

Re-enter PIN Annotation[Input check]

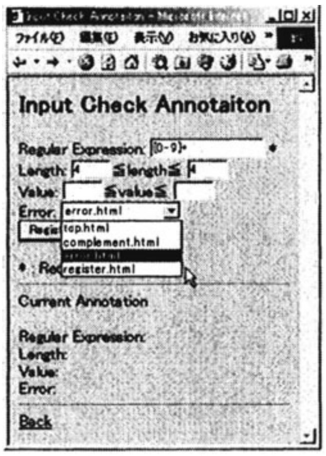
Zip Code: Annotation[Input check]

Phone Number: Annotation[Input check]

Submit

Annotation Editor Top Page

Figure 3a. Screen shot of our annotation editor: HTML page template with embedded hyperlinks



Input Check Annotaiton

Regular Expression: [0-9]* ◆
Length: 4 ≦length≦ 4
Value: ≦value≦
Error: error.html
top.html
complement.html
register.html
Regis
◆ : Rec

Current Annotation

Regular Expression:
Length:
Value:
Error:

Back

Figure 3b. Screen shot of our annotation editor: Web page for input check annotations

3. Generation of Web applications

We present how to generate Web applications from our annotation-based definitions. Our generation method of Web applications consists of the following two steps:

1 Conversion of annotated HTML pages to source files for a Web application generator system called D-Web system

2 Generation of Web applications from D-Web source files

D-Web system is a Web application generator system which generates Web applications from definitions consisting of HTML, constraints, SQL, and communications with external programs. Views and user interfaces of Web applications are defined using HTML. Constraints define validness of values which users input. SQL statements define transactions with databases. Communications with external programs are defined using SOAP envelopes. Using D-Web system, Web applications are easily generated. The current version of D-Web system generates HTML template files and CGI programs written in

Perl. D-Web system can generate secure Web applications in the meaning that users' input values are checked. Without checks of input values, Web applications may be downed by malicious users (7). D-Web system can automatically generate codes for session management and consistency management. In D-Web system, consistency management is to avoid mismatches of HTML pages shown in users' Web browsers and states of sessions. These mismatches may occur when users make use of Web browsers' back buttons. D-Web source files, input of D-Web system, are HTML files including a number of extended tags. Session management, constraints, SQL statements, and communications with external programs are defined by the extended tags. We show an example of D-Web source files in Figure 4.

We describe the two steps of generation in Section 3.1 and Section 3.2.

3.1. Conversion of annotated HTML pages to D-Web source files

Annotated HTML page templates need to be converted to use D-Web system for generation of Web applications. The conversion is done by our annotation editor using the following rules:

1 Session annotations
Session annotations are converted into D-Web extended tags <session>.

2 Input check annotations
Input check annotations are converted into D-Web extended attributes "domain" and "error", and the attributes are appended to input fields which the annotations correspond to.

3 Constraint annotations
Constraint annotations are converted into D-Web extended tags <constraint>. The form elements which the annotations correspond to include the tags <constraint>.

4 SQL annotations
If SQL statements are SELECT statements, SQL annotations are converted into both of D-Web extended tags, <sql> and <sqlfor>. Otherwise, annotations are converted into D-Web extended tags <sql>. Extended tags <sql> define SQL statements. Extended tags <sqlfor> define templates to list up all rows of the results of database queries.

5 SOAP annotations
The annotations are converted into D-Web extended tags <soap> and <set>. Extended tags <soap> define SOAP envelopes and XSLT. Extended tags <set> define mappings of SOAP results and special strings.

