第27卷 第1期 计 算 机 工 程 2001年1月
Vol.27 № 1 Computer Engineering January 2001
· 开发研究与设计技术 · 文章编号：1000—3428(2001)01—0170—03 文献标识码：A 中图分类号：TP393；TP311

用JSP/Servlet技术构建Web应用

吴晨清，荣震华
(上海交通大学计算机系，上海 200030)

摘 要：JSP/Servlet是Java技术在Web上的扩展，支持Web服务器端的应用开发。介绍了JSP/Servlet技术的主要特点，并分析了其在Web应用开发中两种典型的软件体系结构。

关键词：Web应用；Java；Servlet；JSP

Implementing Web Application Using JSP/Servlet Technology

WU Chenqing，RONG Zhenhua
(Department of Computer Science, Shanghai Jiaotong University, Shanghai 200030)

【Abstract】JSP/Servlet technologies are the Java platform technology of choice for extending and enhancing Web servers. This paper discusses key feasures of these technologies, and describes two JSP/Servlet software architectures used in Web application implementation.

【Key words】Web application; Java; Servlet; JSP

JSP/Servlet技术是Java家族中的新成员。作为Java 2 Platform Enterprise Edition(J2EE)标准的一个组成部分，JSP/Servlet主要用于Web服务器端应用的开发，是Java技术在Web服务器上的扩展。

1 Servlet

Java Applet是运行在客户端（浏览器）中的Java类，Servlet正好与之相对，它是运行在服务器端（Web服务器）上的Java类，主要处理Web请求，动态产生HTML页面。

如图1所示，浏览器按照HTTP协议向Web服务器提出请求（如键入一个URL地址），Web服务器响应后，把发给Servlet的请求，转交给Servlet引擎处理。Servlet引擎检查对应的Servlet是否已装载，否则先将其载入内存并初始化，再由该Servlet处理请求。如果Servlet中含有访问数据库的操作，则还要通过相关的JDBC驱动程序，与数据库连接，对数据库进行访问。最后Servlet将动态生成的标准HTML页面，送至客户端浏览器。

从以上的分析可以看出，Servlet基于一种请求/应答的工作模式，其主要特点在于：1) 高效 Servlet技术，为每一个请求创建一个轻量级（Lightweighted）的线程来处理。由于线程占据的系统资源远远小于进程，所以有效避免了CGI中因为为每个请求创建进程而引起的资源紧张、效率偏低的问题；2) 具备Java的所有优点 特别是平台独立性的特点，使Servlet可以不受软硬件环境的影响，运行于包括Windows、Unix在内的任何平台上；3)可以访问丰富的Java API Java API提供对事务、数据库、网络、分布计算等方面的广泛支持，特别是Java JDBC，是Servlet连接数据库的基础。

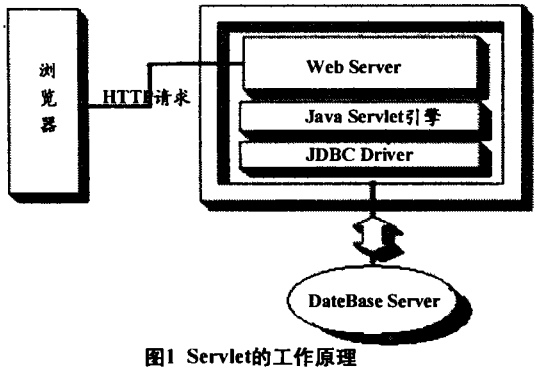

图1 Servlet的工作原理

另外，Servlet提供多级信息共享。HTTP本身是无状态（Stateless）的协议，但在Web应用中，却常常需要保持用户与服务之间，或服务与服务之间的某些信息。在Servlet中，提供请求、会话和应用3级信息共享。

2 JSP

JSP的全称是Java Server Page。基于JSP的页面以.jsp文件形式驻留在Web服务器上。在这种文本文件中，混合了HTML、DHTML标签（tag）和用Java写的脚本，某种程度上类似于.asp文件。

JSP借鉴了许多ASP的思想，但其本质上还是基于Servlet的，是Servlet技术的一种延展。每个JSP文件，总是先被JSP引擎自动编译成Servlet，然后再由Servlet引擎运行。也正因为如此，JSP间接具备了Servlet的诸多特点，包括ASP所没有的平台独立性（platform independence）。

除此之外，JSP还有它自己的特点。用JSP开发动态网页比较简单。纯粹用Servlet实现的Web应用，对网页所作的任何修改，都必须重新编译Servlet。采用JSP则无须编写程序，开发人员可以直接用HTML标签制作页面，并在需要处，加入脚本命令，生成动态内容。

