

JAVA FOR STUDENTS

ФИЛИАЛ КАФЕДРЫ ПОИТ БГУИР В ЕРАМ
SYSTEMS

КУРС: ВЕБ-ТЕХНОЛОГИИ (JAVA)
SERVLETS FUNDAMENTALS

Olga Smolyakova , PhD

Oracle Certified Java 6 Programmer

Olga_Smolyakova@epam.com

Содержание

1. **forward(), include(), sendRedirect()**
2. **ServletContext, ServletConfig**
3. **Sessions**
4. **Cookies**
5. **Listeners**
6. **Filters**

**FORWARD(), INCLUDE(),
SENDREDIRECT()**

RequestDispatcher

void **forward**(ServletRequest request, ServletResponse response)

Направляет запрос из сервлета на другой ресурс (сервлет, файл JSP или HTML файл) на сервере.

void **include**(ServletRequest request, ServletResponse response)

Включает содержимое ресурса (сервлета, страницы JSP, HTML-файл) в ответ.

HttpServletResponse

void **sendRedirect**(String location)

Посылает временный ответ перенаправления к клиенту с помощью указывающего местоположение перенаправления URL и очищает буфер.

forward() vs sendRedirect()

forward()	sendRedirect()
forward() выполняется на стороне сервера	sendRedirect() выполняется на стороне клиента
Запрос передается на другой ресурс в пределах одного сервера	Запрос передается на другой ресурс, возможно на другой сервер
Не зависит от протокола запроса клиента, так как выполнение forward() обеспечивается контейнером сервлетов	sendRedirect() зависит от протокола http и может быть использован только с http-клиентом
Запрос доступен в конечном ресурсе	Новый запрос создается для конечного ресурса
Осуществляется только один вызов метода.	Два вызова (запрос-ответ) происходят.
Может быть использован только в пределах одного сервера	Может быть использован в пределах одного сервера и во вне сервера
Адреса ресурса, на который был сделан forward(), не видно	Адреса ресурса, на который был сделан sendRedirect(), виден
forward() быстрее чем sendRedirect()	sendRedirect() медленнее; когда создается новый запрос – старый теряется

index.jsp

```
<%@ page language="java" contentType="text/html;
charset=utf-8"
    pageEncoding="utf-8"%>
...
<body>
    <form action="ControllerForward" method="post">
        <input type="submit" value="forward()" /><br/>
    </form>

    <br/>

    <form action="ControllerSendRedirect" method="post">
        <input type="submit" value="sendRedirect()" /><br/>
    </form>
</body>
</html>
```



```
<%@ page language="java" contentType="text/html;  
charset=utf-8"  
    pageEncoding="utf-8"%>  
  
...  
<body>  
<h1>  
I am the main JSP.  
</h1>  
</body>  
</html>
```



ControllerForward.java

```
package _java._ee._01._forward;

import ...;

public class ControllerForward extends HttpServlet {
    private static final long serialVersionUID = 1L;

    ...

    protected void doPost(HttpServletRequest request,
        HttpServletResponse response) throws ServletException,
        IOException {
        request.getRequestDispatcher("jsp/main.jsp").
            forward(request, response);
    }
}
```



ControllerSendRedirect.java


```
package _java._ee._01._sendredirect;

import ...;

public class ControllerSendRedirect extends HttpServlet {
    private static final long serialVersionUID = 1L;

    ...

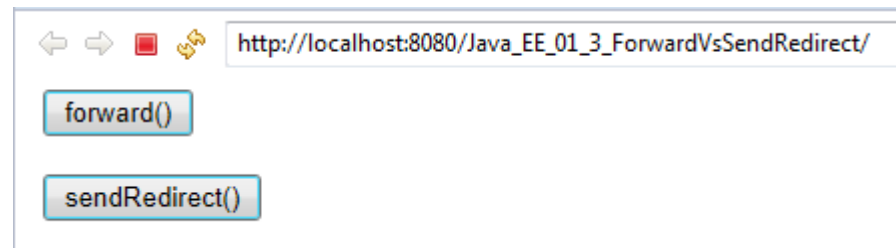
    protected void doPost(HttpServletRequest request,
        HttpServletResponse response) throws ServletException,
        IOException {
        response.sendRedirect("jsp/main.jsp");
    }
}
```



```
<?xml version="1.0" encoding="UTF-8"?>
<web-app ...>
  <display-name>Java_EE_01_3_ForwardVsSendRedirect</display-name>
  <servlet>
    <servlet-name>ControllerForward</servlet-name>
    <servlet-class>_java._ee._01._forward.ControllerForward</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>ControllerForward</servlet-name>
    <url-pattern>/ControllerForward</url-pattern>
  </servlet-mapping>
  <servlet>
    <servlet-name>ControllerSendRedirect</servlet-name>
    <servlet-class>_java._ee._01._sendredirect.ControllerSendRedirect
  </servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>ControllerSendRedirect</servlet-name>
    <url-pattern>/ControllerSendRedirect</url-pattern>
  </servlet-mapping>
</web-app>
```



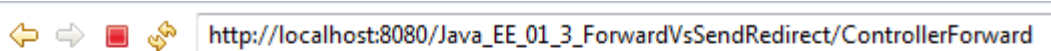
Запрос:



Ответ:



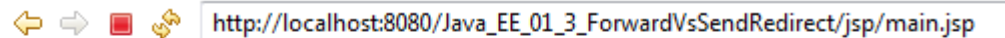
http://localhost:8080/Java_EE_01_3_ForwardVsSendRedirect/ControllerForward



I am the main JSP.



http://localhost:8080/Java_EE_01_3_ForwardVsSendRedirect/jsp/main.jsp



I am the main JSP.

