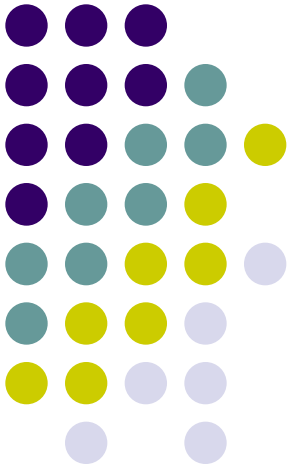
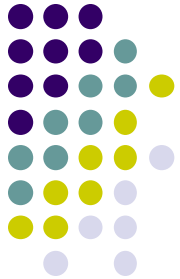
Прикладное программирование

Лекция 13
Сервлеты



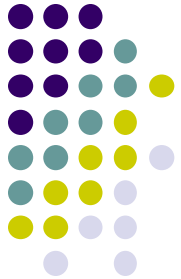
Курс читается при поддержке:

itransition
SOFTWARE DEVELOPMENT COMPANY



Сервлет (servlet) является Java-классом, получающим в качестве входного аргумента запрос и формирующим на его основе ответ. Обязательным условием является соответствие сервлета интерфейсу Java Servlet API.

Сервлеты могут создаваться непосредственно программистами, а могут автоматически конструироваться на основе JSP-страниц специализированным компилятором.



Спецификация Java Servlet API предоставляет стандартный и платформенно-независимый каркас для взаимодействия между web-контейнером и сервлетами, работающими внутри него. Этот каркас состоит из набора классов и интерфейсов Java, принадлежащих пакету javax.servlet, которые в сумме и представляют Servlet API.

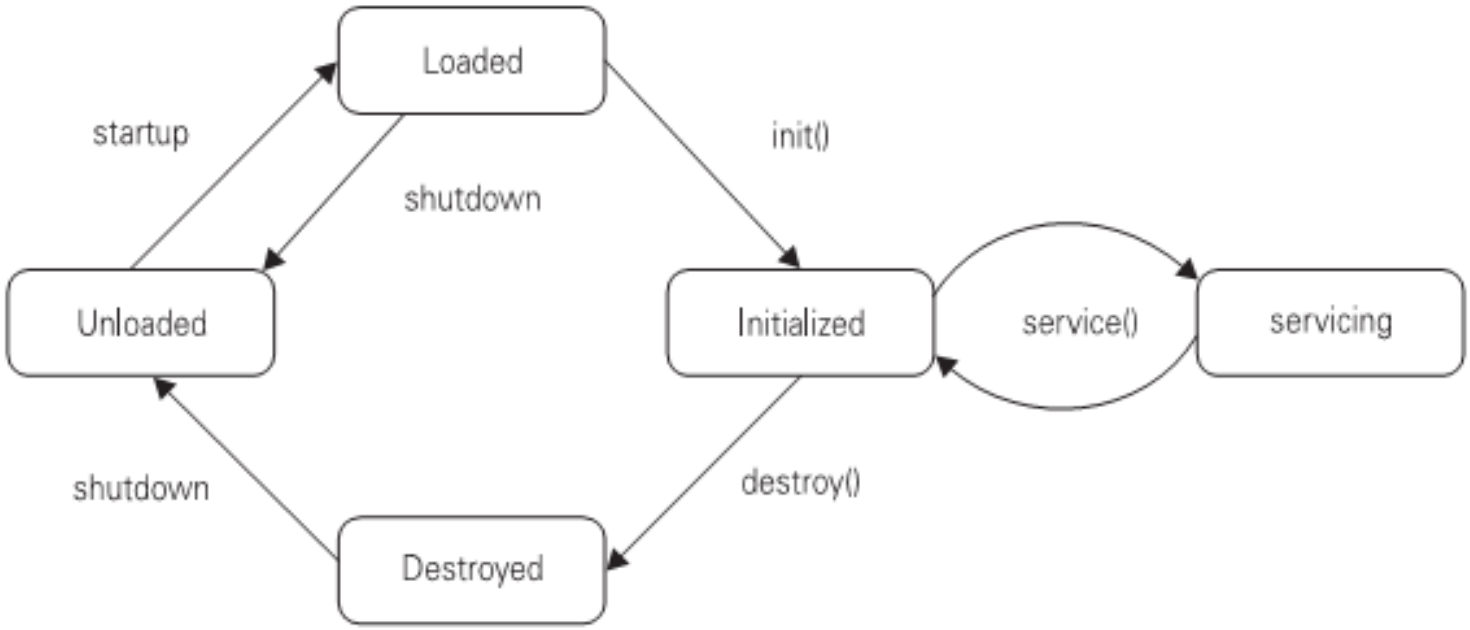
Web-контейнер отвечает за управление жизненным циклом сервлетов, отображение обращений к URL в вызовы определённых сервлетов, проверку прав доступа при обращении к сервлетам.

