

BUILDING WEB SERVICES USING J2EE PLATFORM FOR SEWING PROJECT

B. SAKOWICZ, R. BRZÓZKA, M. DZIENIECKI,
A. NAPIERALSKI
TECHNICAL UNIVERSITY OF LODZ, POLAND

KEYWORDS: Web services, Java 2 Enterprise Edition, World Wide Web, Hypertext Markup Language,

ABSTRACT: Nowadays web pages, due to their advanced features, need appropriate methodology during design. This paper is an overview of the most common approaches to web services development (related to client and server side development). The thin client based architecture replaces the standard client-server one because of its main advantage - portability. This new architecture allows users to use applications all over the world not having to worry about the accessibility of the client side modules. That kind of service has been developed for SEWING project.

INTRODUCTION

Web services are currently the most popular types of applications. Before web generation has started, every application had to be written as stand-alone, or as a client-server modules. That solution had many disadvantages. The biggest one was importability. If someone wanted to use application, he had to use computer, where client application had been installed.

Web services, when started, have changed this situation. Starting from this point of time we are not isolated from our computers any more. In order to use our application we only need to have a computer connected to the network, a web browser and to remember our login.

Creation of web services is possible in several ways. There are many generators, which can create web pages using drag and drop method. These methods of creation are unfortunately totally useless, when it is necessary to design professional site. Every bigger web service provides some public services and confidential ones. Creation of that kind of pages is impossible without professional programming. There are many techniques of programming web services, but recently the most popular are based on Java language provided by Sun Microsystems.

SEWING WEB SERVICE SPECIFICS

SEWING project is realized by many institutions from all over the Europe. Different sides need to have unified way of communication and documents exchange. These documents are sometimes confidential and it is recommended to

protect them from others. At the same time they have to be shared between project participants. Because of this need we had to design structure of web service divided into two parts. One of this part is general information about project, conferences related to SEWING and participants descriptions. This part is accessible for everyone and it is a form of promoting the project. To access the second one, which is confidential part, it is necessary to know login and password. Default password was delivered to every entitled person, and after first accessing to protected part, he was asked to change it. Due to this solution access to confidential data is fully protected.

CLIENT-SERVER COMMUNICATION

SEWING web service contains two sides: server side (invisible for end user) and client side (which is a web browser, called in this case "thin client"). Simple schema of communication between web browser and web server is shown in Figure 1.

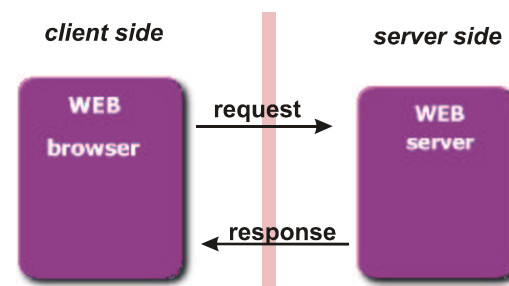


Figure 1. Client-server communication

WEB SERVICE CLIENT SIDE

To create simple web page, we need to use HTML (Hypertext Markup Language) [1,4,10,11]. Building client side using HTML is very simple but results are rather poor.

To add some special effects (and some other features) we use CSS (Cascade Style Sheets) [2,5,10,11] and JavaScript [6,10,11] together with pure HTML. CSS is a special way of describing behaviors of different parts of web page. CSS makes it possible to create a style for each tag, which can differ from default meaning. These styles have very interesting features like possibility of overwriting meanings of specified style in the middle of document and inheritance of styles. Using CSS allows us to centralize web page presentation.

Another part of client side is JavaScript. Using this simple programming language provides us with creating some graphical effects like dynamically changing images under some conditions (e.g. getting focus), validating forms in client side or using some resources from client computer.

WEB SERVICE SERVER SIDE

Another part of each web service is the server side. Using simple HTML makes impossible apply more advanced features like including one part of page into another, showing different pages depending on user interests, controlling authorization and many more things. All these behaviors have to be implemented on the server side. The approach applied for SEWING service is based on Java 2 Enterprise Edition (J2EE) platform [1,7,8,9].

