JOURNAL OF KOREA MULTIMEDIA SOCIETY, VOL. 7, NO. 6, JUNE 2004(pp. 851-857)

A Scheme to Interpret a JSP Page Using a New Concept of Scopes in Web Environment

Yongju Chung†, Dooheon Song††

ABSTRACT

Server-side scripting languages for web applications have a different environment from general programming languages. The reason is that some data in web applications should be transferred to a distinct file of a page or should be maintained for a physical time, that is for session time. So JSP has four kinds of new scopes such as page, request, session, application. And every identifiers in JSP are classified and processed as one of the four scopes. This seems unavoidable to a scripting language because of the web environment. So when a JSP page using these new scope concepts is interpreted the procedure would be different from that of the general programming language's scopes. This thesis has studied the processing of the scopes which are considered in interpreting a script language code. This processing method of the scopes in this article can be applied not only to JSP interpreting but also to a data processing of similar ranges in web.

Keywords: web application, scripting language, JSP, scope, implicit object, directive, scripting tag, action tag

1. INTRODUCTION

ASP, JSP and PHP are server-side script languages that are recently widely used in implementing web applications. They are developed to overcome the difficulty of the CGI method - the difficulty of web design. They use tags to express a web page like HTML. Among them JSP uses its own concepts different from ASP or PHP, for example, custom tags, standard action tags, although the other two script languages use some of the concepts implicitly. One of them is a new concept of scopes. These new concept of scopes in JSP are page scope, request scope, session scope and application scope. The concept of these scopes are designed because a JSP page is interpreted into a servlet and the servlet runs in the web.

A JSP page is interpreted into a servlet by JSP engine. And a servlet is a kind of Java program running in the web to make a HTML file dynamically. In fact a servlet is executed like a CGI program except it needs servlet engine. Since a JSP page is interpreted into a servlet and the servlet makes a HTML file dynamically JSP has a advantage compared with ASP and PHP. That is, one of the general purpose object-oriented languages Java can be coded in a JSP page.(This means all the Java packages can be used in a JSP page.) To get this advantage JSP uses a new concept of scopes. They are page scope, request scope, session scope and application scope.

These scopes are necessary if the underlining language is an objected-oriented language like JSP. For example, a web site can be moved to a different place. And suppose a client visit the old site. Then it is not desirable for a web server to make the client visit the new site again with a new request HTTP. In this case JSP uses an object for the request HTTP and forwards the object from

※ Corresponding Author : Yongju Chung, Address : (330-180) San 27, Anseo-dong, Cheonan, Chungcheongnamdo, Korea, TEL : +82-41-550-3463, FAX : +82-41-550-3460, E-mail : chungy@cs.dankook.ac.kr

Receipt date : June 11, 2004, Approval date : June 18, 2004

† Professor, Dankook Univ.

†† Professor, Dept. of Computer Science, Yong-in Songdam Univ.

(E-mail : mypham@naver.com)

the old site to the new site. But the names of this object in two JSP pages of the old site and the new site are used identically. So when these two JSP pages are being interpreted the names for the object should has the same value. It means the object name needs a new scope.(In JSP this scope is called request scope.) Likewise JSP has four kinds of scopes such as page scope, request scope, session scope and application scope.

An object or an identifier with page scope is an object or an identifier that can be accessible only within the page when it is declared. And all references to such an object or an identifier shall be released after the response is sent back to the client or the request is forwarded somewhere else. JSP has 7 implicit objects with this scope. They are page implicit object, request implicit object, response implicit object, out implicit object, config implicit object, pageContext implicit object, exception implicit object. And the other object with this scope is a bean object whose scope attribute is page. An object or an identifier with request scope is an identifier that can be accessible from pages processing the same request where it is declared. Objects or identifiers that have this scope in JSP are request implicit object, parameter names of the param action tag and bean object names whose scope attribute is request. An object or an identifier whose scope is session scope is an identifier that can be accessible from pages processing the same request that are in the same session as the one in which they are declared. Actually this scope is a range during session implicit object exists. So this scope does not cover logical range but physical time. Objects or identifiers that have session scope in JSP are session implicit object and bean object names whose scope attribute is session. An object or an identifier that has application scope is an object or an identifier that can be accessible from pages processing the requests that are in the same application as the one in which they are declared. Objects or identifiers with application scope in JSP are application implicit object, and bean object names whose scope attribute is application.

