Въведение в Web-програмирането с Java сървлети

© Светлин Наков, 2000
http://www.nakov.com

В последните години в резултат на бурното развитие на ИНТЕРНЕТ и WEB-технологиите генерирането на динамични WEB-страници стана неизбежна част от изграждането на всяка WEB-ориентирана информационна система. Най-вече нуждата от обработката и извличането на информация от корпоративни бази данни през WEB доведе до сериозно развитие на WEB-програмирането като цяло. В настоящата статия ще разгледаме съвсем накратко основните начини за доставка на динамична информация в WEB, като се спрем по-подробно на една сравнително нова технология – JAVA сървлетите.

Една от първите техники използвани в ИНТЕРНЕТ за генериране на страници с динамично съдържание е доста известният Common Gateway Interface (CGI). Чрез CGI web-сървърът подава заявка на външна програма, която се изпълнява на сървъра. Изходът от тази програма се изпраща при клиента като статичен документ или част от документ. Ползата от CGI е възможността за вграждане на всички видове функционалност във динамично генерирани web-страници като това остава скрито от клиентския браузър. Поради големите си възможности и леснота за използване CGI бързо се превръща в стандарт за повечето web-сървъри, като се утвърждава като стандартен механизъм за комуникация между сървъра и външни сървърски приложения (server-side applications). Когато клиентският браузър изпрати заявка за достъп до CGI-програма, web-сървърът създава нов процес за да я изпълни и й подава параметрите чрез променливи от средата или стандартния вход. Програмата изпълнява заявката и връща резултатът на стандартния изход. Лошото на тази мощна технология е че създаването на нов процес е бавна операция и отнема значителни ресурси сървърски ресурси. Като следствие сървърът става бавен и броят клиенти, които могат да бъдат обслужени в едно и също време намалява. Въпреки това CGI-програмите могат да се напишат на почти всички езици за програмиране и скриптове, които се поддържат от сървъра и операционната система. В UNIX-среди най-често се използват Perl-скриптове, SHELL-скриптове и програми на C, докато под WINDOWS могат да се използват и произволни изпълними програми на C++, DELPHI, Basic и т.н. Използването на Perl позволява междуплатформена преносимост на кода, но нивото на интеграция между web-сървъра и сървърската програма не е добро, понеже те са отделни процеси.

За подобряване на скоростта на CGI-технологията се появява неин усъвършенстван модел – FastCGI. FastCGI е почти като CGI, но за разлика то него, web-сървърът не създава всеки път процес при заявка към CGI-програма, а стартира необходимата програма при първата заявка, а след това я използва без да създава нов процес. Въпреки че това подобрение на CGI технологията е стъпка в правилната посока, тя не решава някои от проблемите. При заявка на няколко клиента към една и съща програма едновременно отново се налага стартиране на няколко процеса, а освен това интеграцията с web-сървър отново не е много добра.

Едно добро решение на проблемите е реализирано в най-популярния и най-използвания web-сървър в света – Apache (по последни данни с Apache работят повече от 60% от всички web-сървъри в света). Apache има възможност да вгради в себе си интерпретатора на Perl, като подобрява сериозно скоростта и интеграцията. Както знаем Perl е много мощен скрипт-език за създаване на server-side скриптове, който е проектиран и разработен специално за тази цел. Технологията е известна като mod_perl и напоследък добива популярност.

Напоследък за малки и средни клиенти най-често се използва стандартната конфигурация за интернет сървър – ОС: Linux, за по-сериозните SunOS или друг UNIX, web-сървър: Apache с mod_perl или PHP, база от данни: MySQL, PostGre, а за по-сериозните ORACLE.

Друг опит за подобряване на ефективността на CGI/Perl технолгията е реализацията на PerlEx за WINDOWS NT, която донякъде също интегрира Perl в web-сървъра.

