Интернет програмиране с Java – част 6

Автори:

Светлин Наков, СУ "Св. Климент Охридски"
Web-site: http://www.nakov.com

Борис Червенков, "Информационно обслужване" АД
E-mail: boris@abv.bg

Последна промяна: 18.09.2002

В предходната част от курса започнахме темата за създаването на Web-приложения с Java. В тази част ще продължим с тази тема. Ще изясним основните концепции в Web-програмирането, ще обясним какво е Java сървлет, как се създават и как се изпълняват сървлети със сървъра Tomcat.

Основни концепции в Web-програмирането

Всички сме виждали Web-базирани e-mail системи като mail.yahoo.com, mail.bg и abv.bg. Те са чудесни примери за Web-приложения. Както знаем от предходната част на настоящия задочен курс, Web-приложенията представляват програмни системи, които работят на някакъв Web-сървър и предоставят на потребителите Web-базиран интерфейс. Комуникацията между потребителите и Web-приложенията се основава на заявки и отговори и се извършва по протокол HTTP. Когато потребителят напише адреса на някое Web-приложение, неговият Web-браузър изпраща на съответния Web-сървър заявка за достъп до това Web-приложение и получава динамично генериран отговор във вид на HTML или друг формат, който браузърът разбира. Ще се опитаме да представим различните аспекти на Web-програмирането, неговите предимства и недостатъци. Също така ще направим една уговорка – като средства за изграждане на потребителския интерфейс на Web-приложенията ще имаме предвид само HTML, CSS и JavaScript. Технологии като Flash, ActiveX и Java-аплети няма да бъдат разглеждани в останалата част от този курс. Можем да разделим условно Java-базираните Web-приложения на две-части:

- сървърска част – представлява съвкупност от Java сървлети и JSP-та, които обработват получените от потребителя данни и взависимост от тях динамично генерират HTML документи, CSS стилове и JavaScript код
- клиентска част – представлява съвкупността от динамично генерираните HTML документи, CSS и JavaScript код, които се визуализират от Web-браузъра и изграждат потребителския интерфейс на приложението

Разглеждайки Web-приложенията по този начин, можем да ги определим като многопотребителски клиент-сървър приложения, предназначени за работа в Интернет или Интранет.

HTML

HTML (Hyper Text Markup Language) е създаден като част от WWW (World Wide Web) от Тим Бернерс-Лий в началото на 90-те години. HTML е базиран на SGML (Standard Generalized Markup Language – стандартен формат за представяне на текст, широко използван от американското правителство) и наследява неговия синтаксис. HTML не е програмен, а описателен език за представяне на форматиран текст. HTML документите представляват изцяло текстови файлове, като в тях освен текста, който съдържат, са вмъкнати инструкции за форматиране (наречени тагове), които указват как точно да се изобрази текста, по време на визуализацията. В HTML документите могат да се указват връзки (hyperlinks) към произволни отдалечени ресурси. Въпреки че в последно време HTML претърпя доста бурно развитие, което доведе до неговото

усложняване, основната му сила си остава неговата простота. Когато се използва в комбинация с различни технологии като JavaScript и CSS, езикът HTML предлага доста богати възможности за реализиране на потребителския интерфейс на сложни Web-приложения.

CSS

Cascading Style Sheets е допълнение към HTML. Разработен е от W3C (World Wide Web Consortium) и представлява език за описание на атрибутите и позиционирането на елементите на HTML документи. Чрез CSS се дефинират стилове, които се използват след това в HTML документите за форматиране на текста. При необходимост форматирането на един HTML документ, използващ CSS, може бързо и лесно да се промени, като се променят само стиловете в CSS файла, без да се променя HTML файла.

JavaScript

JavaScript е сравнително прост скриптов език, който се изпълнява от Web-браузъра на потребителя и позволява динамична манипулация на обектите в HTML документите. С негова помощ е възможно създаването на сложни по функционалност и интересни Web-страници. Първоначално е разработен от Netscape, но в момента се поддържа в една или друга степен от всички браузъри.