In general one of the easy way to process objects or identifiers with some kinds of scopes is to maintain a few tables and to check the scopes in compiling a program. But JSP is a tag language and the formats of the tags are simple compared with the general purpose language's format. So an interpreter can compile a JSP page into a servlet. But in interpreting a JSP page it should be considered that an object of a servlet can not be constructed and can not be executed by another servlet. All the objects of servlets in a web site should be constructed and executed by JSP engine calling service()'s of them. And the treatment of scopes is one of the main topics in compiling a language. But there are scarcely papers or articles about this problem. The reason for this might be because the interpreter of JSP is not thought to be so complex or might be because Sun Microsystems, a commercial company, has publicized JSP with the interpreter at the same time. But the solution to interpret a script language with these new scopes is still unknown.

So this thesis shows a method of processing identifiers or objects with these four scopes in interpreting JSP pages. And an interpreter was constructed for experiments. But the interpreter in this paper is not a perfect one. Since the focus of this paper is the concept of new scopes in the web it can not process custom tags. The experiments were carried out by executing two JSP pages and two servlets which were interpreted by the interpreter of this paper. The remainder of this article describes shortly the JSP tags in session 2 and how to implement the interpreter focusing the scopes in session 3 and finally the results.

2. TAGS IN JSP

Tags in JSP are classified into 5 categories, such as directives, scripting element tags, standard action tags, custom tags and comment tags.

(Standard action tags and custom tags can be classified as one because they all are action tags.)

2.1 Directives

Directives are messages to the JSP engine, that is, they have some information on the current page itself or setting for the page which should be preprocessed by the JSP engine. Directives also classified into 3 kinds of tags, such as page directive, include directive and taglib directive. The page directive has 11 attributes, such as, language, extends, import, session, buffer, autoflush, isThreadSafe, info, errorPage, isErrorPage, contentType. The include directive is used to substitute text or code in the current page. So the JSP engine should expand the current page with new code or text. The taglib directive identifies a tag library which is used by the custom tag. So the JSP engine can locate the tag library by this information.

2.2 Scripting Element Tags

Scripting element tags are tags inside which general purpose languages can be coded. There are 3 kinds of tags in these tags, such as a declaration tag, a scriptlet tag and a expression scripting tag.

2.3 Action Tags

There are two kinds of action tags. One is standard a action tag and the other is a custom tag. Standard action tags are tags that may affect the current output stream and use or modify or create objects. Custom tags are tags that are implemented and acted by a web application developer. The sorts of standard action tags are forward action tag, include action tag, param action tag and bean class action tag.

3. THE CONSTRUCTION OF THE INTERPRETER AND THE PROCESSING OF THE SCOPES

3.1 The Construction of the Interpreter

The interpreter was implemented in Java. It has four classes overally, JSP Interpreter, Preprocessor, LexicalAnalyser and PrintServlet.(Fig. 1) JSPInterpreter is the main class calling Preprocessor and PrintServlet. Preprocessor class processes all the directive tags gathering some information for request scope table, RequestTable, which is delivered to JSPInterpreter. PrintServlet is a class that prints servlets using RequestTable and LexicalAnalyser.

The directives were processed by Preprocessor in this paper. The attributes of page directive were implemented as follows.(All the values of these attributes were passed to PrintServlet.) The language attribute is meaningless because JSP currently supports only Java. The values of import and extends attributes were saved to be printed at PrintServlet. Since getPageContext() of JspFactory determines the buffer size and whether the output stream will be flushed automatically or not, the buffer and autoflush attributes were also saved. The value of the contentType attribute were saved also to be used by setContentType() of response implicit object. The value of the session attribute, that is true or false, can be processed when the session implicit object is created. So it was also saved. And since errorPage and isErrorPage are related to exception object the values of them were saved also too, which were used when the exception implicit object was created. When the value of isThreadSafe is true, nothing was done. But when it is false which means all threads should be run separately, all the code inside service() were

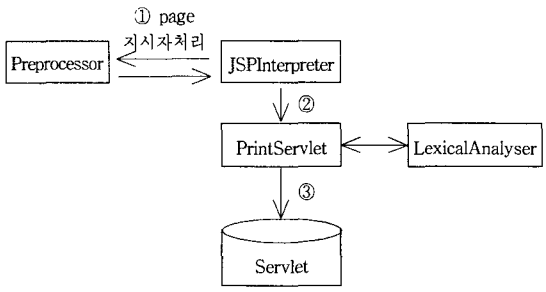


Fig. 1. The structure of the interpreter

declared as synchronized.