Методы интерфейса Servlet

Основным в Servlet API является интерфейс **Servlet**, и любой сервлет прямо или косвенно должен его реализовывать. Он включает пять методов:

Название метода	Описание
`init()`	Вызывается для уведомления сервлета о необходимости проинициализироваться и быть готовым к работе. В качестве параметра передаётся экземпляр класса **ServletConfig**.
`service()`	Вызывается для каждого поступившего от пользователя запроса, позволяя сервлету отреагировать на него.
`destroy()`	Метод вызывается для уведомления сервлета о необходимости освободить занимаемые ресурсы и приготовиться к выгрузке из контейнера.
`getServletConfig()`	Возвращает информацию о сервлете, такую как параметр для метода init().
`getServletInfo()`	Должен возвращать информацию о сервлете – авторе, версии, авторских правах.

Жизненный цикл сервлета

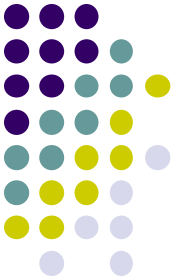


Контейнер создаёт экземпляр класса сервлета с использованием вызова

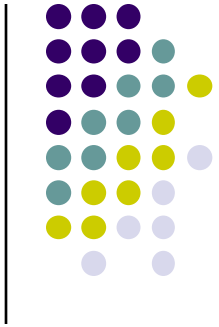
```
Class.forName(className).newInstance();
```

Для этого необходимо, чтобы класс сервлета имел `public`-конструктор без аргументов!

HttpServlet – реализация интерфейса Servlet для HTTP-протокола



Для разработки сервлетов, предназначенных для обработки HTTP-запросов, в рамках Servlet API представлен абстрактный класс **`HttpServlet`**. В данном классе определяется перегруженная версия метода **`service()`** для работы с HTTP-запросами и HTTP-ответами, а также определяется ряд методов, соответствующих методам протокола HTTP: **`doGet()`**, **`doDelete()`**, **`doOptions()`**, **`doPut()`**, **`doTrace()`**.



Существует два основных способа передачи данных от пользователя сервлету – с использованием методов GET и POST протокола HTTP.

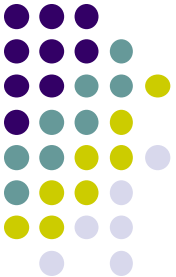
Характеристика	GET	POST
Тип запрашиваемого ресурса	Статический или динамический	Динамический
Тип передаваемых данных	Текст	Текст или двоичные данные
Объём данных	Хотя HTTP не ограничивает длину URL, некоторые браузеры и серверы могут видеть только 255 символов.	Ограничен только настройками web-сервера (заданными администратором)
Видимость	Данные являются частью URL и видны пользователю в строке адреса.	Данные не являются частью URL и отправляются в теле HTTP-запроса, т.е. не видны.
Кэширование	Данные могут кэшироваться браузером в истории посещений.	Данные не кэшируются браузером в истории посещений.

Информация о запросе пользователя – интерфейс HttpServletRequest

Интерфейс **`HttpServletRequest`** является средством доступа к параметрам запроса пользователя. Среди наиболее важных методов класса можно отметить:

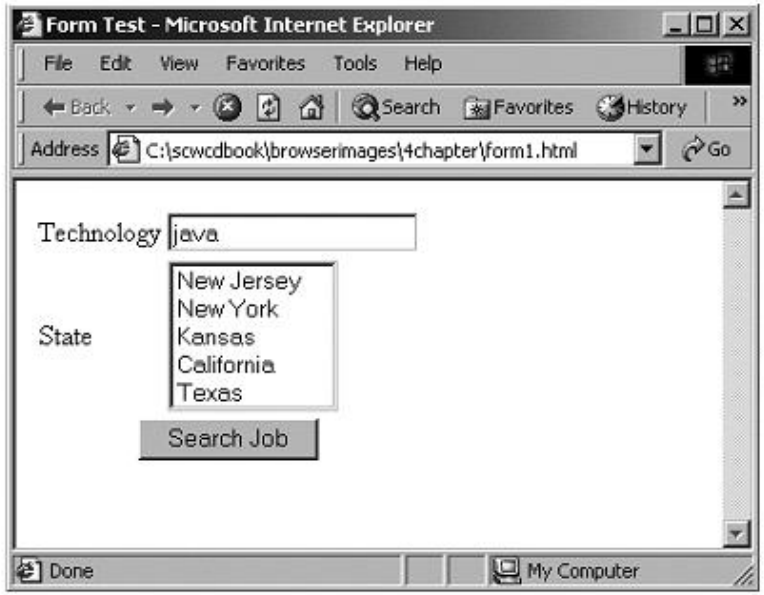
- **`getParameter()`**, **`getParameterValues()`**, **`getParameterNames()`** – доступ к именам и значениям переменных, включенных в запрос;
- **`getHeader()`**, **`getHeaders()`**, **`getHeaderNames()`** – доступ к HTTP-заголовкам запроса;
- **`getCookies()`** – доступ к cookies, отправленным в запросе;
- **`getSession()`** – доступ к сессии.

13. Сервлеты
Доступ к переменным запроса



Метод	Описание
`String getParameter (String parameterName)`	Метод возвращает только одно значение, связанное с указанной переменной.
`String[] getParameterValues (String parameterName)`	Метод возвращает все значения, связанные с указанной переменной.
`Enumeration getParameterNames()`	Метод возвращает множество имён полученных переменных.

13. Сервлеты
Пример обработки HTTP-запроса



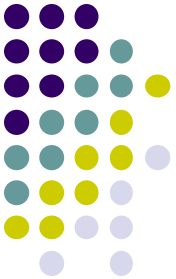

```
public void doPost(HttpServletRequest req,
                   HttpServletResponse res)
{
   String searchString = req.getParameter("searchstring");

   String[] stateList = req.getParameterValues("state");

   //use the values and generate appropriate response
}
```

```
<form action="/servlet/TestServlet" method="POST">
Technology : <input type="text" name="searchstring" value="java">
<br><br>
State : <select name="state" size="5" multiple>
   <option value="NJ">New Jersey</option>
   <option value="NY">New York</option>
   <option value="KS">Kansas</option>
   <option value="CA">California</option>
   <option value="TX">Texas</option>
</select>
<br><br>
<input type="submit" value="Search Job">
</form>
```

Uses HTTP POST

Allows selection of multiple values

13. Сервлеты
Доступ к заголовкам HTTP-запроса



Метод	Описание
`String getHeader (String headerName)`	Метод возвращает только одно значение, связанное с указанным заголовком.
`String[] getHeaderValues (String headerName)`	Метод возвращает все значения, связанные с указанным заголовком.
`Enumeration getHeaderNames()`	Метод возвращает множество имён полученных HTTP-заголовков.

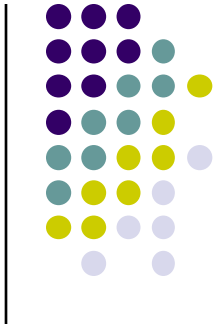
13. Сервлеты
Пример извлечения HTTP-заголовков

```
public void service(HttpServletRequest req,
                    HttpServletResponse res)
{
    Enumeration headers = req.getHeaderNames();     <-- Retrieves header names

    while (headers.hasMoreElements())
    {
        String header = (String) headers.nextElement();
        String value = req.getHeader(header);     <-- Retrieves header values
        System.out.println(header+" = "+value);
    }
}
```

13. Сервлеты
Доступ к данным сессии

Метод	Описание
`HttpSession getSession(boolean create)`	Возвращает экземпляр текущей сессии, ассоциированной с запросом. В случае, если сессия ещё не начата, а флаг **`create`** = **`true`**, создаёт новую сессию.
`HttpSession getSession()`	Вызов данного метода является эквивалентом `getSession(true);`
`void setAttribute(String name, Object value)`	Добавляет объект **`value`** в сессию под именем **`name`**.
`Object getAttribute(String name)`	Извлекает из сессии объект, ассоциированный с именем **`name`**, или **`null`**, если с таким именем не связано объектов.

13. Сервлеты
Реализация поддержки сессий



```
POST /servlet/testServlet HTTP/1.1
User-Agent= MOZILLA/1.0
cookie=jsessionid=61C4F23524521390E70993E5120263C6
Content-Type: application/x-www.formurlencoded

userid=john
```

Header line for the cookie

```
<html>
<head></head>
<body>
A test page showing two URLs:<br>
<a href=
 "/servlet/ReportServlet;jsessionid=C084B32241B2F8F060230440C0158114">
  View Report</a><br>
```

Метод	Описание
`String encodeURL(String url)`	Возвращает URL с включенным идентификатором сессии
`String encodeRedirectURL (String url)`	Возвращает URL с включенным идентификатором сессии для использования в методе `HttpServletResponse.sendRedirect()`.