Разпределеност и платформена независимост на Web-приложенията

Най-голямото предимство на Web-програмирането е, че Web-приложенията са достъпни от различни компютри, работещи под различни операционни системи и различни браузъри, като единственото условие е те да имат достъп до Web-сървъра, където работи съответното Web-приложение. Всичко което е необходимо на потребителя е връзка с Интернет и стандартен Web-браузър. Поради независимостта от операционната система на потребителя и относителната независимост от неговия Web-браузър, Web-програмирането улеснява работата на софтуерните разработчици, като им дава възможност да поддържат само една версия на приложението, която работи на всички платформи, а не отделни версии за всяка отделна платформа. Като се има предвид разнообразието от операционни системи в Web-пространството, това че се поддържа само една версия на приложението е едно серизно предимство пред другите видове програмиране.

Понякога, в стремежа си да създадат по-сложни Web-приложения програмистите се възползват от възможностите на определен браузър. Често срещана практика е да се оптимизира едно приложение за Microsoft Internet Explorer. Той поддържа голяма част от общоприетите спецификации за HTML и CSS, но има и множество специфични, нестандартни тагове и стилове. Ако се използват нестандартните стилове и тагове, има голям риск приложението да не работи с други браузъри. Доста е трудно да се направи нещо сложно с JavaScript, което да работи еднакво добре както на Internet Explorer, така и на Netscape Navigator и Opera. Подобен проблем се появява понякога и от това, че самият HTML не изглежда еднакво на всички браузъри, макар и да не са използвани нестандартните възможности на някой конкретен браузър.

Допълниетелно улеснение на Java разработчиците е платформената независимост на Java. Благодарение на нея не само клиентската, но и сървърската част на Web-приложенията става напълно платформено независима. Вградените възможности на Java за унифициран достъп до бази данни през JDBC или EJB допълнително улесняват работата на програмиста и го правят по-малко обвързан с базата данни, която използва.

Друго голямо предимство на Web-програмирането е неговата разпределеност. Поради разпределеността на WWW, разработката на разпределени Web-приложения е доста лесно. Достатъчно е приложението да се раздели на компоненти и всеки от тях да се разположи на различен Web-сървър. Така отделните компоненти, въпреки че са физически разделени, могат да работят като една цяла система. Използвайки съвременни технологии за балансиране на

натоварването и възстановяване от грешки, надеждността на Web-приложенията става изключително голяма, като това не коства много допълнителни усилия за програмиста.

Несесийност на HTTP протокола

По принцип повечето Web-приложения са предназначени за многопотребителски достъп. Използването на протокола HTTP в Web-програмирането създава проблеми в тази насока, защото има несесиен характер, т.е. няма вградена възможност за идентификация и проследяване на потребителските сесии. Първоначално HTTP е служел за достъп до статични ресурси (HTML файлове, изображения, др.) и за тази си функция е бил чудесен. В днешно време HTTP се използва за много други неща, които не са били предвидени при създаването му и за които не е много удобен, но се използва, защото се е наложил като стандарт. Въпреки че версия 1.1 на HTTP протокола въвежда така наречените keep-alive връзки (възможност за изпълнение на няколко HTTP заявки през веднъж осъществена връзка между клиента и сървъра), това не решава проблема с еднозначната идентификация на потребителя от страна сървърската част на Web-приложението. За щастие в J2EE средствата за осигуряване на многопотребителски достъп се дават стандартно от платформата, поради което за да е възможно няколко потребителя да работят едновременно и независимо един от друг с едно Web-приложение, от програмиста не се изискват никакви допълнителни усилия. Достатъчно е да се използват съответните стандартни обекти за работа с потребителска сесия, които се дават от контейнера за приложения.

Производителност

Производителността е един от големите плюсове на Web-програмирането. При този вид програмиране бизнес операциите на системата са изнесени на сървъва, като по този начин ролята на клиента се свежда до това да обработва и визуализира HTML документи. И тъй като HTML е сравнително прост описателен език, изискванията и натоварването на клиентската машина са минимални.