Като съвсем различен подход може да се разгледат сървърните разширения, които се поддържат от някои web-сървъри. Става въпрос за Netscape NSAPI, Microsoft ISAPI и други. Тези

технологии позволяват да се разшири функционалността на web-сървъра чрез вграждане на изпълним код в процеса на самия сървър, което при добра реализация води до изключително голяма ефективност, но често пъти това е за сметка на надеждността. Обикновено такива програми се пишат на C/C++ и се зареждат като DLL библиотеки в адресно пространство на сървъра. Проблемът е, че при крашване на едно такова вградено сървър-разширение, понякога се крашва или блокирва целият сървър и трябва да се рестартира. Това наистина това е много сериозен проблем.

Друга алтернатива на CGI е технологията Active Server Pages (ASP) на Microsoft. ASP позволява създаване на динамични web-страници, като за целта целият код, който поражда динамичното съдържание се вгражда в HTML документите и се изпълнява на сървъра преди клиентът да получи страницата, за която е дал заявка. Този стандарт се поддържа почти само от Microsoft Internet Information Server (IIS), който има лошата слава, че е слабо надежден и лесно пробиваем. Въпреки това най-после версия 5.0 на IIS може да се смята вече за стабилна (все пак www.microsoft.com работи с нея). Едва 20% от сървърите в ИНТЕРНЕТ работят с IIS.

Подобна технология за генериране на страници с динамично съдържание е PHP. PHP е хипертекстов препроцесор, който позволява да се пишат скриптове в HTML документите. Той се инсталира към web-сървъра и когато клиентът даде заявка за някоя страница, сървърът я взема, замества всички скриптове с резултата от изпълнението им и връща генерираната по този начин динамична страница. Методът се отличава с голяма бързина, понеже скриптовете се интерпретират от препроцесора на PHP без да се прави нов процес. Интеграцията със сървъра е добра. PHP е предназначен най-вече за сървъра Apache, където се реализира като модул подобно на mod_perl. Съществува вариант на PHP за IIS, но там е реализиран като CGI-програма и, разбира се, е по-бавен.

Netscape Enterprise Server използва server-side JavaScript като алтернатива на CGI и ASP. Механизмът е същият. Програмния код се вгражда направо в HTML документа и се интерпретира от сървъра.

Друга подобна алтернатива е Cold Fusion. Чрез собствени тагове в HTML документите се вгражда програмен скрипт-код. Документите се интерпретират от Application-сървъра на Cold Fusion, който се предлага за повечето популярни web-сървъри. Технологията предлага изключително лесен и прост достъп до бази от данни, без да се налага много писане на код.

Същата идеята за вграждане на код в HTML документа е реализирана в JSP (Java Server Pages), само че кодът се пише на JAVA, което дава голяма гъвкавост на приложението, за сметка на повече писане. Повече за JSP четете в края на статията.

Използването на JAVA като server-side език за програмиране в ИНТЕРНЕТ за генериране на динамични web-страници в последните години претърпя бурно развитие и доби голяма популярност. Една от технологиите, които се утвърдиха е програмирането с JAVA сървлети. Първо трябва да изясним, че сървлетите не са аплети. Аплетите са програми на JAVA, които се изпълняват на машината на клиента от неговия web-браузър вътре в самия HTML документ, който той разглежда. За разлика от тях JAVA сървлетите са алтернатива на CGI, която предлага начин за генериране на динамични HTML страници чрез програми написани на JAVA, които се изпълняват на сървъра. Сървлетът представлява компилиран JAVA клас (програма на JAVA), който стои в специална директория на сървъра и се извиква от клиентския браузър чрез заявка за получаване на страница или чрез submit-ване на HTML форма. Сървлетът обработва подадените му параметри и връща като резултат динамично генериран HTML документ. Механизмът е подобен на извикването на програма чрез CGI, но за разлика от него е по-бърз.

За да използваме сървлети, ни е необходим сървър, който ги поддържа (Java Web Server, Netscape Enterprise Server и др.) или трябва да инсталираме специални разширения (plug-ins) към сървъра, който използваме (такива са достъпни за повечето популярни web-сървъри като Apache и IIS).