SERVLETCONTEXT, SERVLETCONFIG

Интерфейс ServletContext

Интерфейс **ServletContext** используется для взаимодействия с контейнером сервлетов.

Все сервлеты в контексте приложения совместно используют атрибуты, доступ к которым можно получить с помощью объекта типа **ServletContext**.

Чтобы избежать столкновений имен атрибутов, для их имен используют те же правила, что и для именования пакетов.

Когда в приложении несколько сервлетов используют атрибут, то каждый должен проинициализировать этот атрибут: каждый сервлет должен проверить значение атрибута, и устанавливать его только в том случае если предыдущий сервлет не сделал этого.

Некоторые методы **ServletContext**:

Метод	Описание
void setAttribute(String name, Object object)	добавляет атрибут и его значение в контекст
Object getAttribute(String name)	возвращает совместный ресурс
Enumeration getAttributeNames()	получает список имен атрибутов
void removeAttribute(String name)	удаляет совместный ресурс
ServletContext getContext(String uripath)	позволяет получить доступ к контексту других ресурсов данного контейнера сервлетов
String getServletContextName()	возвращает имя сервлета, которому принадлежит данный объект интерфейса ServletContext
String getCharacterEncoding()	определение символьной кодировки запроса

Интерфейс ServletConfig

ServletConfig - конфигурация сервлета, используется (в основном) на этапе инициализации. Все параметры для инициализации устанавливаются в web.xml

Некоторые методы **ServletConfig**:

Метод	Описание
String getServletName()	определение имени сервлета
Enumeration getInitParameterNames()	определение имен параметров инициализации сервлета из дескрипторного файла web.xml
String getInitParameter(String name)	определение значения конкретного параметра по его имени

Пример 1

Controller.java

```
package _java._ee._01._servlet;  
import ...;  
public class Controller extends HttpServlet {  
    ...  
    protected void doPost(...) ... {  
        ServletContext servletContext = getServletContext();  
        ServletConfig servletConfig = getServletConfig();  
        String name;    String value;  
        PrintWriter out = response.getWriter();  
  
        Enumeration<?> initName =  
            servletConfig.getInitParameterNames();  
        out.println("init-params:<br/>" );  
        while (initName.hasMoreElements()) {  
            name = initName.nextElement().toString();  
            value = servletConfig.getInitParameter(name);  
            out.println(name + " - " + value + "<br/>");  
        }  
    }  
}
```




```
Enumeration<?> initNameContext =  
    servletContext.getInitParameterNames();  
out.println("<br/>context-params:<br/>" );  
while (initNameContext.hasMoreElements()) {  
    name = initNameContext.nextElement().toString();  
    value = servletContext.getInitParameter(name);  
    out.println(name + " - " + value + "<br/>");  
}  
}  
}
```

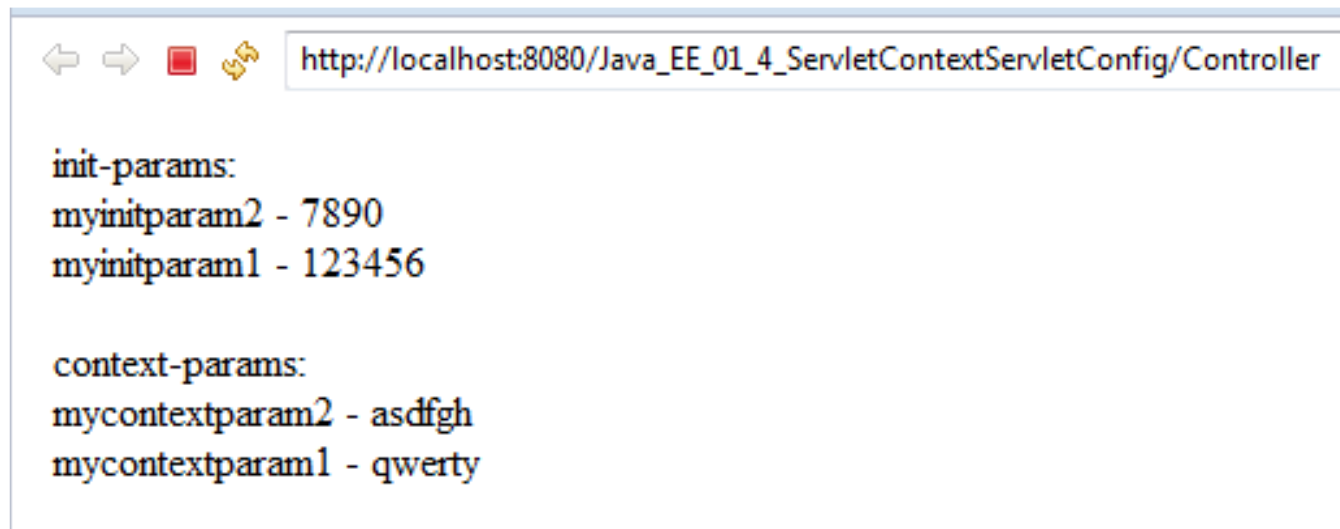


web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app ...>
    ...
    <servlet>
        <servlet-name>Controller</servlet-name>
        <servlet-
class>_java._ee._01._servlet.Controller</servlet-class>
        <init-param>
            <param-name>myinitparam1</param-name>
            <param-value>123456</param-value>
        </init-param>
        <init-param>
            <param-name>myinitparam2</param-name>
            <param-value>7890</param-value>
        </init-param>
    </servlet>
    <servlet-mapping>
        <servlet-name>Controller</servlet-name>
        <url-pattern>/Controller</url-pattern>
    </servlet-mapping>
</web-app>
```



Результат:



Пример 2

Servlet1.java