Сигурност

Трябва да отбележим несъмненото превъзходство на Web-програмирането пред конвенционалното програмиране по отношение на сигурността.

От страна на клиента

В компютърната индустрия вирусите, които се разпространяват с изпълнимите програми, са нанесли щети на много хора и корпорации. Използването на антивирусен софтуер не винаги спасява потребителя от действието на злонамерени програми, а освен това забавя бързодействието на компютъра му. За разлика от настолният софтуер, Web-приложенията са напълно безопасни за потребителя, тъй като при тях клиентската част се състои във визуализирането на документи, описани с HTML/CSS, и изпълнението на JavaScript.

От страна на сървъра

Предимството на Web-приложенията по отношение на сигурността се проявява най-вече от гледна точка на сървъра. Голям брой клиенти могат да имат достъп до някакъв ресурс на сървъра (база данни, изчислителна мощ, т.н.), без да могат да го достъпват директно, а само посредством някое Web-приложение. Съответното Web-приложение може да дава достъп до определени ресурси само след автентикация на достъпа с парола, клиентски сертификат или друга технология. Въвеждането на HTTPS като стандарт прави невъзможно подслушването на конфиденциална информация и дава голяма сигурност на Web-приложенията.

Неодстатъци на Web-програмирането

Първият недостатък на Web-приложенията се отнася до производителността. Бавната връзка между клиента и сървъра води до ниска производителност на цялата система. Макар и сървърът да обработва и отговаря мигновенно на клиентските заявки, потребителите често имат усещането, че системата, с която работят, е бавна заради забавянето, което е необходимо за пренос на данните между сървъра и клиента. Заради това забавяне Web-приложенията трябва да се разработват внимателно, особено ако се очакват потребители с лоша Интернет връзка.

Другият проблем за Web-програмирането е статичният характер на HTML. Въпреки че JavaScript донякъде решава този проблем, разработката на сложен, интерактивен и удобен потребителски интерфейс често пъти е много трудна задача, която изисква сериозни усилия и творчество. При силно интерактивни Web-приложения обикновено има проблеми с различните Web-браузъри и различните версии на един и същ браузър. С HTML в комбинация с CSS и JavaScript не може да се направи всичко, дори ако програмиста показва завидно майсторство. Понякога изискванията на крайния потребител относно потребителския интерфейс се указват просто неосъществими и се налага да бъдат променяни, за да стане възможно изпълнението им. Поради факта, че Web-приложенията не са подходящи за създаване на сложен потребителски интерфейс, обикновено интерфейсът на Web-базирания софтуер е прост, макар и достатъчно функционален.

Друг, много сериозен проблем на Web-програмирането е еднопосочната връзка между потребителя и клиента. Web-програмирането е базирано на механизма “заявка-отговор”, който непозволява на сървъра да изпраща данни на клиента без негова заявка. Това силно затруднява някои интерактивни приложения, които разчитат на получаване на данни асинхронно от сървъра. Пример за такива приложения са приложенията за разговори (chat), които често пъти се реализират с аплети, поради слабостта на HTML, CSS и JavaScript. Докато не се намери добро решение на проблема с еднопосочността на комуникацията, Web-приложенията никога няма да достигнат функционалността на настолните приложения.

Защо Web-програмирането е толкова разпространено

Въпреки всичките изложени недостатъци, Web-програмирането си остава един от най-предпочитаните подходи за разработка на големи многопотребителски приложения. Най-важната причина за това големите компании да предпочитат Web-базирани приложения, е че поддръжката на Web-приложенията е значително по-лесна, отколкото на настолните. Преминаването към нова версия на едно Web-приложение става без инсталиране на нищо допълнително. Просто се подменя сървърската част на приложението и всички потребители (които понякога може да са милиони, дори десетки милиони) не трябва да правят абсолютно нищо, за да преминат на новата версия. Достатъчно е да заредят отново адреса на Web-приложението от своя Web-браузър и започват работа с новата система.