Както езикът JAVA, така и JAVA сървлетите са проектирани с цел да позволят преносимост. Те се поддържат на всички платформи, които поддържат JAVA. Като изобретение на Java Software (подразделение на Sun Microsystems) те са разширение на вградената във

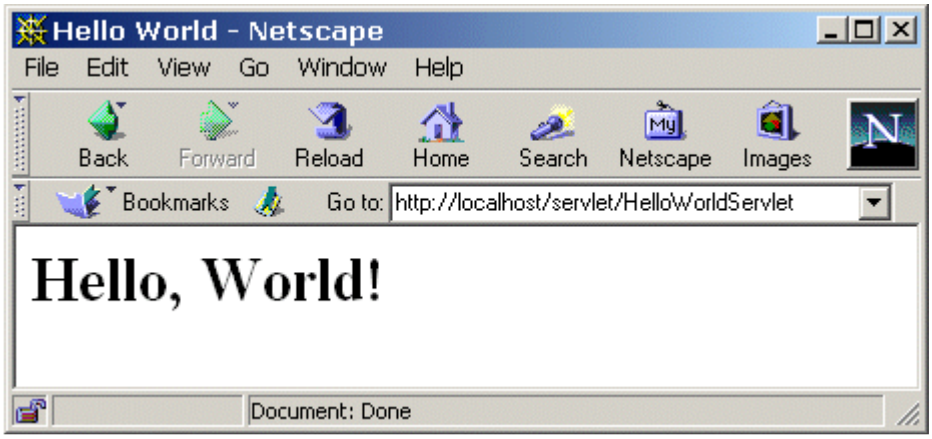
виртуалните машини на JAVA стандартна библиотека с класове JDK, но не са част от нея. Библиотеката за поддръжка на сървлети се нарича JSDK и е достъпна за изтегляне от http://java.sun.com/products/servlet/. Основните качества на технологията са преносимост, мощност, издръжливост, надеждност, разширяемост, интеграция. За голяма бързина не може да се говори, понеже езикът JAVA по принцип се интерпретира от въртуалната машина, а този процес е доста бавен в сравнение с компилирана до изпълним код програма. Освен това езикът JAVA е от по-високо ниво в сравнение с C++ и други езици за програмиране, което, разбира се, го прави доста по-бавен от тях. Независимо че самите сървлети са по-бавни от компилиран изпълним код, понеже са програми на JAVA, стартирането на сървлет е бърза операция, защото сървлетите се зареждат в паметта на сървъра като обект веднъж, при първото им стартиране и след това само се извикват, което става почти моментално. Едновременното изпълняване на няколко заявки към един и същ сървлет се извършва в отделни thread-ове едновременно. Сървлетите са напълно преносими. Можете да разработвате сървлети например под Windows NT с JDeveloper и да ги тествате с Java Web Server, а после да ги пренесете към крайното им местоназначение, например UNIX система с Apache. Поради мощността на езика JAVA, можете да направите всичко, което поискате – от най-прост брояч на посетителите до динамично генериране на изглед от географски карти или автоматичен междуезиков провод на текст. В тази насока JAVA сървлетите превъзхождат голяма част от останалите алтернативи за web-програмиране като Perl, PHP и server-side JavaScript. Относно стабилността и надежността, може да се каже, че сървлетите притежават голяма сигурност и надеждност, което идва от самия език JAVA, който притежава тези качества. Добре написан JAVA сървлет не може да бъде използван за неоторизирано проникване в сървъра, докато голяма част от CGI програмите, които са компилирани до изпълним код биха могли да се пробият (например чрез стандартната техника “buffer overflow”) и да се осъществи нелегално хакерско проникване на машината, последвано от кражба или унищожение на важна информация (например кредитни карти, фирмени документи, пароли и т.н.). Относно интеграцията можем да кажем, че понеже сървлетите се изпълняват вътрешно от сървъра, сървърът и сървлетите могат да си комуникират лесно, пожене в повечето случаи те са едно цяло. Технологията е лесно разширяема. С използването на нови допълнителни класове и чрез разширяване на Servlet API-то, може да се постигне поддържката на новите технологии и стандарти, който ще се появят в бъдеще.