It is possible to use J2EE platform in several ways. On the basic level J2EE consists of Java Server Pages (JSP), Servlets and Java Beans. All these components are designed to realize different parts of Model-View-Controller (MVC) architecture. Typical representation of server side components is shown in Figure 2.

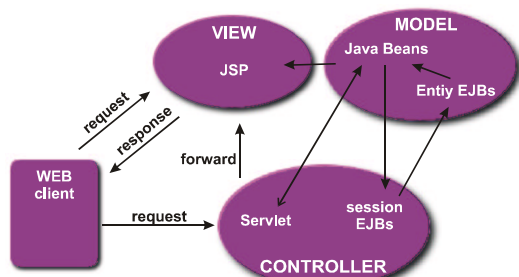


Figure 2. J2EE web components [9].

Not all of them need to be used in every application. Generally we can divide solutions into two parts: servlet-centric approach and JSP-centric approach. Page-centric approach is more intuitive and suitable for simple structures, when requests go directly from page to page. Servlet-centric approach is much better with more complicated logic. Servlet improves the abstraction between presentation and application logic. Business logic can be debugged in servlets before it is passed to JSP pages.

SERVER SIDE IMPLEMENTATION

In SEWING web pages we had to build application containing two main parts: public and protected for some group of persons. Application logic has been mostly included in servlet called "control". This servlet dispatches all requests and takes control over protecting resources. Every request is analyzed and depending on it control is redirected to appropriate part of application. If requested resource is accessible for everyone, it is just send to browser. In other case it is necessary to control permissions of current user. Information about user is kept in session. After successful authorization, special object (Java Bean) is put into session. This object contains information like name, password, professional title and permissions. To check if user is currently logged in we only need to find that object. If he is not already logged in he is redirected to login page. After entering login and password, appropriate servlet (called "login") checks in database provided data. If user has necessary privileges (which base on groups and roles schema) he receives asked resource (and his data is put into session). In other way, if he has not necessary privileges or he provided unsuitable data during login, he is adequately informed. Simplified set of server side components is shown in Figure 3.

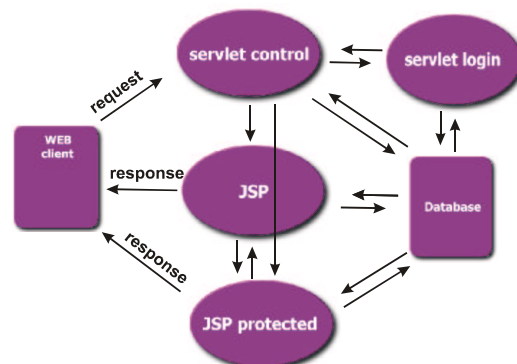


Figure 3. Server side components.

To protect confidential resources we used code shown in Table 1 (implemented in servlet "control").

Table 1. Protecting confidential resources.

```
Object obj = session.getValue("userLogin");
// find in session object containing
// information about current user
if (obj == null){
    response.sendRedirect("/log/index.jsp");
    // if it doesn't exist redirect
    // user to login page
} else if
(!"PARTNER".equals(((UserLogin)obj).getRole())){
    response.sendRedirect("/forbidden.jsp");
    // if user has no proper
    // privileges inform about it
} else {
    response.sendRedirect("/confidential.jsp");
    // show confidential page to the user
}
```

Authorization process (realized by servlet "login") is introduced in Table 2.