We show an example of D-Web source files in Figure 4. This source file is generated from the annotated HTML page template named Complement. Extended tags added by our annotation editor are indicated using boldface. Figure 5 shows an example of SOAP envelopes. This SOAP envelope represents search of addresses from ZIP codes. This SOAP envelope is referred by the extended tag <soap> in the annotated HTML page template named Complement.

```
<html>
  <head>
    <title>User Registration(2/3)</title>
    <session type=”check” />
  </head>
  <body>
    <soap envelope=”getZIPInfoByZipcode.soap”
        url=”http://hostname.domain/soap/servlet/rpcrouter”>
      <set name=”prefecture”
        xpath=”/Envelope/Body/getZIPInfoByZipcodeResponse/return/item/prefname” />
      <set name=”city”
        xpath=”/Envelope/Body/getZIPInfoByZipcodeResponse/return/item/cityname” />
      <set name=”town”
        xpath=”/Envelope/Body/getZIPInfoByZipcodeResponse/return/item/townname” />
    </soap>
    <h2>Please complete your address.</h2>
    <form action=”register.html”>
      <p>Zip Code: $zip$<input type=”hidden” name=”zip” value=”$zip$” /></p>
      <p>Prefecture: $prefecture$
        <input type=”hidden” name=”prefecture” value=”$prefecture$” /></p>
      <p>City: $city$<input type=”hidden” name=”city” value=”$city$” /></p>
      <p>Town: $town$<input type=”hidden” name=”town” value=”$town$” /></p>
      <p>Address: <input type=”text” name=”address”
        domain=”/.*/$$.length[1, 100]” error=”error.html” /></p>
      <p><input type=”submit” value=”Register” />
        <input type=”hidden” name=”firstName” value=”$firstName$” />
        <input type=”hidden” name=”lastName” value=”$lastName$” />
        <input type=”hidden” name=”gender” value=”$gender$” />
        <input type=”hidden” name=”birthDate” value=”$birthDate$” />
        <input type=”hidden” name=”pin” value=”$pin$” />
        <input type=”hidden” name=”zip” value=”$zip$” />
        <input type=”hidden” name=”prefecture” value=”$prefecture$” />
        <input type=”hidden” name=”city” value=”$city$” />
        <input type=”hidden” name=”town” value=”$town$” />
        <input type=”hidden” name=”phone” value=”$phone$” />
      </p>
    </form>
  </body>
</html>
```

Figure 4. Example of D-Web source files

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <ns1:getZIPInfoByZipcode xmlns:ns1="urn:zipsearch-service"
        SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <zipcode xsi:type="xsd:string">1528552</zipcode>
    </ns1:getZIPInfoByZipcode>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 5. Example of SOAP envelopes(getZIPInfoByZipcode.soap)

3.2. Generation of Web applications from D-Web source files

D-Web system generates Web applications from HTML files including a number of D-Web extended tags. CGI programs generated by D-Web system consist of the following four parts in the order of execution:

1 Check that constraints are satisfied

2 Manage sessions and consistency

3 Execute SQL statements and Communicate with external programs.

4 Output HTML pages

Check that constraints are satisfied. Codes for checking that constraints are satisfied are generated using specified constraint expressions. For all specified constraint expressions, D-Web system generates codes as follows:

```
if (!(specified_constraint_expression)) {
    goto output_specified_error_page;
}
```

Manage sessions and consistency. CGI programs generated by D-Web system manage sessions and consistency using hidden controls, input elements whose attributes "type" are "hidden". To manage sessions, we use session IDs to identify which session HTTP requests belong to. D-Web system generates codes for session management as follows:

1 Add hidden controls whose names are "_sessionId" to form elements. Generate codes to embed each user's session ID to the added hidden controls.

2 Generate codes for creation of unique session IDs and management of the created session IDs using a database table for session management.

3 Generate codes to check users have valid session IDs.

To manage consistency, reexecution of CGI programs from HTML pages before the last side effects to databases occur is avoided. We give unique numbers to all output HTML pages and embed the numbers in each output HTML page using hidden controls. Pairs of the numbers where the last side effects occur and session IDs are stored in a database table. By comparison of the numbers sent from Web browsers and the numbers in the database table, we can detect inconsistency.

Execute SQL statements. We can generate codes for execution of SQL statements using database drivers. Generated codes written in Perl using a database connection module DBI are as follows:

```
use DBI;
$dbh = DBI−>connect("dbi:$databaseType:"
  . "host=$host "
  . "dbname=$dbname", $userName);
$sth = $dbh−>prepare($sqlStatement);
$result = $sth−>execute();
$dbh−>disconnect();
```

Communicate with external programs. D-Web system are given SOAP envelopes from definitions of Web applications. We can communicate with external programs by sending the SOAP envelopes just as they are to SOAP servers.

Output HTML pages. Output HTML pages are defined by HTML page templates. Output HTML pages are quite same as HTML page templates except for special strings and templates to list the results of SELECT statements. Special strings are replaced by the actual values. Templates to list the results of SELECT statements are implemented using "for" statements as follows:

```
for ($i = 0; $i < $result− >ntuples; $i++) {
  $replaced_template =
    replace_special_string_to_actual_values(
      $template, $result− >get_row($i));
  output($replaced_template);
}
```

4. Comparison

Among many approaches to Web application construction, we give comparison with four approaches.