JSP支持对组件的访问。组件本身体现了封装和复用的思想，基于组件的开发能有效提高软件生产效率。多数JSP页面依靠JavaBeans及企业级JavaBeans(Enterprise JavaBeans，EJB)组件，来完成应用所需的复杂处理，并通过页面中的脚本把具有特定功能的组件集成在一起。现在，由第三方开发的共享组件越来越多，大大简化了用JSP实现

作者简介：吴晨清(1974～)，女，研究生，主攻方向：面向对象技术与分布式仿真；荣震华，教授

收稿日期：2000-05-15

动态网页的工作.

JSP允许开发人员自己定义标签（tag）。这种标签，一般对应于标签库(tag library)中的Java程序，代表特定的应用功能。对页面的设计者来讲，它们类似于普通XML标签，使用方便，有利于简化动态页面的开发。

JSP使网页的外观设计与其动态内容分离。JSP页面中的HTML标签定义了网页的外观，嵌在页面中的脚本、JavaBeans组件和标签库，共同生成与应用逻辑相关的动态内容。这样，精通网页制作的设计者只需关注HTML，而精通Java的软件工程师则可以集中精力开发应用逻辑（Java类或JavaBeans组件）。某一方对页面布局或应用逻辑的改动，都不会影响另一方。

3 用JSP/Servlet构建Web应用

正如前面所述，JSP和Servlet是两种极具特色的动态Web技术，在许多方面并不逊色甚至超过已熟悉的CGI和ASP。尽管这两种技术都能够接收用户请求，实现交互处理，但两者还是各有侧重。相对而言，Servlet更适合于程序员，它本身是一个Java程序，Java面向对象的特性和强大的处理能力，使其易于实现复杂逻辑处理。相比之下，JSP技术在网页的编辑上更有优势。.jsp文件相当于一个嵌入了脚本的html文本，有专门的工具辅助网页的设计，适合网页的制作人员。

如果撇开底层运行机制上的相同之处（JSP被翻译成Servlet再执行），单从开发人员的角度来看，完全可以单独采用其中一种技术实现一个动态Web应用，纯JSP和纯Servlet方案都是可行的。但在实践中，更多是将这两种技术共同用来开发Web应用，即根据每个子任务的要求，以及JSP和Servlet的特点，选择合适的实现方式：某些交互处理用JSP实现，某些交互处理用Servlet实现，而某些则可以考虑用JSP和Servlet的互操作来解决。

运用JSP/Servlet技术实现Web动态交互，没有统一的模式。我们主要采用了两种模型。下面介绍这两种模型。

3.1 模型I

图2是模型I的体系结构图。从浏览器发来的请求，由JSP接收处理。JSP通过访问JavaBeans，连接数据库或后端服务器，获取相关数据，进行相应的处理。从JavaBeans返回的结果，经JSP提取并重新组织后，动态产生HTML页面，返回给浏览器。用户从显示的页面中得到交互的结果。

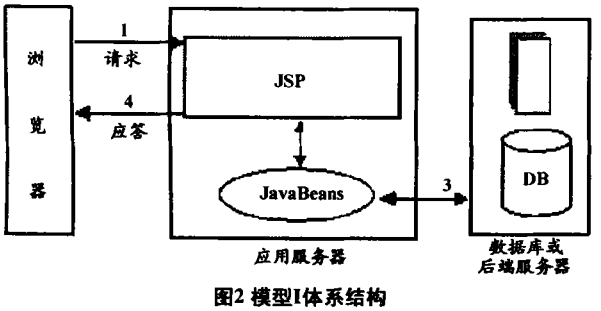

图2 模型I体系结构

模型I最大的特点是简单。这是一个纯JSP的方案，它充分利用了JSP技术易于开发动态网页的特点，依靠一个或几个JavaBeans组件实现具体的应用功能，生成动态内容。