Java базирани Web-приложения

Едно Java базирано Web-приложение представлява съвкупност от сървлети, JSP-та (разширение на сървлетите, което позволява в HTML документи да се вгражда Java код), Java архиви и други файлове, които заедно изграждат една обща Web-система. Файловете на приложението се разполагат в поддиректориите на една директория с фиксирана структура, която се задава от J2EE спцификацията. JSP файловете се разполагат в главната директория на приложението. Настройките на приложението се задават в специален файл с име web.xml намиращ се в поддиректория WEB-INF. Класовете, които приложението използва се разполагат в поддиректория WEB-INF\classes. Java архивите, които могат да съдържат класове, изображения и други ресурси се разполагат в поддиректория WEB-INF\lib. Така сървърът, който изпълнява приложението (Web-контейнерът) знае къде да търси различните файлове, когато му потрябват.

Java Сървлети

Java платформата дава няколко стандартни средства за създаване на динамични Web-страници. Основната технология, на базата на която се изгражда всичко останало са ***сървлетите***. Сървлетите представляват програми на Java, които приемат като вход някакви данни от потребителя, обработват ги и връщат като резултат динамично генериран HTML или друг документ. Например, един сървлет може да приема като входни данни име на потребител и парола, да проверява валидността им по някакъв начин и да пренасочва браузъра към друга страница от Web-приложението, ако са валидни или да връща съобщение за грешка в противен случай.

Предимства на сървлетите

От гледна точка на ефективността сървлетите превъзхождат стандартната CGI технология, защото изпълнението на сървлет не води до създаване на нов процес в операционната система, което е традиционно бавна операция. Java виртуалната машина стои постоянно заредена в паметта и когато се извика някой сървлет, той просто се изпълнява и резултатът се връща на клиента, без да се създава нов процес за обработка на заявката. Освен това, веднъж изпълнен, сървлетът остава в компилиран вид активен в паметта, чакайки ново извикване. Това допълнително повишава производителността. По подобен начин работят повечето съвременни Web-технологии като PHP, ASP и Perl. При тези технологии обикновено интерпретаторът на съответния скрипт език стои като модул зареден постоянно в паметта на Web-сървъра и се включва при клиентска заявка. Така производителността зависи от производителността на интерпретатора. Заради традиционната не много добра производителност на Java виртуалната машина, може да се спори дали JSP/сървлет технологията е по-бърза или по-бавна от останалите, но при всички случаи скоростта й не се различава значително от тази на конкурентните технологии.

Съвкупността от средства (framework-ът) за работа със сървлети и JSP-та, който ни дава J2EE платформата предлага доста вградени удобства. Едно от тях е вече споменатото автоматично управление на многопотребителския достъп, което дава възможност за съхранение на различни данни за потребителя в рамките на неговата сесия. Освен това извличането и декодирането на параметрите е силно улеснено, дава се възможност за достъп на по-високо ниво до HTTP хедърите, за пренасочване на потребителския браузър, за достъп до общи за приложението данни, за обмяна на данни между приложенията, както и достъп до ресурси, принадлежащи на Web-контейнера (сървърът, изпълняващ сървлетите и JSP-тата).

Java базираните Web-приложения са изключително лесно преносими. В повечето случаи всичко, което е необходимо за да се прехвърли едно приложение от един сървър на друг, е да се прехвърли един единствен файл. Дори новият сървър да е от друг производител и да работи на друга операционна система, рядко се налага да се извършват промени или допълнителни настройки за да заработи Web-приложението на новия сървър. Тази изключително добра съвместимост се дължи на стандартите за Web-приложения, които се дават от платформата J2EE и се спазват стриктно от почти всички производители на Web-приложения и Web-контейнери.