```
package _java._ee._01._servlet;
import ...;
public class Servlet1 extends HttpServlet {
    private ServletConfig config;

    public void init(ServletConfig config) throws ServletException {
        this.config = config;
    }

    protected void doGet(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
        response.setContentType("text/html");
        String param = (String) config.getServletContext().getAttribute("myparam");
        if (param == null) {
            config.getServletContext().setAttribute("myparam", "servlet1");
            response.getWriter().println("myparam = servlet1 set first<br/>");
        }
        response.getWriter().println("From Servlet1 - "
            + config.getServletContext().getAttribute("myparam"));
    }
}
```




Servlet2.java

```
package _java._ee._01._servlet;
import ...;
public class Servlet2 extends HttpServlet {
    private ServletConfig config;

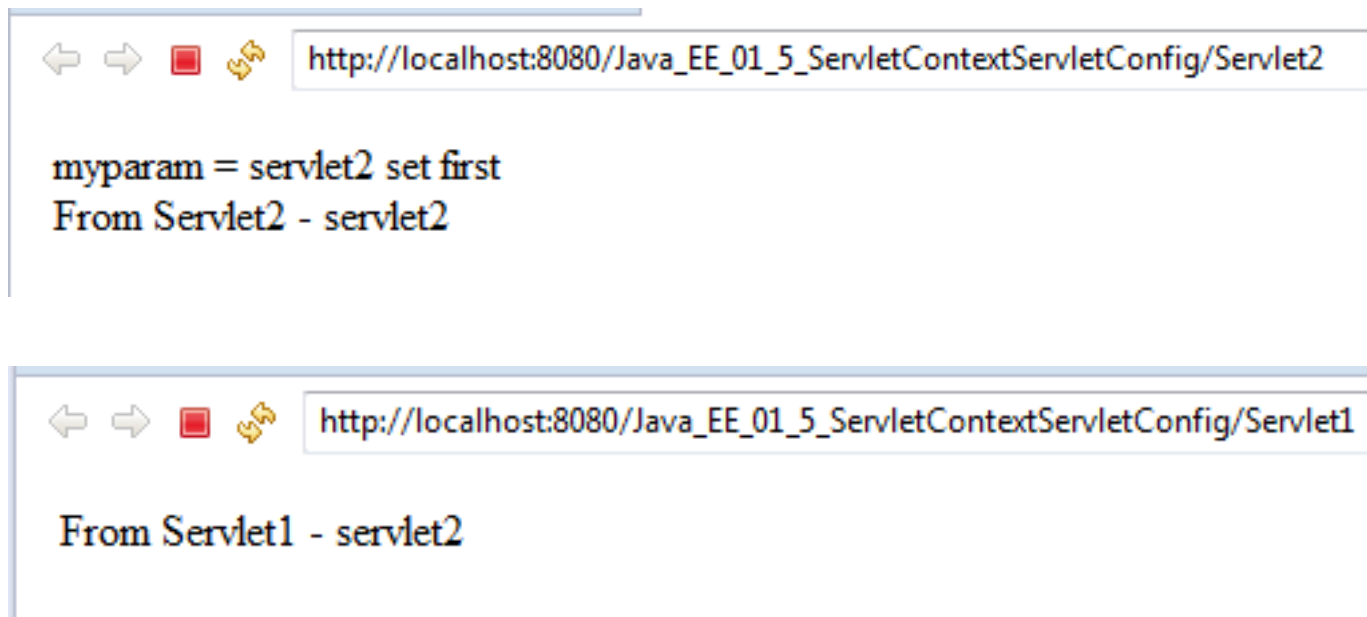
    public void init(ServletConfig config) throws ServletException {
        this.config = config;
    }

    protected void doGet(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
        response.setContentType("text/html");
        String param = (String) config.getServletContext().getAttribute("myparam");

        if (param == null) {
            config.getServletContext().setAttribute("myparam", "servlet2");
            response.getWriter().println("myparam = servlet2 set first<br/>");
        }
        response.getWriter().println("From Servlet2 - "
            + config.getServletContext().getAttribute("myparam"));
    }
}
```



Результат:



SESSIONS

Для поддержки статуса между сериями запросов от одного и того же пользователя используется механизм отслеживания сессии.

Сессии используются разными сервлетами для доступа к одному клиенту.



Чтобы использовать отслеживание сессии:

- Создайте для пользователя сессию (объект HttpSession).
- Сохраняйте или читайте данные из объекта HttpSession.
- Уничтожьте сессию (необязательно).

Создание или получение сессии

HttpSession getSession(boolean create) объекта

HttpServletRequest возвращает сессию пользователя.

Когда метод вызывается со значением *create* равным **true**, реализация при необходимости создает сессию или возвращают уже имеющуюся, ассоциированную с этим request-ом. Вызов метода со значение *create* равным **false** возвратит **null**, если сессия не обнаружена .

```
...  
    public void doGet (HttpServletRequest request,  
        HttpServletResponse response)  
        throws ServletException, IOException {  
        HttpSession session = request.getSession(true);  
        out = response.getWriter();  
    }  
...
```

Работа с сессией

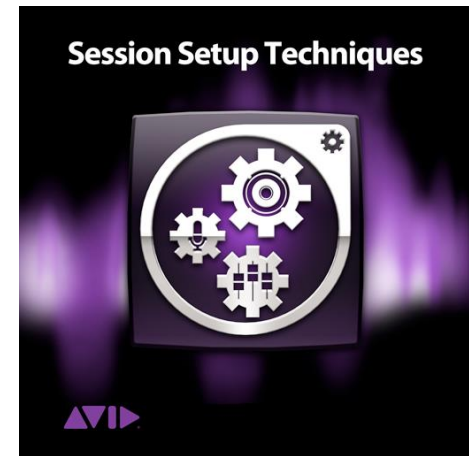
Методы интерфейса HttpSession.

```
interface HttpSession {  
    Object getAttribute(String name);  
    Enumeration getAttributeNames();  
    long getCreationTime();  
    String getId();  
    long getLastAccessedTime();  
    int getMaxInactiveInterval();  
    ServletContext getServletContext();  
    void invalidate();  
    boolean isNew();  
    void removeAttribute(String name);  
    void setAttribute(String name, Object value);  
    void setMaxInactiveInterval(int interval);  
}
```

Завершение сессии, время жизни сессии

Сессия пользователя может быть завершена вручную или, в зависимости от того, где запущен сервлет, автоматически. (время неактивной жизни сессии по умолчанию – 30 минут.)

Завершить сессию означает удалить объект HttpSession и его величины из системы.



Время жизни сессии (в минутах) также можно определить в дескрипторе развертывания web.xml.

```
<session-config>  
    <session-timeout>10</session-timeout>  
</session-config>
```

Программно можно установить интервал времени ожидания сессии (в секундах) с помощью метода

```
public void setMaxInactiveInterval(int seconds)
```

интерфейса **HttpSession**.

При отрицательном значении параметра время ожидания сессии никогда не истечет.

index.jsp

```
<%@ page language="java" contentType="text/html; charset=utf-8"
    pageEncoding="utf-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Session</title>
</head>
<body>
<br/>
    <form action="Controller" method="post">
        <input type="text" name="paramname" value="" /> <br/>
        <input type="text" name="paramvalue" value="" />
<br/>
        <input type="submit" value="send next HttpRequest"
/><br />
    </form>
</body>
</html>
```



Controller.java

```
package _java._ee._01._servlet;

import ...;

public class Controller extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse
        response) throws ServletException, IOException {
        doPost(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse
        response) throws ServletException, IOException {
        response.setContentType("text/html");
        HttpSession session = request.getSession();

        String name;
        String value;

        name = request.getParameter("paramname");
        value = request.getParameter("paramvalue");
        session.setAttribute(name, value);
    }
}
```



```
PrintWriter out = response.getWriter();
Enumeration<String> sessionParams =
    session.getAttributeNames();

out.println(session.getId() + "<br/>");
while(sessionParams.hasMoreElements()){
    name = sessionParams.nextElement();
    value = (String) session.getAttribute(name);

    out.println(name + " - " + value + "<br/>");
}

request.getRequestDispatcher("/index.jsp").
    include(request, response);
}
```



Результат:

← → ■ ⚙ http://localhost:8080/Java_EE_01_6_Session/

a

11

send next HttpRequest

← → ■ ⚙ http://localhost:8080/Java_EE_01_6_Session/Controller

E1B6C0B52C1EA842BC999D1CB6C27F85

a - 11

send next HttpRequest

← → ■ ⚙ http://localhost:8080/Java_EE_01_6_Session/Controller

E1B6C0B52C1EA842BC999D1CB6C27F85

b - 22

a - 11

c

33

send next HttpRequest

Перезапись URL

Сессии становятся доступными при помощи обмена уникальными метками, которые называются идентификаторами сессии (session id), между клиентом, осуществляющим запрос, и сервером.

Если в браузере клиента разрешены cookies, идентификатор сессии будет внесен в файл cookie, отправляемый с HTTP-запросом/ответом.

```

package _java._ee._01._servlet;
import ...;

public class Controller extends HttpServlet {
...
    protected void doPost(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
        response.setContentType("text/html");
        HttpSession session = request.getSession();

        PrintWriter out = response.getWriter();
        Cookie[] cookie = request.getCookies();
        for (Cookie cook : cookie){
            out.println(cook.getName() + " - " + cook.getValue());
        }
    }
}

```

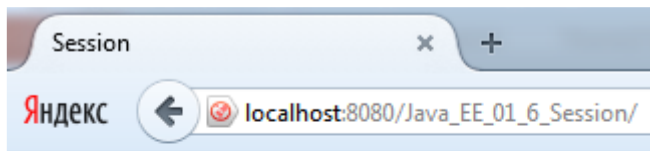
Результат:



http://localhost:8080/Java_EE_01_6_Session/Controller

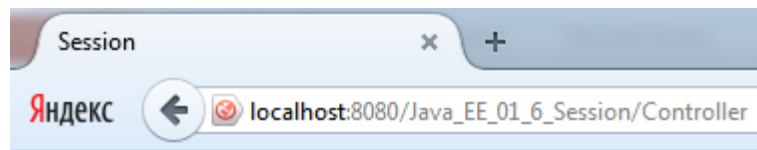
JSESSIONID - 9BBD271DA83D6605F1671AD2EE3986CC

Если в браузере запретить cookies – получится следующий результат:



a
11

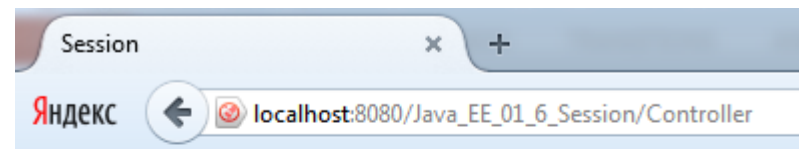
send next HttpRequest



E3995583E89732C1819476853101F989
a - 11

b
22

send next HttpRequest



CB73418E2AB6A42E404EDB491154C14C
b - 22

send next HttpRequest

Для браузеров, не поддерживающих cookies, для того, чтобы сделать возможной обработку сессий, используется способ перезаписи URL.

Если используется перезапись URL, идентификатор сессии должен быть прибавлен к URL, включая гиперссылки, которым необходим доступ к сессии, а также ответы сервера.

Методы **HttpServletResponse**, которые поддерживают перезапись URL:

- `public String encodeURL(String url)`
- `public String encodeRedirectURL(String url)`

Метод **encodeURL()** кодирует заданный URL, добавляя в него идентификатор сессии, или, если кодирование не требуется, возвращает исходный URL.

Метод **encodeRedirectURL()** кодирует заданный URL для использования в методе **sendRedirect()** **HttpServletResponse**. Этот метод тоже возвращает неизмененный URL, если кодирование не требуется.

index.jsp

```
<%@ page language="java" contentType="text/html; charset=utf-8"
    pageEncoding="utf-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
...
<body>
<br />
<c:if test="${requestScope.encodeURL == null}">
    <c:set var="encodeURL" scope="request" value="Controller" />
</c:if>

<form action="${requestScope.encodeURL}" method="post">
    <input type="submit" value="send next HttpRequest" /><br />
</form>

<c:out value="${requestScope.encodeURL}" />
</body>
</html>
```



Controller.java

```
package _java._ee._01._servlet;
import ...;
public class Controller extends HttpServlet {
    protected void doPost(HttpServletRequest request,
        HttpServletResponse response) throws ServletException,
        IOException {

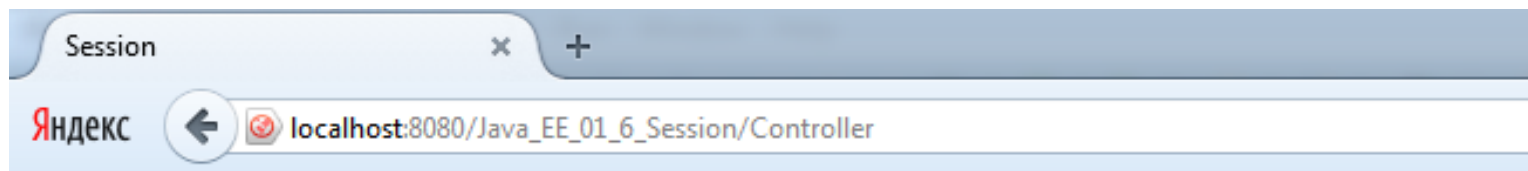
        HttpSession session = request.getSession();
        String encodeURL = response.encodeURL(request.getContextPath())
                                + "/Controller";

        request.setAttribute("encodeURL", encodeURL);

        request.getRequestDispatcher("/index.jsp").forward(request, response);
    }
}
```



Результат (браузер с отключенными cookies):



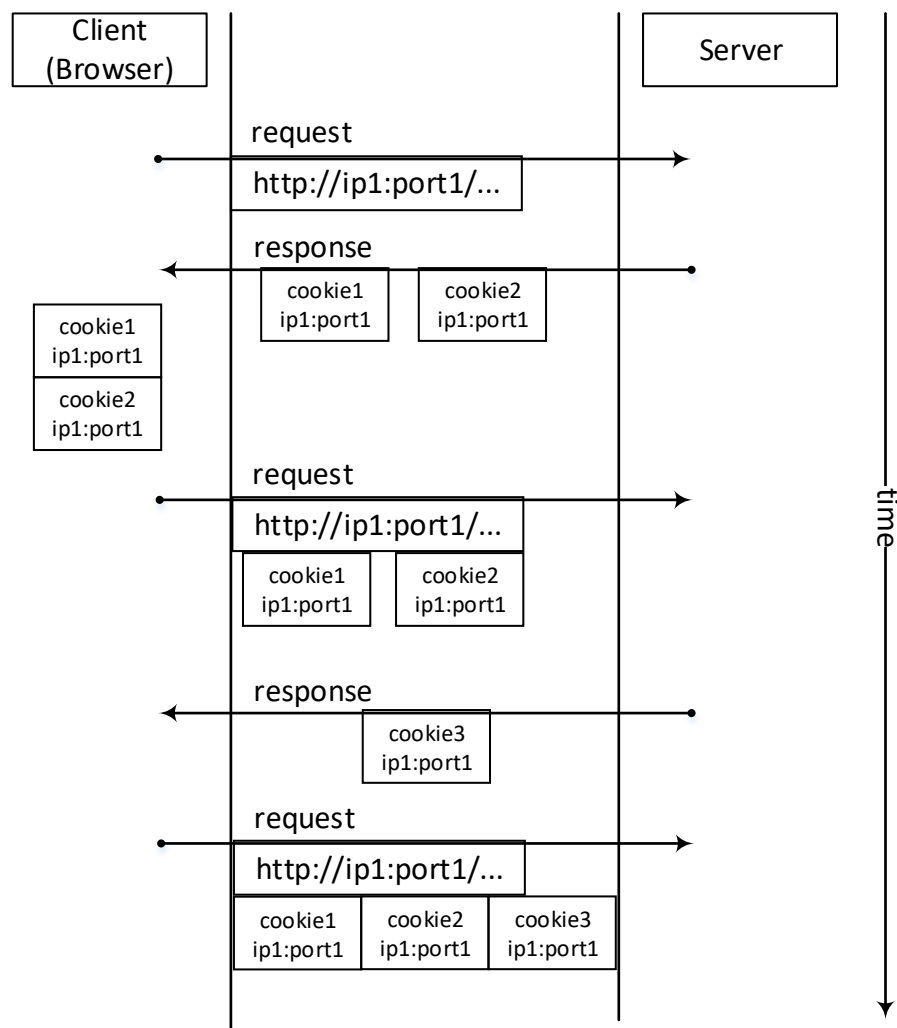
send next HttpRequest

/Java_EE_01_6_Session;jsessionid=20F08ACEC375996AED34EBA9EA78630B/Controller

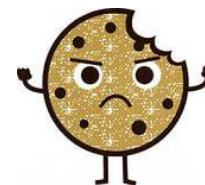
COOKIES

Использование Cookie

- Cookies - используются для хранения части информации на машине клиента.
- Закладки передаются клиенту в коде ответа в HTTP.
- Клиенты автоматически возвращают закладки, добавляя код в запросы в HTTP заголовках.



- Сервер может обеспечить одну или более закладок для клиента. Предполагается, что программа клиента, как web браузер, должна поддерживать 20 закладок на хост, как минимум четыре килобайта каждая.



- Закладки, которые клиент сохранил для сервера, возвращаются клиентом только этому серверу. Сервер может включать множество сервлетов. Потому как закладки возвращаются серверу, сервлеты работающие на этом сервере совместно используют эти закладки.



Создание Cookie и отправка его клиенту

Конструктор класса `javax.servlet.http.Cookie` создает закладку с начальным именем и значением. Можно изменить значение закладки позже, вызвав метод `setValue`.

Установить время жизни закладки можно с помощью метода `setMaxAge()`.

```
String bookId = request.getParameter("Buy");  
if (bookId != null) {  
    Cookie getBook = new Cookie("Buy", bookId);  
}
```

Отправка закладки клиенту

Закладки отправляются как заголовки ответа клиенту; они добавляются с помощью метода **addCookie()** класса **HttpServletResponse**.

Если используется **Writer** для отправки закладки пользователю, необходимо использовать метод **addCookie()** прежде, чем вызвать метод **getWriter()** класса **HttpServletResponse**.

```
if (values != null) {  
    bookId = values[0];  
    Cookie getBook = new Cookie("Buy", bookId);  
    getBook.setComment("User has indicated a desire " +  
        "to buy this book from the bookstore.");  
    response.addCookie(getBook);  
}  
...
```

Чтение cookies

Клиенты возвращают закладки как поля, добавленные в HTTP заголовок запроса.

Cookie[] getCookies() из класса **HttpServletRequest** – возвращает все закладки ассоциированные к данному хосту.

Получить значение закладки можно с помощью метода **getValue()**.

```
String bookId = request.getParameter("Remove");  
if (bookId != null) {  
    Cookie[] cookies = request.getCookies();  
}  
...
```


index.jsp

```
<%@ page language="java" contentType="text/html; charset=utf-8"
    pageEncoding="utf-8"%>
...
<body>
    <form action="Controller" method="post">
        <input type="text" name="cookiekey" value="" /><br/>
        <input type="text" name="cookievalue" value="" /><br/>
        <input type="submit" value="send next request" /><br/>
    </form>
</body>
</html>
```



Controller.java

```
public class Controller extends HttpServlet {  
    ...  
    protected void doPost(HttpServletRequest request,  
        HttpServletResponse response) throws ServletException,  
        IOException {  
        response.setContentType("text/html");  
  
        String cookieKey;  
        String cookieValue;  
  
        cookieKey = request.getParameter("cookiekey");  
        cookieValue = request.getParameter("cookievalue");  
  
        if ((cookieKey != null) && (cookieValue != null)) {  
            Cookie cookie = new Cookie(cookieKey, cookieValue);  
            response.addCookie(cookie);  
        }  
    }  
}
```




```
Cookie[] reqCookie = request.getCookies();  
if (reqCookie != null) {  
    for (Cookie c : reqCookie) {  
        response.getWriter()  
            .println(c.getName() + " - "  
                    + c.getValue() + "<br/>");  
    }  
} else {  
    response.getWriter().println("No cookies");  
}  
request.getRequestDispatcher("/index.jsp")  
    .include(request, response);  
}
```



Результат:

← → ■ ⚙ http://localhost:8080/Java_EE_01_7_Cookie/

← → ■ ⚙ http://localhost:8080/Java_EE_01_7_Cookie/Controller

JSESSIONID - 59B74C79FF38816CFBDBDC8696A9CB04
key1 - param1

← → ■ ⚙ http://localhost:8080/Java_EE_01_7_Cookie/Controller

JSESSIONID - 59B74C79FF38816CFBDBDC8696A9CB04
key1 - param1

← → ■ ⚙ http://localhost:8080/Java_EI

JSESSIONID - 59B74C79FF38816CF
key1 - param1
key2 - param2

LISTENERS

Существует несколько интерфейсов, которые позволяют следить за событиями, связанными с сеансом, контекстом и запросом сервлета, генерируемыми во время жизненного цикла Web-приложения.

Интерфейс	Описание
<code>javax.servlet.ServletContextListener</code>	обрабатывает события создания/удаления контекста сервлета
<code>javax.servlet.http.HttpSessionListener</code>	обрабатывает события создания/удаления HTTP-сессии
<code>javax.servlet.ServletContextAttributeListener</code>	обрабатывает события создания/удаления/модификации атрибутов контекста сервлета
<code>javax.servlet.http.HttpSessionAttributeListener</code>	обрабатывает события создания/удаления/модификации атрибутов HTTP-сессии
<code>javax.servlet.http.HttpSessionBindingListener</code>	обрабатывает события привязывания/разъединения объекта с атрибутом HTTP-сессии
<code>javax.servlet.http.HttpSessionActivationListener</code>	обрабатывает события связанные с активацией/деактивацией HTTP-сессии
<code>javax.servlet.ServletRequestListener</code>	обрабатывает события создания/удаления запроса
<code>javax.servlet.ServletRequestAttributeListener</code>	обрабатывает события создания/удаления/модификации атрибутов запроса сервлета

Методы интерфейсов-слушателей сервлета.

Интерфейсы и их методы

ServletContextListener

contextInitialized(ServletContextEvent e)

contextDestroyed(ServletContextEvent e)

HttpSessionListener

sessionCreated(HttpSessionEvent e)

sessionDestroyed(HttpSessionEvent e)

ServletContextAttributeListener

attributeAdded(ServletContextAttributeEvent e)

attributeRemoved(ServletContextAttributeEvent e)

attributeReplaced(ServletContextAttributeEvent e)

HttpSessionAttributeListener

attributeAdded(HttpSessionBindingEvent e)

attributeRemoved(HttpSessionBindingEvent e)

attributeReplaced(HttpSessionBindingEvent e)

Методы интерфейсов-слушателей сервлета.

Интерфейсы и их методы

HttpSessionBindingListener

valueBound(HttpSessionBindingEvent e)

valueUnbound(HttpSessionBindingEvent e)

HttpSessionActivationListener

sessionWillPassivate(HttpSessionEvent e)

sessionDidActivate(HttpSessionEvent e)

ServletRequestListener

requestDestroyed(ServletRequestEvent e)

requestInitialized(ServletRequestEvent e)

ServletRequestAttributeListener

attributeAdded(ServletRequestAttributeEvent e)

attributeRemoved(ServletRequestAttributeEvent e)

attributeReplaced(ServletRequestAttributeEvent e)

ProjectRequestListener.java

```
package _java._ee._01._listener;
import ...;
public class ProjectRequestListener
    implements ServletRequestListener {
    public void requestDestroyed(ServletRequestEvent arg0) {
        HttpServletRequest request
            = (HttpServletRequest) arg0.getServletRequest();
        System.out.println("Request from "
            + request.getContextPath() + " was destroyed.");
    }

    public void requestInitialized(ServletRequestEvent arg0) {
        HttpServletRequest request
            = (HttpServletRequest) arg0.getServletRequest();
        System.out.println("Request from " +
            request.getContextPath() + " was created.");
    }
}
```



Controller.java

```
package _java._ee._01._servlet;
import ...;
public class Controller extends HttpServlet {
    ...
    protected void doPost(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
        request.getRequestDispatcher("/index.jsp")
            .forward(request, response);
    }
}
```

index.jsp

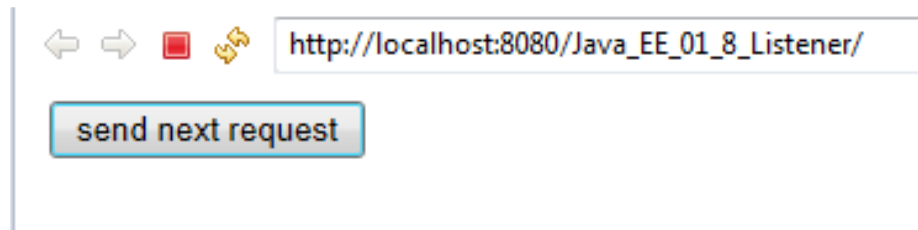
```
<%@ page language="java" contentType="text/html; charset=utf-8"
    pageEncoding="utf-8"%>
...
<body>
    <form action="Controller" method="post">
        <input type="submit" value="send next request" /><br />
    </form>
</body>
</html>
```



web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app ...>
<servlet>
<servlet-name>Controller</servlet-name>
  <servlet-class>_java._ee._01._servlet.Controller</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>Controller</servlet-name>
  <url-pattern>/Controller</url-pattern>
</servlet-mapping>
<listener>
  <listener-class>
    _java._ee._01._listener.ProjectRequestListener
</listener-class>
</listener>
</web-app>
```

Результат:



```
Aug 01, 2014 1:51:38 PM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["ajp-bio-8009"]
Aug 01, 2014 1:51:38 PM org.apache.catalina.startup.Catalina start
INFO: Server startup in 3277 ms
Request from /Java_EE_01_8_Listener was created.
Request from /Java_EE_01_8_Listener was destroyed.
Request from /Java_EE_01_8_Listener was created.
Request from /Java_EE_01_8_Listener was destroyed.
```

FILTERS

Реализация интерфейса **Filter** позволяет создать объект, который может трансформировать заголовок и содержимое запроса клиента или ответа сервера.

Фильтры не создают запрос или ответ, а только модифицируют его. Фильтр выполняет предварительную обработку запроса, прежде чем тот попадает в сервлет, с последующей (если необходимо) обработкой ответа, исходящего из сервлета.



Основные действия, которые может выполнить фильтр:

- перехват инициализации сервлета и определение содержания запроса, прежде чем сервлет будет инициализирован;
- блокировка дальнейшего прохождения пары request-response;
- изменение заголовка и данных запроса и ответа;
- взаимодействие с внешними ресурсами;
- построение цепочек фильтров;
- фильтрация более одного сервлета.

