Server-Side Java:
A New Direction for Teaching Computer Programming

Robert Stumpf[1]
Professor
Computer Information Systems Department
California State Polytechnic University
Pomona, CA 91768, USA

And

Steven Curl[2]
Associate Professor
Computer Information Systems Department
California State Polytechnic University
Pomona, CA 91768, USA

Abstract

The current direction in software development clearly favors server-side programming – often using the Java language. Still, many courses today may not yet be updated to keep pace with this trend. This paper explores the design of a new course, Server-Side Java Programming, which we will be offering at our university. The course will be the third in a three-course sequence that we intend to pilot test during spring 2002 with full implementation beginning in that fall. We believe this paper will be of interest to educators, who, much like ourselves, are only now beginning to appreciate this new direction in application software development.

Keywords: Java, server, programming, architecture

1. THE CIS PROGRAM

Our CIS Department is housed within the College of Business Administration. Other departments within the College include Accounting, Finance, International Business & Marketing, Management & Human Resources, and Technology & Operations Management. The program immerses the students in the object-oriented paradigm while allowing them to choose a specialization in application software development, business systems analysis, interactive web design, and telecommunications. Our school is on the quarter system and awards students successfully completing the CIS program with the degree of Bachelor of Science in Business Administration with an option in Computer Information Systems.

Students choosing to specialize in applications software development must take courses in object-oriented programming with Java, object-oriented systems analysis and design using the Unified Modeling Language, telecommunications networks, interactive web development, database, client-server computing with Visual Basic, rapid applications development, C++, advanced Java programming for business, and a capstone project course. Beginning in fall 2002, all application software development students will also take a server-side Java programming course.

[1] rvstumpf@csupomona.edu
[2] scurl@csupomona.edu

2. JAVA, JAVA, AND MORE JAVA

We believe that a three-course (120 contact hour) Java sequence provides a better match between the emerging needs of today's business applications developers than we can provide given our previous two-course (80 contact hour) sequence. In our new curriculum, our first two courses will continue to cover fundamental and advanced programming concepts using the Java language. Our current texts for our fundamental and advanced courses are Core Java: Volumes 1 and 2 (Horstman and Cornell 1999). These first two courses are not the focus of this paper and are omitted from the remaining discussion.

3. WHY SERVER-SIDE JAVA?

Java server programming, or server-side Java, alleviates the need for CGI (Common Gateway Interface) scripts typically used for writing to files or databases on a server and for communicating with HTML pages. Why server-side programming at all? Server-side programming is important because in a distributed browser-based environment, programming on the client-side means writing applets or using a scripting language. Applets are considered to be risky for consumer cites as they take longer to download and later versions of Java (Java 2+) may not be enabled. While the versioning problem can be alleviated by using "automatic plug-ins" (Horstman 1999a), only increased bandwidth can address the lengthy download time. We do not expect a return to applet programming until bandwidth capacity grows to the point where the installation of plug-ins becomes transparent to the consumer.

Restricting programming to an applet-based environment is also undesirable since applets are, by design, limited in their capabilities. To protect unwitting client machines from unscrupulous behavior on the part of an applet, the Java security model uses a concept called a "sandbox" to place limits scope and range of an applet's abilities. Activities outside of the sandbox are not permitted. In practice, this means placing severe limits on what can be accomplished.

Similar arguments apply to scripting. In practice, a scripting language is essentially a cut-down version of its full language counterpart. Scripting languages are designed to run within a narrower set of conditions than is the case for a full version of the language. By definition, this limits the range of available features that are supported in the scripting version.

The above restrictions apply only to client-side programming, not server-side. Teaching a server-side Java programming course means teaching the full version of the language, without limits, in an environment targeted at the growing market for electronic commerce applications.

4. WHICH APPROACH TO USE?

In Java, a server program, or servlet, typically exists to provide services to a client-side program. There are several ways to communicate between the server and client. Three common approaches are direct calls from an HTML page, calls from a Java applet, and using JSP (Java Server Pages).

From an instructional standpoint, we need to provide some means of communicating with the server, and we intend to provide examples using all three of these approaches. However, the course will clearly focus on the server aspects, including distributed objects and database access.

The question on which approach to use presents the instructor with choices. Two criteria might be used to judge which approach is best. One is how much time is available and the other is the student's background in the fundamental concepts of arrays and streams.