Нека да погледнем как изглежда един прост JAVA сървлет:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorldServlet extends HttpServlet {
   public void doGet (HttpServletRequest req, HttpServletResponse resp)
                    throws ServletException, IOException {
      resp.setContentType("text/html");
      ServletOutputStream out = resp.getOutputStream();
      out.println("<HTML>");
      out.println("<HEAD><TITLE>Hello World Servlet</TITLE></HEAD>");
      out.println("<BODY>");
      out.println("<H1>Hello, World!</H1>");
      out.println("</BODY></HTML>");
   }
}
```

Този сървлет наследява и разширява стандартния за JSDK клас HttpServlet. Всеки път, когато сървърът получи GET-заявка за достъп до сървлета, той извиква doGet() метода му, като му подава обекти HttpServletRequest и HttpServletResponse, чрез които сървлетът може да взима параметрите на заявката и да създава отговора на сървъра. Показания по-горе сървлет игнорира параметрите и връща динамично генериран HTML документ, който изглежда по следния начин:

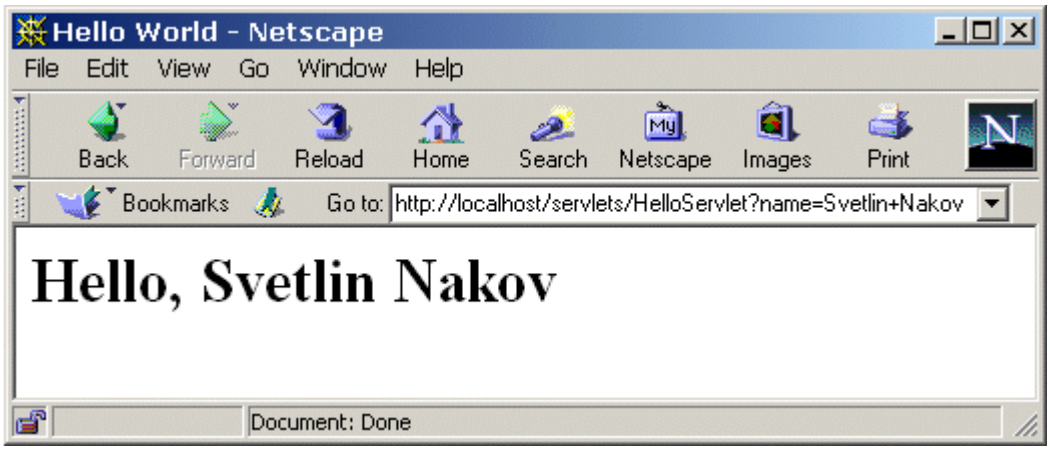


Следващият сървлет демонстрира леснотата с която се получават параметрите. Той получава като вход име на човек и му казва “здравей”. Разликите от предния са минимални.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloServlet extends HttpServlet {
   public void doGet (HttpServletRequest req, HttpServletResponse resp)
                  throws ServletException, IOException {
      resp.setContentType("text/html");
      ServletOutputStream out = resp.getOutputStream();
      String name = req.getParameter("name");
      out.println("<HTML>");
      out.println("<HEAD><TITLE>Hello Servlet</TITLE></HEAD>");
      out.println("<BODY>");
      out.println("<H1>Hello, " + name + "</H1>");
      out.println("</BODY></HTML>");
   }
}
```

Ето и резултатът от изпълнението му с параметър “Svetlin Nakov” :



В INTERNET често се срещат HTML форми, в които потребителят въвежда някаква информация и я подава на сървъра. Сървърът изпълнява някакъв скрипт, програма или сървлет, който обработва заявката и връща динамично генерирана от него страница на клиента. Ето един пример как бихме могли да се направим такава HTML форма:

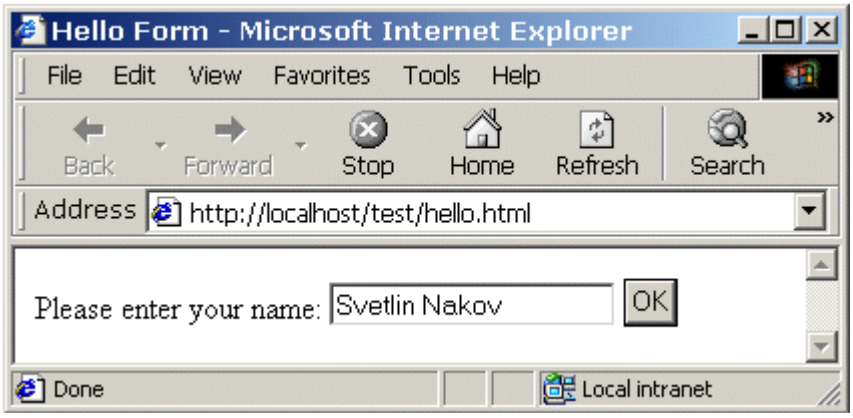
```
<HTML>
<HEAD> <TITLE> Hello Form </TITLE> </HEAD>
```

```
<BODY>

<FORM METHOD=GET ACTION="/servlets/HelloServlet">
Please enter your name:
<INPUT TYPE=TEXT NAME="name">
<INPUT TYPE=SUBMIT VALUE="OK">
</FORM>

</BODY>
</HTML>
```

Ето и резултатът, който потребителят вижда:



При натискане на бутона "OK" се извиква сървлетът за поздрав по име и му се подава параметъра "name", въведен в полето за име. В нашия случай натискането на този бутон е еквивалентно на поискване на адреса http://localhost/servlets/HelloServlet?name=Svetlin+Nakov. По подобен начин, чрез комбинация от HTML форми, сървлети и бази от данни, може да се направят сложни информационни web-базирани системи, които осъществяват има диалог с потребителя. Някои съсвъри поддържат специален таг <SERVLET>, като по този начин позволяват извикване на сървлет директно от HTML страница. Тази технология опростява и олекотява сървлетите, като разделя интерфейса от логиката. Почти всичко отнасящо се до интерфейса се оформя от webmaster-а, който създава .html файл с предварително създадения шаблон на HTML документа. Всичко отнасящо се до обработката на данните и извличането на динамичното съдържание се извършва от програмиста, който вместо да връща цял HTML документ, връща само една част от него. Програмистът пише програмата, компилира я до .class файл и я записва в съответната директория на сървъра. Проблеми има при тестването, понеже както изяснихме по-горе, сървлетът се зарежда само веднъж, а след това се извиква от паметта на сървъра. В този случай ако програмистът промени сървлета, сървърът няма да го прочете наново и промените няма да се отразят. В такива случаи програмистът трябва да рестартира или целия сървър или само частта от него, която поддържа сървлетите или, ако сървърът позволява, само въпросния сървлет. Ето и един пример за вграждане на <SERVLET> таг в HTML документ. Нека нашата страница се казва "test.shtml". Забележете разширението ".shtml". То подсказва на сървъра, че страницата съдържа сървлети, които трябва да се изпълнят и да се заместят с върнатото от тях:

```
<HTML>
<BODY>

This page is visited
<SERVLET CODE="CounterServlet"> </SERVLET>
times.

</BODY>
</HTML>
```

