

СЪРВЛЕТИ

Ненко Табаков

Пламен Танов

Технологическо училище “Електронни системи”

Технически университет – София

18 ноември 2008



ЛИТЕРАТУРА НЕОБХОДИМИ ПРОГРАМИ

- The Java EE 5 Tutorial - <http://java.sun.com/javaee/5/docs/tutorial/doc/JavaEETutorial.pdf>
- Java API документация - <http://java.sun.com/javase/6/docs/api/>
- Eclipse - www.eclipse.org
- Apache Tomcat - <http://tomcat.apache.org/>
- Step-by-step tutorial:
<http://www.java-tips.org/java-tutorials/tutorials/introduction-to-java-se1>

ВЪВЕДЕНИЕ

- Разширява възможностите на сървъри, които обработват клиенти посредством модел „заявка-отговор“
- Работят от страната на сървъра (server side)
- Обработват заявките на клиента и генерират отговор
- Използват се основно за WEB сървъри за генериране на динамично съдържание (HTTP Servlets)
- Решават някои от проблемите при използване на CGI (Common Gateway Interface)
- Платформено независими
- Основна технология на базата, на която се изграждат динамични WEB страници в Java

УПОТРЕБА

- Обработка и съхранение на данни, предадени от HTML форма
- Генериране на динамично съдържание
- Достъп до база данни и други ресурси
- Организация на сесии (в чист вид HTTP не поддържа сесия)
- Идентификация на потребители и нива на достъп (login)

КОНТЕЙНЕР ЗА СЪРВЛЕТИ (SERVLET CONTAINER)

- Всеки един HTTP сървлет съществува в контейнер за сървлети, който е част от WEB сървъра
- Той се грижи за работата на сървлета (стартиране, спиране), ниво на достъп (security)
- Такива сървъри са:
 - Apache Tomcat
 - Borland AppServer
 - IBM WebSphere Application Server
 - JBoss
 - Sun GlassFish Enterprise Server (преди: Sun Java System Application Server)
 - и много други
- Сървлетът се описва пред сървъра посредством дескриптора за внедряване (deployment descriptor)

СРАВНЕНИЕ С CGI

- CGI – интерфейс за стартиране на външни приложения, които да обработят заявка и да върнат отговор (най-често HTML)
- Сървлетите предоставят редица предимства пред CGI:
- Сървлетът не се извиква в отделен процес
- Сървлетът остава в паметта между отделните заявки (не се зарежда всеки път, за всяка заявка)
- За всеки сървлет се създава само една инстанция, която обработва всички клиенти едновременно (не се зареждат различни инстанции за отделните потребители и по-лесно се обработват общите данни)
- Може да бъде извикан в ограничена среда (sandbox), което позволява да се стартират потенциално опасни сървлети без да възникне проблем със сигурността

ПОСЛЕДОВАТЕЛНОСТ ПРИ РАБОТА СЪС СЪРВЛЕТ

- Сървърът зарежда сървлета
 - Прочита дескриптора за внедряване (deployment descriptor)
- Сървърът създава инстанция (обект) на класа на сървлета
- Сървърът извиква метода `init()` на всеки създаден обект
- Сървърът обработва пристигащите HTTP заявки:
 - Избира за кой сървлет е предназначена
 - Извиква метода `service()` на избрания сървлет
 - Връща отговора от извикването на `service()` на клиента
- Сървърът прекратява (unload) сървлета след като извика метода `destroy()`
 - Повечето сървъри изпълняват тази стъпка единствено при спирането си
 - Други следват политики за оптимизация – ако не е ползван дълго време и т.н.

ЙЕРАРХИЯ НА КЛАСОВЕ

- Сървлетите реализират интерфейса:
 - `javax.servlet.Servlet`
- Предоставени са следните базови класове:
 - `javax.servlet.GenericServlet`
 - Базов сървлет, независещ от протокола
 - `javax.servlet.http.HttpServlet`
 - HTTP сървлет

СТРУКТУРА НА СЪРВЛЕТ

- Метод `init()` – незадължителен
 - Първоначална инициализация на сървлета
 - Извиква се само веднъж за дадена инстанция на сървлет
- Метод `destroy()` – незадължителен
 - Освобождава използваните от сървлета ресурси
 - Извиква се единствено ако е извикан методът `init()`
 - Извиква се само веднъж за дадена инстанция на сървлет
 - При извикването му все още клиенти може да изпълняват `service()`, `doGet()`, ... (ако са много продължителни)
- Метод `service()`
 - Обработка заявка
 - Възможно е няколко нишки едновременно да извикват метода (няколко клиента едновременно правят заявка)

HTTP СЪРВЛЕТ

- Методът `service ()` в `HttpServlet` преглежда заявката и извиква подходящия Java метод `doXXX ()` на класа
- Всички тези методи имат два аргумента:
 - `HttpServletRequest request`
 - Съхранява заявката:
 - Параметри подадени към сървлета (от форми, през URL)
 - Използваният протокол (HTTP), хедъри, метод
 - Сесията на клиента и други данни за него (IP адрес)
 - Входен поток за достъп до данни на заявката (POST, PUT)
 - `HttpServletResponse response`
 - Чрез него сървлета връща отговор на клиента
 - Достъп до хедъри, `content-type`, съдържание (изходен поток)
- В общия случай няма причина да предефинираме метода `service ()`

HTTP СЪРВЛЕТ

- Поддържаните HTTP методи от `service ()` са:
 - GET – извиква метода `doGet ()` на класа
 - HEAD – извиква метода `doHead ()` на класа
 - PUT – извиква метода `doPut ()` на класа
 - POST – извиква метода `doPost ()` на класа
 - DELETE – извиква метода `doDelete ()` на класа
 - OPTIONS – извиква метода `doOptions ()` на класа
 - TRACE – извиква метода `doTrace ()` на класа
 - В случай на друг метод (ако сървърът го поддържа) – генерираният отговор е грешна заявка, код 400 (Bad Request)
- Ако съответния `doXXX ()` метод не се предефинира от класа наследник, то той връща същия отговор
 - Необходимите методи трябва да бъдат предефинирани

ПРИМЕР

```
package org.elsysbg.courses.it.servlet.intro;

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

public class HelloWorld extends HttpServlet {
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
                      throws IOException, ServletException {
        response.setContentType("text/html");
        final PrintWriter out = response.getWriter();
        out.println("<html><head>");
        out.println("<title>Hello World!</title></head>");
        out.println("<body><h1>Hello World!</h1></body>");
        out.println("</html>");
        out.close();
    }
}
```

HTTP СЪРВЛЕТ

- Отговорът на един HTTP сървлет може да не бъде HTML страница, а всякакви други двоични данни – изображения, аудио/видео, ...

```
public void doGet(...) throws ... {
    response.setContentType("image/jpeg");
    ServletOutputStream out = response.getOutputStream();

    String filename = getServletContext().getRealPath("/") +
                     "WEB-INF/images/image1.JPG";
    FileInputStream source = new FileInputStream(filename);
    byte[] data = new byte[BUFFER_SIZE];

    int count;
    while((count = source.read(data, 0, BUFFER_SIZE)) != -1) {
        out.write(data, 0, count);
    }
    source.close();
    out.close();
}
```

ПРЕДАВАНЕ НА ПАРАМЕТРИ

- Чрез URL:

- `http://localhost:8080/url_to_servlet?var1=val1&var2=val2`

- Чрез POST метод:

```
<form action='/url_to_servlet' method='post'>  
<input type='text' name='var1' value='val1' />  
<input type='text' name='var2' value='val2' />  
<input type='submit' />  
</form>
```

- Достъп до тях посредством:

```
String val1Value = request.getParameter("val1");  
String val2Value = request.getParameter("val2");
```

ПРЕПРАЩАНЕ НА ЗАЯВКА

- В отделни случаи е необходимо дадена заявка да бъде препратена към друга страница или сървлет
- Препращането може да се извърши по два начина:
 - Посредством HTTP код 302 (Moved Temporarily):

```
response.sendRedirect(getServletContext().getContextPath() +  
                        "/ShowPosts");
```

- Преди хедърите да бъдат изпратени
- На клиента се връща отговор:

```
HTTP/1.1 302 Moved Temporarily  
Server: Apache-Coyote/1.1  
Location: http://localhost:8081/SimpleForum/ShowPosts  
Content-Length: 0  
Date: Sun, 16 Nov 2008 12:41:19 GMT  
Connection: close
```

ПРЕПРАЩАНЕ НА ЗАЯВКА

- В отделни случаи е необходимо дадена заявка да бъде препратена към друга страница или сървлет
- Препращането може да се извърши по два начина:
 - Вътрешно, в контекста (по два начина):

```
request.getRequestDispatcher("/ShowPosts").forward(request,  
                                                response);  
// или  
request.getRequestDispatcher("/ShowPosts").include(request,  
                                                response);
```

- `forward()`
 - Целия отговор се генерира от избрания сървлет
 - Не трябва да бъде променян от текущия
- `include()`
 - Добавя отговора от избрания сървлет към отговора от текущия

ПРЕПРАЩАНЕ НА ЗАЯВКА

- В отделни случаи е необходимо дадена заявка да бъде препратена към друга страница или сървлет
- Препращането може да се извърши по два начина:
 - Вътрешно, в контекста, отговор:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/html
Content-Length: 260
Date: Sun, 16 Nov 2008 12:54:40 GMT
Connection: close
```

```
<html>
<head><title>Simple Forum</title></head>
<body>There are no posts there!
<form action='/SimpleForum/AddPost' method='post'>...
</form>
</body>
</html>
```

ЗАПИС НА ИНФОРМАЦИЯ ЗА СЕСИЯТА НА КЛИЕНТА

Достъпни са два механизма:

- Чрез Cookies
 - Дава възможност на сървлета да запише малка информация върху клиента
 - При всяка заявка клиентът връща тази информация
 - Съхранение на ID, на действия и други
- Чрез организиране на сесия
 - Дава възможност да се пази информация за клиента между отделните негови заявки за определен интервал от време

COOKIES

- <http://java.sun.com/javaee/5/docs/api/javax/servlet/http/Cookie.html>
- Информация, която е изпратена от сървлета до клиента
- Клиентът съхранява тази информация
- Всеки път, когато клиентът прави заявка до сървлета я изпраща обратно до сървъра
- Тази информация може еднозначно да описва клиента за това често се използва за организиране на сесия
- Има име, стойност и незадължителни атрибути: коментар, път, домейн, време за живот (0 – изтрива я), версия и други
- Не всички браузъри поддържат добре незадължителните атрибути

COOKIES

- Изпращат се от сървлет посредством:
 - `HttpServletResponse.addCookie (javax.servlet.http.Cookie)`
 - Добавя запис в хедъра на отговора
 - Трябва да се използва преди да бъдат изпратени хедърите
 - За да се изтрие `Cookie`, сървлетът установява нейното време за живот да е равно на 0:
 - `cookie.setMaxAge (0) ;`
 - Клиентът я изтрива и не я изпраща повече до сървлета
- Клиентът връща `cookies` посредством хедъри в заявката
- Достъпни са от сървлет посредством:
 - `HttpServletRequest.getCookies () ;`

ПОДДЪРЖАНЕ НА СЕСИЯ

- Дава възможност да се пази информация за клиента между отделните негови заявки за определен интервал от време
- Информацията се пази на сървъра
- Всяка сесия се определя посредством ID на сесията
- За съхранение на това ID между заявките се ползват два подхода:
 - Чрез cookies
 - Чрез добавяне на параметри в URL
- Ако браузърът поддържа cookies то те се използват за поддръжка на сесия, в противен случай – добавяне на параметри в URL
- По този начин при всяка заявка клиентът изпраща ID на сесията си на сървлета
- Всички сървлети имат достъп до сесията на клиента

ИЗПОЛЗВАНЕ НА СЕСИЯ

- Последователността е следната:
 - Взимане на сесията
 - `HttpSession session = request.getSession();`
 - За правилно функциониране трябва да се извиква преди да се променя отговора (преди да е изпратен)
 - Не създава сесия ако такава не съществува (връща `null`):
 - `request.getSession(false);`
 - Съхранение или извличане на информация от сесията
 - `session.setAttribute("number", 5);`
 - `Object number = session.getAttribute("number");`
 - Изтриване на сесията (`invalidate`), незадължителен
 - `session.invalidate();`
 - Автоматично при липса на действия за известен период

ПОДДЪРЖАНЕ НА СЕСИЯ

- Ако браузърът не приема cookies, то ID на сесията се предава посредством параметри в URL
- ID на сесията трябва да бъде добавена към ВСИЧКИ линкове в страницата посредством:
 - `HttpServletResponse.encodeURL()`
 - ```
out.println(
 String.format("link",
 response.encodeURL("/someLink")));
```
- Ако браузърът поддържа cookies то методът връща адреса непроменен, в противен случай добавя параметър

# ПОДДЪРЖАНЕ НА СЕСИЯ

- При препращана на заявка посредством HTTP код 302 (Moved Temporarily) за запазване на ID на сесия, когато браузърът не поддържа cookies **ВИНАГИ** се ползва:
  - `HttpServletResponse.encodeRedirectURL()`
  - `response.sendRedirect(  
    HttpServletResponse.encodeRedirectURL(  
        "someLink"))`;
- Ако браузърът поддържа cookies то методът връща адреса непроменен, в противен случай добавя параметър
- `encodeRedirectURL()` и `encodeURL()` използват различна логика за установяване на това дали браузърът поддържа cookies и за това трябва да се ползват в съответните случаи



# ФИЛТРИ

- <http://java.sun.com/javaee/5/docs/api/javax/servlet/Filter.html>
- Обект, който може да променя съдържанието както на заявката, така и на отговора
- Предоставят възможност за:
  - Идентификация на потребители и определяне на нива на достъп до даден ресурс (authorization)
  - Водене на протокол на събитията (logging)
  - Преобразуване на данни
  - Криптиране на съдържанието
  - Компресиране на съдържанието
  - Добавяне на съдържание (банери, брой посетители)

# ДЕСКРИПТОР ЗА ВНЕДРЯВАНЕ (DEPLOYMENT DESCRIPTOR)

- Файл web.xml указва как сървлетите да бъдат заредени от сървъра
- Състои се от множество секции вложени в корена на документа
- Секциите служат за описание на сървлет, връзката между сървлет и URL, страници при изключение, филтри и много други

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app ... id="WebApp_ID" version="2.5">
 <servlet>...</servlet>
 <servlet>...</servlet>

 <servlet-mapping>...</servlet-mapping>

 <error-page>...</error-page>
 <filter>...</filter>
</web-app>
```

# ДЕСКРИПТОР ЗА ВНЕДРЯВАНЕ (DEPLOYMENT DESCRIPTOR)

- Всеки сървлет трябва да се опише посредством:
  - Име
  - Клас

```
<servlet>
 <servlet-name>
 GetImage
 </servlet-name>
 <servlet-class>
 org.elsysbg.courses.it.servlet.intro.GetImage
 </servlet-class>
</servlet>
```

# ДЕСКРИПТОР ЗА ВНЕДРЯВАНЕ (DEPLOYMENT DESCRIPTOR)

• Връзката между сървлет и неговия адрес в рамките на сървъра се описва посредством:

- Име
  - Зададено в `<servlet-name>`
- URL

```
<servlet-mapping>
 <servlet-name>GetImage</servlet-name>
 <url-pattern>/image</url-pattern>
</servlet-mapping>
```

# ДЕСКРИПТОР ЗА ВНЕДРЯВАНЕ (DEPLOYMENT DESCRIPTOR)

- При възникване на изключение е възможно клиентът да се пренасочи към избрана страница в зависимост от типа на изключението
  - Тип на изключението
  - URL, към което да се пренасочи клиента

```
<error-page>
 <exception-type>
 java.lang.NullPointerException
 </exception-type>
 <location>/errorpage.html</location>
</error-page>
```