The scripting tags were processed in Print-Servlet. Since the code inside the declaration scripting tags can be used in the current JSP page it was printed as a field level in the servlet. And since the expression scripting tag is a value in the HTML file the code inside this tag was converted to a string using toString() at the position where this tag appeared in the current JSP page. The code inside scriptlet tag was printed in service() method of the servlet because they are code for each client.(As for action tags they will be explained in the following sessions.)

3.2 The processing of page scope

An identifier with page scope can be used in the same JSP page and the service() method is the actual only method among init(), service() and destroy() to be called by the JSP engine. So identifiers with page were interpreted within service(). But declarations within the declaration tags were interpreted into class member declaration parts of servlet since they can be used in other methods. So some kinds of data structure are not needed. For example the implicit objects with this scope-out, page, congif etc.- are constructed in service() method. Bean class with this scope was made to be declared and the object of it was constructed within service() method.

3.3 The processing of request scope

There are two action tags that have this scope in JSP, forward standard action tag and include standard action tag. An identifier with request scope can be used in more than two servlets. So a table is needed to check the scope. URL rewriting method was used to deliver data. The forward() or include() method of pageContext implicit object has URL as their parameters to forward or to include another JSP page. So instead of passing only a URL a query string was added to the URL, for example, "URL?parameterID=value". And the forwarded JSP page or the included JSP page gets the value using getParameter() method of the request implicit object.

3.4 The processing of session scope

session scope is not a logical range but a physical time. So if an identifier has this scope it should be checked before evaluating it whether the client has visited the site within a given time, for example 30 minutes. So a table is necessary to check the time elapsed. But checking a table can cause a delay. session implicit object exists only during session time. So session implicit object itself was used for an identifier with this scope. For example an identifier or an object has this scope then the identifier or the name of the object can be enrolled in session implicit object using session.setAttribute(). So if the value of the identifier or the reference of object is needed it can be accessed by session.getAttribute(). After session time passed the value can not be accessed because session object does not exist any more.

3.5 The processing of application scope

Objects or identifiers with this scope are objects or identifier that can be used in any JSP page of the application. So application implicit object was used for this process. That is, setAttribute() and getAttribute() of application implicit object was used to deliver data.

4. CONCLUSIONS

This paper has studied new concepts of scopes which were defined in JSP. Although the concepts are simple these may be helpful not only to the development of a server-side scripting language which sometimes require an object or an identifier to be used with the same name in the consecutive pages but also to the implementation of web applications which need a similar circumstance. For experiments an interpreter was constructed,

even though it is not a perfect one. The experiments were carried out by comparing the results of JSP pages and those of servlets interpreted by the interpreter.(At the appendix there are 4 programs. Two of them are JSP pages using request scope and two of them are servlets interpreted by the interpreter. The servlets showed the same results as the JSP pages. The JSP engine was Tomcat 4.0.)

5. REFERENCES

[1] Watt, D.A., *Programming Language Concepts and Paradigms*, Prentice Hall, 1990.

[2] Lindholm, T. and Yellin, F. *The Java Virtual Machine Specification*, 2nd ed. Addison-Wesley, 1999.

[3] David A Watt and Deryck F Brown, *Programming Language Processors in JAVA*, Prentice-Hall, 2000.

[4] Hans Bergstein, *JavaServer Pages*, O' Reilly, 2001 Jan.

[5] *JavaServer Pages, Specfications 1.2*, Sun-microsystems, 2001 Aug.

[6] Jeisn Hunter etc., *Java Servlet Programming*, O' Reilly, 2001

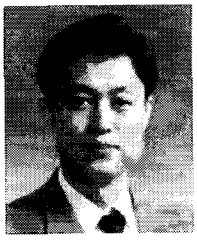
[7] Gal Shachor etc., *JSP Tag Libraries*, MManning, 2001.

[8] Yongju Chung, *JSP Explained in Servlet*, Sangneuing, Seoul, 2003, Feb.

[9] *JavaServer Pages Specfications 2.4*, draft final ver., Sun-microsystems, 2003 Mar.

[10] *Java Servlet Specification Version 2.4*, Sun-microsystems, 2003 April.