3.2 模型II

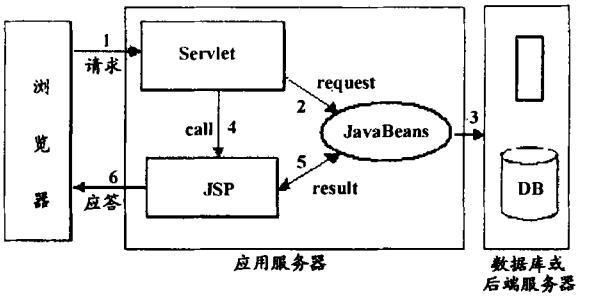

图3 模型II体系结构

模型II是一种混合的体系结构(见图3)。它通过JSP和Servlet的合作来实现交互处理。这种模型体现了MVC（model-view-control）的设计模式。JavaBeans组件构成了应用模型（model），实现各个具体的应用逻辑和功能。Servlet处于控制者（controller）的位置，负责处理HTTP请求，包括对输入数据的检查和转换，通过JavaBeans访问数据库或Enterprise JavaBeans，初始化JSP中要用到的Beans或对象，根据处理中不同分支和执行的结果（如成功或失败），决定转向哪个JSP等。JSP作为用户界面程序(View)，负责生成交互后返回的页面。它主要通过信息共享，获取Servlet生成的对象或Bean，从中取出相关数据，插入到HTML页面中，几乎没有处理逻辑。

3.3 两种模型的讨论

从上可以看出，模型I和模型II，整体结构都比较清晰，易于实现。它们的根本思想都是要尽可能地分离Web应用中的页面外观制作和应用逻辑实现。模型I中，大部分的处理在JavaBeans组件中进行；JSP文件主要用于产生动态网页。而在模型II中，Servlet处理HTTP请求，JSP仅仅负责生成网页的工作。这样的设计，使网页和应用逻辑两个部分的开发调试相对独立，便于修改和维护。

两种模型的区别主要在于，处理流程的主控部分所处的位置。尽管JavaBeans组件中封装了有关的应用逻辑，可以用来产生动态内容，但是仍然需要一个主控逻辑，将JavaBeans、用户的输入和返回的网页3个部分有效衔接起来。在这里，模型I利用了JSP中的脚本代码作为主控逻辑，而模型II则专门用了一个Servlet。也正是因为这一点上的不同，两种模型在适用性方面亦存在差异。虽然都是实现Web交互的有效方案，但发现，模型I更适合于简单的交互处理，而模型II则更适合较复杂的交互处理。

当处理逻辑比较简单时，用模型I实现是非常方便的：只要用HTML标签和少量的脚本就可以实现动态交互，根本不必编写程序。然而，当处理逻辑比较复杂、分支较多或者需要涉及多个JavaBeans组件时，模型I有可能使JSP文件中嵌入大量的脚本语句，相当一部分处理逻辑和页面描述被混在一起，不利于两个部分的独立开发和维护。特别是在大型项目开发中，由于页面制作与逻辑处理分别由不同的专业人员承担，这有可能引起分工不清，影响项目的管理。

与模型I相比，模型II更彻底地分离了应用处理与页面生成的工作。它利用了两种技术的优势，让Servlet完成具体处理流程的控制，让JSP负责动态网页的生成，便于不同专长的专业人员合作开发Web项目。越复杂的交互处理，用模型II的好处越明显。但是，由于模型II需要编写Servlet和

JSP，对简单的交互处理，实现远不如模型I方便。

模型I与模型II是两种用JSP/Servlet开发Web应用的方法，有很好的实用性。当然，用JSP/Servlet技术实现动态交互Web应用，不限于这两种，关键还是要把握JSP/Servlet的特点，根据不同的应用逻辑和客户需求，选择适合的模型实现不同的交互处理，力求使整个应用的体系结构更趋合理。

4 结束语

Web应用正逐步走向电子商务和企业级计算(Enterprise Computing)，其复杂性和动态性的要求更高。凭借Java技术的诸多优势，JSP/Servlet，以其动态、高效、简洁、与平台无关等特点对Web应用的开发，提供了良好的支持。作为一项新兴的Web技术，它正在受到全世界越来越多的Web应用开发者的重视。

参考文献

1 Java Servlet API Specification 2.2. http://java.sun.com/products/servlet

2 JavaServer Pages 1.1 Specification. http://java.sun.com/products/jsp

3 JavaServer Pages Key Features. http://java.sun.com/products/jsp/keyfeatures.html

4 Nordahl C. Java Server Pages for the ASP Developer. http://www.asptoday.com/articles/19991022.htm

5 Lavandowska L. Jspbook. http://www.esperanto.org.nz/jspbook

6 Bergsten H. An Introduction to Java Servlets. Web Developers Journal, 1999-03-10. http://webdevelopersjournal.com/articles/intro_to_servlets.html