A method for development of Web applications is to write processing programs manually (with certain level of automated facilities) and to create HTML pages using Web page composers. To write processing programs, developers need detailed knowledge about programming languages and Web programming. In our approach, we need only five types of knowledge at most. For construction of most typical Web applications, we need only three types of knowledge.

OpenZOLAR (8) is a system which interprets and executes extended HTML files as a CGI program. The extended HTML called ZHTML includes a number of extended tags to query databases and to evaluate expressions. ZHTML does not provide easy methods to manage sessions and to check input values. In session management, developers have to implement it using a number of the extended tags. In checking input values, developers have to implement it as queries to databases which handle matching regular expressions. If available databases do not handle regular expressions, input values can not be checked. On the other hand, our system has annotations to manage sessions and to check input values using constraints. Using our system, we can develop Web applications which manage sessions and check input values easily. In defining Web applications, OpenZOLAR has no visual tool to create ZHTML files. The extended tags of ZHTML are embedded in HTML files using ordinary text editors. In our approach, we can visually define processing of Web applications using our annotation editor.

Microsoft FrontPage (9) is a tool to develop Web applications. FrontPage handles Web page composition, form field validation and transaction with databases. FrontPage allows us to define valid strings to input in each form field by specifying types of characters such as alphabets, digits or symbols. Form fields may require strings in certain formats such as phone numbers or E-mail addresses. Using FrontPage, it is difficult to define Web applications which validate these strings. On the other hand, our input check annotations are defined using regular expressions. We can easily define Web applications which validate these strings. FrontPage can generate Web applications which transact with databases. Web applications generated by FrontPage need special programs called FrontPage Server Extensions on Web servers. Web applications generated by our system run on any Web servers which handle CGI. FrontPage does not provide an easy method to manage sessions. Our method is easy to handle session management using session annotations.

JSP handles extended HTML files in which programs written in Java are embedded. Developers need procedural programming to handle business logic.

And JSP needs special programs called Servlet engines on Web servers. In our approach, we can handle business logic without procedural programming by using SOAP services. Web applications generated by our system run on any Web servers which handle CGI.

5. Conclusion

We presented annotation-based definitions of Web applications and a method for generation of Web applications from these definitions. We defined Web applications by HTML page templates with annotations. HTML page templates define views and user interfaces of Web applications. Annotations define processing of Web applications. Web applications are automatically generated from annotated HTML page templates using our annotation editor and D-Web system. Using our approach, we can define Web applications without knowledge of detailed implementation methods or procedural programming.

The annotation editor and D-Web system are currently in the final stage of implementation. We can generate Web applications from our annotation-based definitions which consist of a subset of our features. In our approach, we use two tools, Web page composers and our annotation editor, to define Web applications. By combining Web page composer with our annotation editor, we could define Web applications more efficiently.

Our annotation-based definitions are independent of implementation architecture such as CGI, JSP/Servlet and ASP. We plan to implement another systems which generates JSP/Servlet or ASP from same definitions for current CGI version of our system.

References

M. Hall. *Core Servlets and JavaServer Pages (JSP).* (Prentice Hall PTR, 2000).

L. Atkinson. *Core PHP Programming: Using PHP to Build Dynamic Web Sites.* (Prentice Hall PTR, 2000).

A. Homer, D. Sussman, B. Francis et al. *Professional Active Server Pages 3.0.* (Wrox Press, 1999).

K. Asami, and T. Tokuda. *Generation of Web Applications from HTML Page Templates with Annotations.* Proceedings of the IASTED International Conference, APPLIED INFORMATICS, pp.295-300, 2002, Austria.

K. Scribner and M. C. Stiver. *Understanding SOAP: The Authoritative Solution.* (Sams, 2000).

D. Martin, M. Birbeck, M. Kay et al. *Professional XML.* (Wrox Press, 2000).

L. D. Stein. *Web Security: A Step-by-Step Reference Guide.* (Addison-Wesley Pub. Co., 1998).

International System Research Inc. *The OpenZOLAR project.* http://www.isr.co.jp/openzolar/

J. Buyens. *Microsoft FrontPage Version 2002 Inside Out.* (Microsoft Press, 2001).