[11] Junsoo Youn, Implementation and Experiments of a JSP Interpreter, M.A. Thesis, Dankook Univ. 2003, Dec.



Yongju Chung

1981 Computer Science, BA, Seoul National University
1983 Computer Science, MS, KAIST
1984~Professor, Dankook University

Research Interests : Programming Language, Parallel Processing, Internet Traffic, Pattern Recognition & Processing, Natural Language Processing

Doohyeon Song

1981 Computer Science, BA, Seoul National University
1983 Computer Science, MS, KAIST UC Irvine Ph. D. A. B. D.) IST Researcher Professor, Dept. of Computer Science, Yong-in Songdam University

Research Interests : Machine Learning, CRM, Data Mining, Security, Bioinformatics, Image Recognition & Processing

Appendix

```
<%--   RequestTest.jsp   --%>
<%@ page language="java"
        contentType="text/html;charset=euc-kr" %>
<html>
<head>
<title> Test of request Scope </title>
</head>
<body>
<jsp:forward page="NewHome.jsp">
<jsp:param name="count" value="30" />
</jsp:forward>
</body>
</html>
```

Program 1. JSP page, RequestTest.jsp, using a request scope

```
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;

 public class RequestTest extends HttpServlet {
  public void service(HttpServletRequest request, HttpServletResponse response)
throws java.io.IOException, ServletException {
    JspFactory _jspxFactory = null;
    PageContext pageContext = null;
    HttpSession session = null;
    ServletContext application = null;
    ServletConfig config = null;
    JspWriter out = null;
    Object page = this;
      try {
_jspxFactory = JspFactory.getDefaultFactory();
response.setContentType("text/html;charset=euc-kr");
pageContext = _jspxFactory.getPageContext(this, request, response, null, true, 8192, true);
application = pageContext.getServletContext();
config = pageContext.getServletConfig();
session = pageContext.getSession();
out = pageContext.getOut();
Throwable exception = (Throwable) request.getAttribute("javax.servlet.jsp.jspException");
out.println("<%--  RequestTest.jsp  --%>" );
out.println("<html>" );
out.println("<head>" );
out.println("<title>" );
out.println("Test of request Scope" );
out.println("</title>" );
out.println("</head>" );
out.println("<body>" );
pageContext.forward("NewHome.jsp"+"?count=30");
out.println("</body>" );
out.println("</html>" );
    } catch (Throwable t) {
    out.clearBuffer();
    } finally {
    if (_jspxFactory != null) _jspxFactory.releasePageContext(pageContext);
    }
  }
}
```

Program 2. a servlet RequestTest.java translated from RequestTest.jsp

```
<%--   ReadData.jsp   --%>
<%@ page language="java"
        contentType="text/html;charset=euc-kr" %>
<html>
<head>
<title> Result of request Scope </title>
</head>
<body>
forwarded data = <%= request.getParameter("count") %>
</body>
</html>
```

Program 3. ReadData.jsp, a JSP page which shows the delivered data from RequestTest.jsp

```
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;

 public class ReadData extends HttpServlet {
  public void service(HttpServletRequest request, HttpServletResponse response)
throws java.io.IOException, ServletException {
    JspFactory _jspxFactory = null;
    PageContext pageContext = null;
    HttpSession session = null;
    ServletContext application = null;
    ServletConfig config = null;
    JspWriter out = null;
    Object page = this;
      try {
_jspxFactory = JspFactory.getDefaultFactory();
response.setContentType("text/html;charset=euc-kr");
pageContext = _jspxFactory.getPageContext(this, request, response, null, true, 8192, true);
application = pageContext.getServletContext();
config = pageContext.getServletConfig();
session = pageContext.getSession();
out = pageContext.getOut();
Throwable exception = (Throwable) request.getAttribute("javax.servlet.jsp.jspException");
out.println("<%--   ReadData.jsp   --%>" );
out.println("<html>" );
out.println("<head>" );
out.println("<title>" );
out.println("Result of request Scope" );
out.println("</title>" );
out.println("</head>" );
out.println("<body>" );
out.println(" forwarded data = " + request.getParameter("count").toString());
out.println("</body>" );
out.println("</html>" );
    } catch (Throwable t) {
    out.clearBuffer();
    } finally {
    if (_jspxFactory != null) _jspxFactory.releasePageContext(pageContext);
    }
  }
}
```

Program 4. ReadData.java translated from ReadData.jsp.