7 Chowdhry P. JSP Will Win Hearts, But Competition is Strong. http://www.zdnet.com/pcweek, 1999-08-02

※※

（上接第142页）

在文献[3]中,Abadi和Needham凑了设计密码协议的11条原则；上面讨论的弱点违反了第10条原则：如果一种编码被用来表示一个报文的内容，那么它必须能够分辨出使用了哪种编码。通常情况下，编码是依赖于协议的，推断某个报文属于一个协议，事实上是这个协议的一次特定运行，是可能的，并且知道此报文在协议中的编号。

我们应该再加上一条，即当一个报文有多于一个组成部分时，应能分辨它是哪一部分。

为了抵抗这些攻击，需要确信在协议中不同的加密部分有不同的形式。像文献[3]中一样，一种方法是包含每个加密部分的文本："这是Woo和Lam相互认证协议中message 4的第二个加密部分"，或者是一些更精确的表达。

在Andrew协议中（第4节），利用的弱点是不能分辨出用密钥K_{ab}加密的某个特定报文是从A到B，还是从B到A的。类似地，在站到站协议中（第4节），分辨出谁是加密部分的目的接收者也是不可能的。

类似的缺陷也是能对其它一些协议进行攻击的原因：对CCITT X.509的攻击；在文献[3]中所报告的对Denning-Sacco协议的攻击；在文献[3]中报告的对SPLICE协议的两个攻击；对Needham-Schroeder公钥协议的攻击。这也是在文献[8]中提出的对Neumann-Stubblebine协议进行攻击的原因，它也能攻击KSL。

这些弱点违反了文献[3]中的第三条原则：如果一个主体的标识只对一个报文的内容是必不可少的，那么在报文中显式地提到主体的名字是明智的。

（我们用"加密元组"而不用"报文"，是为了强调对内容取决于主体标识的元组应该把那个主体标识包含进去。）

对上述原则的一个附带条件是所设计的协议应能抵抗入侵者猜测一个弱密钥，例如一个口令。如果用这样一个弱密钥加密的报文中包含一个主体的标识，那么若入侵者正确地猜出了那个密钥，则他可能很容易验证自己的猜测。

对于已经提出的协议，只有当后来发现它有缺陷时，需要开发更好的协议分析方法的必要性才显得那么清楚。

最著名的方法是使用BAN逻辑。它擅长捕捉某些种类的协议错误，特别是关于更新的错误，用这种方法已检测出了很多错误。不幸的是，使用这个逻辑很容易犯错误。如上所述，这个逻辑不能捕捉入侵者重新排列组成元组而造成的错误。甚至在理想化阶段经常会出现错误，特别是一个报文被理想化而包含报文本身所没有的信息这种错误。最后，它并不考虑自身事件的保密性。建议在协议分析的第一阶段使用BAN逻辑，但是要认识到它不能检测出所有的错误。

几年前，绝大多数对协议的新型攻击是因为协议缺少更新标志。幸亏有BAN逻辑，协议设计者们才能够避免犯这些错误。我们的经验表明现在的绝大多数新型攻击是因为违反了文献[3]中所列出的某条原则而引起的，特别是以上所引用的两条。我们建议把文献[3]作为任何协议设计者或者协议攻击者的必读物。

希望本文所述的攻击方法能够激励协议设计者们吸取他人的教训，避免在新协议中引入同样的脆弱性。

参考文献

1 Roscoe A W. Intensional Specification of Security Protocols. In 9th IEEE Computer Security Foundations Workshop, 1996

2 Gollmann D. What Do We Mean by Entity Authentication? IEEE Symposium on Research in Security and Privacy,1996

3 Abadi M , Needham R. Prudent Engineering Practice for Cryptographic Protocols. Research Report 125,Digital Equipment Corporation Systems Research Center, 1994

4 Diffie W , Orschot P C V , Wiener M J. Authentication and Authenticated Key Exchanges. Designs , Codes and Cryptography, 1992

5 Woo T Y C , Lam S S. A Lesson on Authenticated Protocol Design. Operating Systems Review,1994

6 Satyanarayanan M. Integrating Security in a Large Distributed System. ACM Transactions on Computer Systems ,1989

7 Kehne A , Schonwalder J , Landendorfer H. A Nonce-based Protocol for Multiple Authentications. Operating Systems Review, 1992

8 Hwang T , Lee N Y , Li C M et al. Two Attacks on Neuman-Stubblebine Authentication Protocols. Information Processing Letters,1995

9 Millen J K , Clark S C , Freedman S B. The Interrogator: Protocol Security Analysis . IEEE Transactions on Software Engineering. 1987

10 Demmerer R , Meadows C , Millen J. Three Systems for Cryptographic Protocol Analysis, Journal of Cryptology,1994