Table 2. Authorization process.

```
String login = (String)
request.getParameter("login");
// take parameter "login" from request
String password = (String)
request.getParameter("password");
// take parameter "password" from request
UserLogin userLogin=
logCheck.getUserLogin(login,password);
// use special object to connect
// to database and validate login
// and password
if (userLogin.isPassword() &&
    userLogin.isLogin()){
    session.putValue("userLogin", userLogin);
    // put information in session, that
    // user is already login
    response.sendRedirect("/protected/index.jsp");
    // redirect user to protected resource
} else
    response.sendRedirect("/login/index.jsp?login=
false");
// inform user, that login/password
// doesn't match
```

ACCESSING OTHER TYPES OF RESOURCES

Controlling access to web pages is relatively simple. Even if we know exact address of protected page, application immediately redirects us to proper component. Quite different solutions have to be made in case of other types of files. Including documents like .doc or .pdf in document root of web server makes possible to get them only by knowing their URL. But if they are confidential (e.g. financial reports saved in .pdf format), they need to be protected from unauthorized access. To solve that problem it is

necessary to place all these types of files out of document root. In this case confidential documents are protected from unauthorized access but at the same time they are out of range of web server (e.g. Apache). To deliver them to entitled person it was necessary to create another object, which could check privileges and at the same time had access to files placed over document root. Ideal solution is another servlet which has these capabilities. The only problem is necessity to serve files through the medium of servlet. Depending on different types of documents it is necessary to put adequate headers informing about MIME type and afterwards to send requested file as a binary stream. Sending .pdf document can be realized as shown in Table 3.

Table 3. Sending .pdf file through the medium of servlet.

```
// if user has necessary permissions :
Object objFile = request.getParameter("file");
// get parameter file
if (objFile != null){
    String pathName = (String) objFile;
    // convert it to String
    response.setContentType("application/pdf");
    // set in response content type and
    // content disposition
    String fileName =
pathName.substring(pathName.lastIndexOf("/") + 1,
    pathName.length());
    Response.setHeader("Content-Disposition",
    "inline;filename=\"" + fileName + "\"");
    File inputFile = new File(pathName);
    FileInputStream in =
    new FileInputStream(inputFile);
    int c;
    while ((c = in.read()) != -1) {
        out.write(c);
        // send file
    }
    in.close();
    out.flush(); // important !
    out.close();
}
```

CONCLUSIONS

It should be pointed out that presented approach is suitable for wide range of WWW applications. Similar solution has been already successfully implemented in another European Project – REASON (“REsearch And Training Action for System On Chip Design”, No. IST - 2000 – 30193, <http://www.reason.mixdes.org>). Of course problem of protecting resources is generally more complicated and presented solution is not always optimal. Presented method simplifies designing typical pages, when some of them have to be protected.

ACKNOWLEDGEMENTS

This work has been supported by the European Community under the SEWING project (“System for European Water MonitorING”, No. IST-2000-28084, <http://www.sewing.mixdes.org>).

AUTHORS

Bartosz Sakowicz, Rafal Brzozka, Michal Dziegiecki and prof. Andrzej Napieralski are with the Department of Microelectronics & Computer Science, Technical University of Lodz, Poland.

E-mail: napier@dmcs.p.lodz.pl
sakowicz@dmcs.p.lodz.pl

REFERENCES

- [1] HTML specification <http://www.w3.org>
- [2] CSS specification <http://www.w3.org>
- [3] Java Sun pages: <http://java.sun.com>
- [4] Bryan Pfaffenberger, Bill Karow: “HTML 4. Biblia” Wydawnictwo Helion 2001
- [5] Eric A. Meyer : “CSS. Kaskadowe arkusze stylów. Przewodnik encyklopedyczny” Wydawnictwo Helion 2001
- [6] Danny Goodman:”JavaScript. Księga eksperta” Wydawnictwo Helion 2000
- [7] James Goodwill:”Java Server Pages” Wydawnictwo Helion 2001
- [8] Wojciech Romowicz: “Java Server Pages oraz inne komponenty JavaPlatform” Wydawnictwo Helion 2001
- [9] Sun Tech Days - developer conference, 29-30 October 2001
- [10]“Strony WWW bez tajemnic” Chip Special grudzie• 1999
- [11] “Strony WWW bez sekretów” Chip Special pazdziernik 2000