If time is short, using HTML pages is clearly the easiest as it is very simple. HTML communication to the servlet uses simple commands involving parameters. HTML has the added benefit of simplicity, as no scripting language is required – only basic knowledge of HTML forms.

Using applets to servlets requires knowledge of arrays and streams. This approach is also much more difficult and time consuming to handle in the classroom. At our school, we have already attempted to introduce server-side programming using applets to servlets. Unfortunately, not all students were able to finish the projects because at this time the students had only one Java course behind them. We expect this to be less of a problem after the third Java course is fully implemented next year.

The third approach is to use Java server pages. The difficulty with this approach is that a greater knowledge of building HTML pages is required. If the students are not very familiar with HTML pages using forms, additional class time will be required on this topic.

Another way to answer this question is to ask which method is preferred by business clients. We have been informed by the consulting companies that build web sites for customers in a business to customer environment, that JSP or HTML pages connecting to a server are best due to the unsophisticated nature of the consumer (Hsu 2000). In practice, this means support for a modem connection with a browser that may not be Java enabled. Thus, applets would either take too long to download or just not work because of lack of version support.

However, in a business to business environment, the reverse is true. We have been informed that in this controlled environment, there is likely to be a high-speed connection and enforceable minimum browser standards (Hanna 2000). Thus, the desktop can use applets for data validation or local processing while the server can be saved for calls to databases. This of course reduces the need for large servers.

5. COURSE OVERVIEW

The text for our course is Marty Hall's *Core Servlets and JavaServer Pages* (Hall 2000). Because of prerequisites, we expect that students will still have access to both of the Horstman and Cay texts. As discussed earlier, the course is designed for a quarter system school with students meeting four hours per week for ten weeks (not including final exams). Note that we have eliminated any discussion of CGI from the course. Our course

Course Outline:

Mtg	Topic
1.	Server-based Programming Architecture
2.	Anatomy of a Servlet
3.	Handling Client Requests - 1
4.	Handling Client Requests - 2
5.	Generating Server Reponses - 1
6.	Generating Server Reponses - 2
7.	HTML-Servlet Communication
8.	Cookies
9.	Session Tracking
10.	Midterm Exam
11.	Java Server Pages (JSP)
12.	Java Beans
13.	Database Connectivity - 1
14.	Database Connectivity - 2
15.	Applet-Servlet Communication - 1
16.	Applet-Servlet Communication - 2
17.	Sockets
18.	Connection Pooling
19.	Distributed Objects
20.	Emerging Issues in Server Programming
21.	Final Exam

Figure 1: Course Outline

outline appears in Figure 1.

The course begins with an introductory lecture on web-enabled applications, distributed software architecture, and the role of server-based programming. We follow this with a discussion on the anatomy of a Java servlet, including both its internal architecture and public interface. Lectures 3 through 6 cover the dialog between client and server, first from the client side where we examine handling client requests and then from the server side where we examine the server generated response to these requests. In lecture 7 through 9 we introduce the first of the three approaches to server communication discussed earlier as we look at HTML-server communication, the supporting role of cookies, and need for session tracking. At this point, we conclude the first half of our course with a midterm exam.

We begin the second half of the course with a discussion on Java server pages. This is the second approach to server communications discussed earlier and is followed closely by a look at Java beans and their supporting role in a component-based architecture. No server-based programming course should be without a database component and so lectures 13 and 14 cover database connectivity and its place in a distributed processing environment. In lectures 15 and 16, we introduce applets as our third approach to server communication. The remaining lectures, 17 through 20, provide support for the above material. In these lectures, we examine several related topics, including programming for sockets, the performance benefits of connection pooling, and the whys and wheres of distributing objects. The course concludes with a look ahead to examine whatever new topics may be of interest with a discussion on emerging issues in server programming.

The course uses between three and five programming assignments, with each assignment building on its predecessor so that in the end the students will complete one large project. At present, we have two projects developed: one is a rental car reservation system; the other an online store and shopping cart.

6. JAVA ENVIRONMENT

A reputable Java course needs a solid development environment. In reviewing the range of available tools, we feel fortunate that so many vendors provide excellent support for the Java language. While we have not yet reached a final decision on a programming environment, some of our considerations include support for the latest version of Java, features of the development environment, hardware requirements, ease of use, and price. Thus far, in preparing for our course, we have tried tools by Sun, IBM, and Jakarta. All of these tools are available free of charge to students. No single product excelled in all areas.

The first product we used was JSDK (Java ™ Servlet Development Kit) version 2.1 from Sun Microsystems, Inc. It is free and Sun grants you a non-exclusive and non-transferable license for internal use only. This makes it available to educators. Its major advantage is it is simple to install (just extract a zip file and it is ready to execute). Most of the time no adjustment in path or class paths are needed. This assumes a Java Develop-

ment Environment is already installed. It also is a simple download (367 KB). Its disadvantage is that it opens the DOS (Disk operating System) console in windows. However, no entries are required in this environment.

The second product we used was IBM Visual Age for Java™. Many versions of Visual Age are available on IBM's web site for free downloading. However to run server side Java, requires either the enterprise or professional edition. This, in turn, requires IBM to authorize its use. Since we have a close working relationship with IBM, we were able to obtain copies, and also make additional copies that we placed in our library for students to check out. Our first experience was with the enterprise edition, version 3.0. It was a very stable, but slow product. It required at least 128 MB of random access memory and, at 73 MB, proved too large for many students to download. Its major advantage was that it allowed one to put break points in the sever code to enable debugging. It was also possible to put breakpoints in client code as well. This package uses a scaled down version of IBM WebSphere ™. It worked fine. We had to stop using it as it only includes the JDK 1.7, which did not have all the latest collection classes. Also its version of the Java "Swing" classes was dated. Since it was not possible to change the Java JDK it uses, we had to discontinue the product. Then we tried to use IBM's new 3.5 version professional edition. As of this time, we found it to be a very unstable product that was unable to export correct code. We had to return to Sun's JSDK in order to complete the course.

A more successful product is "Tomcat" provided by the Apache Software Foundation. Its license states "distribution and use in source and binary forms, with or without modification, are permitted." It is a reliable product that is almost as easy to set up as the one from Sun. Actually, it evolved from the JWDK from Sun. Its major difficulty was in setting paths – which requires more than a little of knowledge of DOS. Its only other disadvantage is that it does not do enough to aid in debugging code. On the plus side, we found this package to be a simple download (3 MB) and comes with the added advantage of enabling students to upload web pages and servlets to the server by FTP (File Transfer Protocol) without turning off the web server. We were also pleased to find that students can test their work on a server using any web browser. Currently this is the tool we are using.

We have considered products such as Visual Café and JBuilder™ from Borland. The difficulty is that versions that support server side Java are not free. Since the cost to our students is a major criteria we are concentrating on no or low cost options.

7. COMPUTING SUPPORT

Besides running in a standalone environment, students are required to upload and run their programs on our server, which we have dedicated for this purpose. This machine is configured with Windows 2000 Server and TOMCAT. Upload and execution of programs is done by students using anonymous FTP to a common folder with students granted write-only access. Individual programs are distinguished by students prefixing their initials to the project name and adding and an optional version number. As a security precaution, a directory listing is not provided so that students cannot see the progress of others. All administrative support is performed by the faculty member teaching the course.

8. CONCLUSION

Developing a new course is always challenging, but more so when the course involves cutting-edge information systems technology. In this paper, we have examined the rationale, placement, and structure for a server-side Java programming course. For us, this course is the logical next step in our ongoing process of curriculum revision and we believe this may be the case for other schools as well.

9. ACKNOWLEDGMENTS

We wish to thank Robert Hanna, Jet Propulsion Laboratory, Pasadena, California and Fong Hsu, IBM Center for Innovation, Santa Monica, California, for sharing their knowledge and insights with us.

10. REFERENCES

Hall, Marty, 2000, *Core Servlets and JavaServer Pages (JSP),* Englewood-Cliffs, NJ: Prentice-Hall.

Hanna, Robert, 2000, Interview notes: Jet Propulsion Laboratory, Pasadena, California.

Horstman, Cay and Gary Cornell, 1999a, *Core Java 2, Volume 1 – Fundamentals*, Prentice-Hall.

Horstman, Cay and Gary Cornell, 1999b, *Core Java 2, Volume 2 – Advanced Features*, Prentice-Hall.

Hsu, Fong, 2000, Interview notes: IBM Center for Innovation, Santa Monica, California.