Ето и кода на CounterServlet програмката, която извършва съвсем просто (и малко некачествено) преброяване на посетителите:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorldServlet extends HttpServlet {

   static int count = 0;

   public void doGet (HttpServletRequest req, HttpServletResponse resp)
                  throws ServletException, IOException {
      resp.setContentType("text/html");
      ServletOutputStream out = resp.getOutputStream();
      out.println(++count);
   }
}
```

Няма да разглеждаме по-подробно практическото програмирането с JAVA сървлети, понеже целта на настоящата статия е да запозне читателите с технологията, като цяло и какво тя предлага, без да набляга на детайлите. Повече информацията за програмирането с JAVA сървлети можете да получите като посетите следните адреси:
www.servlets.com, developer.java.sun.com/developer/onlineTraining/Servlets/Fundamentals/index.html

За нас българите един от най-честите проблеми е поддръжката на кирилица. Дали документите и заявките на кирилица ще се обработват правилно зависи донякъде от сървъра и донякъде от програмиста. Препоръчва се използване на сървър, който е пригоден за работа с кирилица, например Apache (http://www.apache.org), който има руска версия. За останалите сървъри, обикновено поддръжката на кирилица се установява с подходящи настройки. Проблеми с кирилицата има и при достъп до сървъри за бази от данни. Проблемът отново се решава с подходящи настройки на сървъра и драйверите за достъп. Всички популярни сървъри за бази от данни поддържат кирилица. Трябва да се отбележи, че проблемите с кирилица не се отнасят само за JAVA сървлетите, а също и за голяма част от сървърите и технологиите, които генерират динамичен HTML. Важно е да се отбележи, че при предаване на параметрите към сървлет, програмистът може да използва следната програмна хитрост, за да преодолее проблемите с кирилица: [1]

```
String name = request.getParameter("name");
try {  // Decode the cyrillic parameter value
   name = java.net.URLDecoder.decode(name);
} catch (Exception e) {}
```

С използването на този код се постига правилното разкодиране на параметъра “name”, като се избягват проблемите с кирилицата и други нелатински абзуки.

В обобщение трябва да отбележим кога е подходящо да се използват JAVA сървлети и кога е по-добре да се използват други технологии. Сървлетите ни осигуряват голяма мощност и функционалност, голяма сигурност и надеждност, а също и преносимост. Подходящи са за големи проекти, където трябва да се извършва сложна обработка на данни, за да се получи необходимия динамичен резултат, а също и когато надеждността е от съществено значение. Трябва да отбележим, че дори за прости неща на JAVA трябва да се изпише доста програмен код, за разлика от някои други езици за програмиране, подходящи за web-разработка. Като добавим, че JAVA е слаб и бавен език при текстообработката, лесно се съобразява, че за малки и прости програмки е по-подходящ Perl или PHP. Когато става въпрос за достъп до бази от данни, JAVA сървлетите се справят добре, но отново това става повече труд и много писане на програмен код, в сравнение с предназначените специално за това сървърни разширения, като например ColdFusion и Oracle

[1] Авторът благодари за предоставеният код и редакторските забележки на Димитър Проданов

Application Server. Така че за прости неща не мога горещо да препоръчам сървлетната технология. Напоследък Sun се опитва да наложи Java Server Pages (JSP) технологията в добавка към сървлетната. Както вече отбелязахме JSP технологията позволява вграждане на JAVA код директно в HTML документи посредством специални тагове. Страниците с JSP се обработват от специален JSP-engine, който ги прочита, преобразува ги в JAVA сорс код, компилира ги и ги изпълнява. Бързодействието се постига от това, че този процес се извършва еднократно – при първото зареждане на страницата с JSP код, а след това компилираният JAVA код се изпълнява като нормален JAVA сървлет. Един недостатък на технологията е че при нея HTML текстът се вгражда като константи в програмния код, а не е в чист вид. Това, например в PHP, е преодоляно чрез възможността за съчетаване на чист HTML тескт и фрагменти програмен код в един и същ файл.

Използвана литература:

1. Hunter J., “Java Servlet Programming”, O’Reilly, Sebastopol, USA, 1998
2. Zeiger S., “Servlet Essentials”, http://www.novocode.com/doc/servlet-essentials/
3. Fischer D. “Oracle Java Roadmap: Java Servlets”, http://technet.oracle.com/tech/java/servlets/