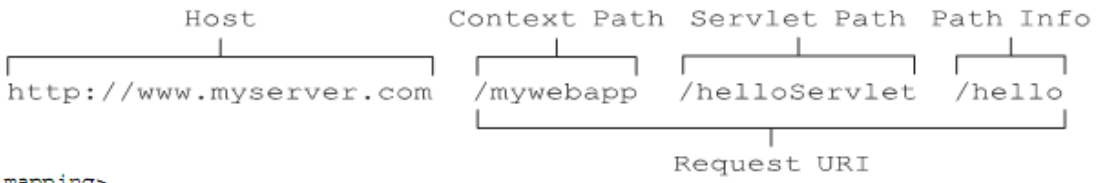
13. Сервлеты

Привязка сервлетов к URL

```
<!ELEMENT servlet-mapping (servlet-name, url-pattern)>
```

```
<servlet-mapping>
    <servlet-name>accountServlet</servlet-name>
    <url-pattern>/account/*</url-pattern>
</servlet-mapping>

<servlet-mapping>
    <servlet-name>accountServlet</servlet-name>
    <url-pattern>/myaccount/*</url-pattern>
</servlet-mapping>
<servlet-mapping>
    <servlet-name>pdfGeneratorServlet</servlet-name>
    <url-pattern>*.pdf</url-pattern>
</servlet-mapping>
<servlet-mapping>
    <servlet-name>reportServlet</servlet-name>
    <url-pattern>/report</url-pattern>
</servlet-mapping>
```

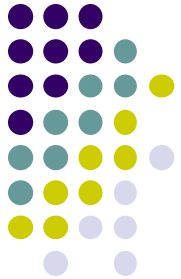
13. Сервлеты
Привязка сервлетов к URL




```
<servlet-mapping>
   <servlet-name>RedServlet</servlet-name>
   <url-pattern>/red/*</url-pattern>
</servlet-mapping>

<servlet-mapping>
   <servlet-name>RedServlet</servlet-name>
   <url-pattern>/red/red/*</url-pattern>
</servlet-mapping>

<servlet-mapping>
   <servlet-name>RedBlueServlet</servlet-name
   <url-pattern>/red/blue/*</url-pattern>
</servlet-mapping>

<servlet-mapping>
   <servlet-name>BlueServlet</servlet-name>
   <url-pattern>/blue/</url-pattern>
</servlet-mapping>

<servlet-mapping>
   <servlet-name>GreenServlet</servlet-name>
   <url-pattern>/green</url-pattern>
</servlet-mapping>

<servlet-mapping>
   <servlet-name>ColorServlet</servlet-name>
   <url-pattern>*.col</url-pattern>
</servlet-mapping>
```

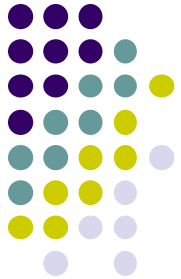
Request URI	Servlet used	Servlet path	Path info
/colorapp/red	RedServlet	/red	null
/colorapp/red/	RedServlet	/red	/
/colorapp/red/aaa	RedServlet	/red	/aaa
/colorapp/red/blue/aa	RedBlueServlet	/red/blue	/aa
/colorapp/red/red/aaa	RedServlet	/red/red	/aaa
/colorapp/hello/aa.col	ColorServlet	/hello/aa.col	null
/colorapp/blue	NONE (Error message)		
/colorapp/hello/blue/	NONE (Error message)		
/colorapp/blue/mydir	NONE (Error message)		

Ответ пользователю – интерфейс HttpServletResponse



Интерфейс **`HttpServletResponse`** является средством отправки пользователю ответа, сконструированного на основе запроса. Среди наиболее важных методов ответа являются:

- **`setHeader(), addHeader()`** – установка HTTP-заголовков ответа;
- **`containsHeader()`** – проверка, был ли уже установлен такой HTTP-заголовок;
- **`getWriter()`** – получить экземпляр текстового выходного потока для записи ответа;
- **`getOutputStream()`** – получить экземпляр двоичного выходного потока для записи ответа.



Вариант 1 (текстовые данные): посредством PrintWriter

```
PrintWriter pw = res.getWriter();
```
⟵ Gets the PrintWriter object

```
pw.println("<html>");
pw.println("<head>");
pw.println("</head>");
pw.println("<body>");
```
Uses PrintWriter to write the HTML page

Вариант 2 (двоичные данные): посредством ServletOutputStream

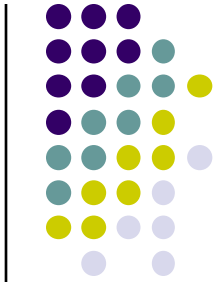
```
res.setContentType("application/jar");
```
⟵ Sets the content type

```
File f = new File("test.jar");
byte[] bytearray = new byte[(int) f.length()];
FileInputStream is = new FileInputStream(f);
is.read(bytearray);
```
Reads the file into a byte array

```
OutputStream os = res.getOutputStream();
```
⟵ Gets the OutputStream

```
os.write(bytearray);
```
⟵ Sends the bytes of the byte array to the browser

```
os.flush();
```
⟵ Flushes the data



На сегодня всё, до свидания и до следующей встречи!

На следующей лекции мы поговорим о создании веб-страниц на основе технологии JSP