Интерфейсы, для работы с фильтрами:

- **javax.servlet.Filter** – интерфейс для реализации фильтра
- **javax.servlet.FilterChain** – список фильтров
- **javax.servlet.FilterConfig** – доступ к параметрам инициализации и контексту сервлета

Методы интерфейса **Filter**:

```
interface Filter {  
    void destroy();  
    void doFilter(ServletRequest request, ServletResponse response,  
                  FilterChain chain);  
    void init(FilterConfig filterConfig);  
}
```

CharsetFilter.java

```
package _java._ee._01._filter;
import ...;
public class CharsetFilter implements Filter {

    private String encoding;
    private ServletContext context;


    public void destroy() {}

    public void doFilter(ServletRequest request, ServletResponse response,
        FilterChain chain) throws IOException, ServletException {
        request.setCharacterEncoding(encoding);
        response.setCharacterEncoding(encoding);

        context.log("Charset was set.");

        chain.doFilter(request, response);
    }

    public void init(FilterConfig fConfig) throws ServletException {
        encoding = fConfig.getInitParameter("characterEncoding");
        context = fConfig.getServletContext();
    }
}
```



ServletLoggingFilter.java

```
package _java._ee._01._filter;
import...;
public class RequestLoggingFilter implements Filter {

    private ServletContext context;

    public void init(FilterConfig fConfig) throws ServletException {
        this.context = fConfig.getServletContext();
        this.context.log("RequestLoggingFilter initialized");
    }

    public void doFilter(ServletRequest request, ServletResponse response,
                        FilterChain chain) throws IOException, ServletException {

        HttpServletRequest req = (HttpServletRequest) request;
        Enumeration<String> params = req.getParameterNames();
        while (params.hasMoreElements()) {
            String name = params.nextElement();
            String value = request.getParameter(name);
            this.context.log(req.getRemoteAddr() + "::Request Params::{ "
                            + name + "=" + value + " }");
        }
    }
}
```




```
Cookie[] cookies = req.getCookies();
if(cookies != null){
    for(Cookie cookie : cookies){
        this.context.log(req.getRemoteAddr() +
            "::Cookie::{ "+cookie.getName()+" , "+cookie.getValue()+" }");
    }
}

chain.doFilter(request, response);
}

public void destroy() {
}

}
```



Controller.java

```
package _java._ee._01._servlet;
import ...;
public class Controller extends HttpServlet {
    ...

    protected void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        doPost(request, response);
    }

    protected void doPost(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        request.getRequestDispatcher("/index.jsp")
            .forward(request, response);
    }
}
```



Index.jsp

```
<%@ page language="java" contentType="text/html; charset=utf-8"
    pageEncoding="utf-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
...
<body>
    <form action="Controller" method="post">
        Name: <input type="text" name="name" value="" /> <br />
        Surname: <input type="text" name="surname" value="" /><br/>
        <input type="submit" value="send next request" /><br />
    </form>
</body>
</html>
```



web.xml

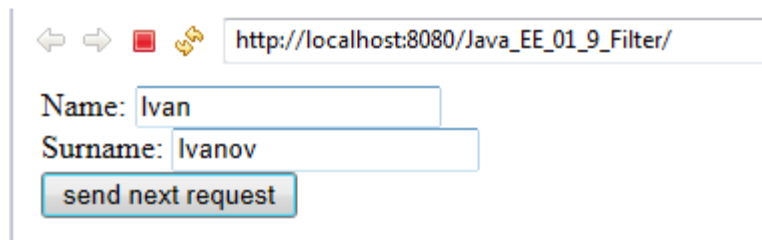
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app ...>
  ...
  <filter>
    <display-name>RequestLoggingFilter</display-name>
    <filter-name>RequestLoggingFilter</filter-name>
    <filter-class>_java._ee._01._filter.RequestLoggingFilter</filter-class>
  </filter>
  <filter>
    <display-name>CharsetFilter</display-name>
    <filter-name>CharsetFilter</filter-name>
    <filter-class>_java._ee._01._filter.CharsetFilter</filter-class>
    <init-param>
      <param-name>characterEncoding</param-name>
      <param-value>utf-8</param-value>
    </init-param>
  </filter>
</web-app>
```



```
<filter-mapping>
  <filter-name>CharsetFilter</filter-name>
  <url-pattern>/Controller</url-pattern>
</filter-mapping>
<filter-mapping>
  <filter-name>RequestLoggingFilter</filter-name>
  <url-pattern>/Controller</url-pattern>
</filter-mapping>
</web-app>
```



Результат:



← → 🚫 💰 http://localhost:8080/Java_EE_01_9_Filter/

Name:

Surname:

```
Aug 04, 2014 10:49:27 AM org.apache.catalina.core.ApplicationContext log
INFO: Charset was set.
Aug 04, 2014 10:49:27 AM org.apache.catalina.core.ApplicationContext log
INFO: 0:0:0:0:0:0:0:1::Request Params::{name=Ivan}
Aug 04, 2014 10:49:27 AM org.apache.catalina.core.ApplicationContext log
INFO: 0:0:0:0:0:0:0:1::Request Params::{surname=Ivanov}
Aug 04, 2014 10:49:27 AM org.apache.catalina.core.ApplicationContext log
INFO: 0:0:0:0:0:0:0:1::Cookie::{JSESSIONID,C9282AA3A42C30D86F4E2CD351DF32E8}
```



СПАСИБО ЗА ВНИМАНИЕ!

ВОПРОСЫ?

Java for students

Филиал кафедры ПОИТ БГУИР в Eram Systems
курс: Веб-технологии (JAVA)

Author: Olga Smolyakova , PhD

Oracle Certified Java 6 Programmer

[Olga Smolyakova@epam.com](mailto:Olga_Smolyakova@epam.com)