Сигурността и надеждността на Java базираните Web-приложения е изключително голяма. Това се дължи на надеждността и сигурността на самия език Java. Уязвимости като препълване на буфери са изключени, а проблеми с лошо декодиране на данни, неправилна работа с базата данни, непозволен достъп до паметта и още много други при Java платформата са значително по-трудно осъществими, отколкото при другите платформи.

Друго важно предимство на Java сървлетите, JSP-тата и Java-базираните Web-приложения е че те могат да се използват напълно безплатно. За работата им не е необходимо закупуването на скъп сървърски софтуер, защото има достатъчно добри безплатни Web-контейнери и J2EE сървъри за приложения (application servers), които са идеални за малкия и средния бизнес. Пример за такива безплатни сървъри са Web-контейнерът Tomcat (http://jakarta.apache.org), който ще разгледаме след малко и сървърът JBoss (http://www.jboss.org), който е почти пълна имплементация на J2EE платформата.

Структура на сървлетите

За създаването на Java сървлет е необходимо да се наследи класа `javax.servlet.http.HttpServlet` и да се припокрие метода `doGet()` или `doPost()` ако сървлетът ще обработва съответно GET или POST HTTP заявки. И двата метода doGet() и doPost() приемат като аргументи два обекта `HttpServletRequest` и `HttpServletResponse`. HttpServletRequest служи за извличане на входните параметри на сървлета – данните от HTML форми, изпратени от потребителския Web-браузър, хедърите на HTTP заявката, информация за клиента, неговия IP адрес и браузър. HttpServletResponse служи за изпращане на данни в отговор на получената HTTP заявка. Позволява задаване на кода на резултата и полетата в хедъра на HTTP отговора, както и разбира се, самия текст на отговора. При простите сървлети по-голямата част от сървлета представлява код, който отпечатва динамично създадения HTML документ в изходния поток на заявката. При по-сложните сървлети се налага работа с HTTP хедърите, работа с cookies и други, които ще разгледаме по-късно. Не всички сървлетите са HTTP сървлети. Има сървлети, които обслужват други протоколи и за реализацията им се наследява класа `GenericServlet`, а не `HttpServlet`. В нашия курс ще бъдат разглеждани само HTTP сървлети. Да дадем пример за прост HTTP сървлет:

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

public class NakovFirstServlet extends HttpServlet {
    public void doGet(HttpServletRequest request,
        HttpServletResponse response)
    throws ServletException, IOException {
        PrintWriter out = response.getWriter();
        out.println("<HTML>");
        out.println("The time is: " +
            new java.util.Date());
        out.println("</HTML>");
    }
}
```

Всичко, което прави дадения сървлет `NakovFirstServlet` е да наследи класа `HttpServlet` и в метода `doGet()` да получи изходния поток на отговора на заявката и да отпечата в него съвсем прост HTML документ, който се състои от 3 реда и съдържа текущата дата и час.

Работа със сървъра Tomcat

Сървърът Tomcat представлява безплатен Web-контейнер, който може да изпълнява Java сървлети, JSP-та и Web-приложения. Tomcat е Web-сървър, написан на Java, който освен статични файлове, може да връща и динамични документи, създадени в резултат от изпълнението на сървлет или JSP.

Инсталиране на Tomcat

Сървърът Tomcat е достъпен от адрес http://jakarta.apache.org/. След като издърпате последната версия, която представлява един .zip файл, го трябва да го разархивирате в някоя директория. Препоръчва се в името на директорията да не се съдържат интервали, защото интервалите служат за разделители в Java и могат да създадат някои досадни проблеми. Ако не сте инсталирали JDK на вашия компютър, инсталирайте последната версия. Можете да я намерите на адрес http://java.sun.com/j2se/. Добавете в променливите на средата променливата `JAVA_HOME` със стойност директорията където е инсталирана JDK. Това може да стане например от конзолата с командата "`set JAVA_HOME=C:\jdk1.3.1`". Намерете директорията, в която сте инсталирали Tomcat. Да предположим, че това е `C:\Tomcat`. В поддиректория `bin` има файл за стартиране `startup.bat` (`startup.sh` за Unix/Linux), с който можете да стартирате сървъра. Стартирайте този файл. Това е всичко необходимо за стартиране на сървъра. За да проверите дали работи, стартирайте вашия

Web-браузър и въведете адреса http://localhost:8080/. Ако всичко е наред, ще се появи заглавната страница на Tomcat. Можете да тествате и стандартните примерни сървлети и JSP-та, като следвате линковете от главната страница. Забележете, че по подразбиране Tomcat работи на порт 8080, а не на стандартния за протокола HTTP порт 80.

Как да стартираме нашия сървлет

Освен поддиректорията `bin`, в директорията на Tomcat има още една важна поддиректория – `webapps`. В тази поддиректория стоят всички инсталирани Web-приложения. Директория `webapps\ROOT` е главната виртуална директория на сървъра. За нашите тестови цели трябва да създадем нова поддиректория в `webapps`. Един полезен съвет при разучаване на непознат сървър е когато задавате имена на нови файлове и директории, имена на проекти, имена на услуги и т.н. да избирате нестандартно име, за да можете да различите след това стандартните неща от нещата, които вие сте създали. По този принцип не трябва да кръщавате новата директория за вашите тестове с имена като `test`, `samples`, `examples`, `tests`, `web`, `root` и т.н. защото рискувате първо избраното от вас име да съвпадне с някое служебно име със специално предназначение и второ ще ви е трудно да различите стандартните имена, идващи по подразбиране със сървъра, който разучавате, от вашите, които вие сте създали. Аз лично в такива случаи обикновено задавам за име "`nakov_directory`", "`nakov_service`" или нещо подобно, което ми подсказва че това име е избрано от мен и какво точно се крие зад това име – директория, услуга или нещо друго. Да предположим, че сме създали директорията "`nakov_test_dir`". За да изпълним нашия сървлет е необходимо да създадем поддиректория на новосъздадената с име `WEB-INF\classes`, да копираме некомпилирания сървлет в нея и да го компилираме. Ако сме инсталирали Tomcat в `C:\Tomcat`, а за нашиите тестове сме избрали директория `nakov_test_dir`, ще трябва да запишем нашия тестов сървлет `NakovFirstServlet.java` в директория `C:\Tomcat\webapps\nakov_test_dir\WEB_INF\classes`. За да го компилираме ще имаме нужда от пакета `javax.servlet.*`, защото той не се разпространява стандартно с JDK. Класовете от този пакет са включени в Tomcat и се намират във файла `C:\Tomcat\lib\servlet.jar`. След като компилираме нашия сървлет и получим файла `NakovFirstServlet.class` в директорията `C:\Tomcat\webapps\nakov_test_dir\WEB_INF\classes`, за да го изпълним, е необходимо да стартираме сървъра чрез скрипта `startup.bat` от `bin` директорията на Tomcat. По подразбиране в Tomcat всички файлове от главната директория на едно приложение се публикуват във виртуална директория с име името на приложението, а всички `.class` файлове от `WEB-INF\classes` директорията се публикуват като сървлети във виртуална директория с име `<име_на_приложението>/servlet`. Изключение прави специалната поддиректория `WEB-INF`, която остава недостъпна през браузъра. Така нашият сървлет е достъпен автоматично от адрес http://localhost:8080/nakov_test_dir/servlet/NakovFirstServlet, а всички файлове от директория `C:\Tomcat\webapps\nakov_test_dir\` – от адрес http://localhost:8080/nakov_test_dir/. Имайте предвид, че ако промените файловете от нашата тестова директория, промените ще са видими веднага, но ако промените и прекомпилирате тестовия сървлет, е необходимо да рестартирате Tomcat, за да влязат в сила промените. Това се обяснява с кеширането на заредените класове, което Tomcat прави с цел по-голяма производителност. Можете да тествате първия си сървлет с вашия Web-браузър. Ще получите нещо подобно на това: