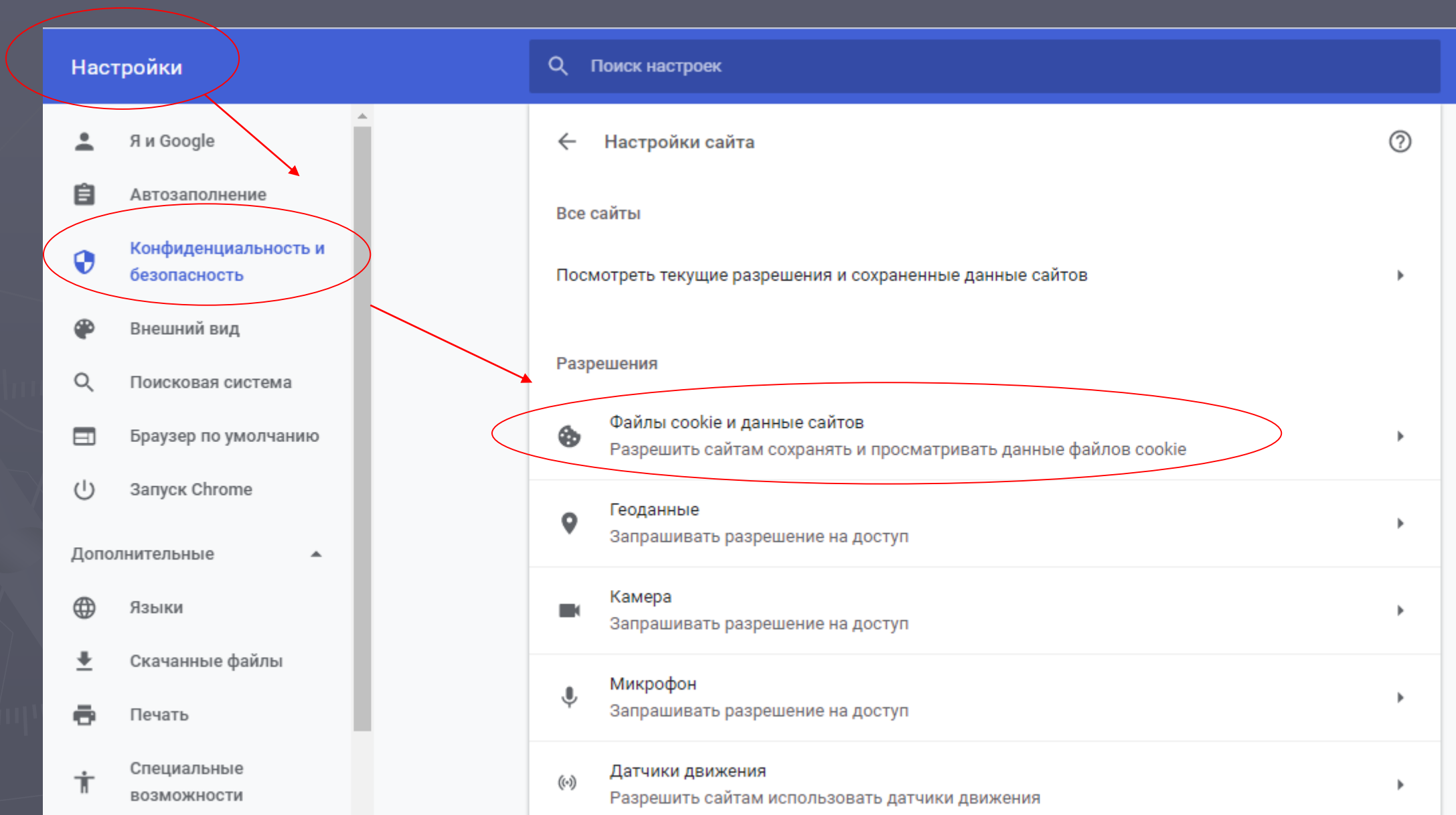













# Java EE

## Filter, Listener, Cookies

# Cookies



live.worldbank.org 1 файл cookie	▶	
live13.livejournal.com Локальное хранилище	▶	
livebook.manning.com Локальное хранилище, 1 файл cookie	▶	
livedune.ru Хранилище для базы данных, Локальное хранилище	▶	
livetyping.com Локальное хранилище, Файловая система	▶	
lkqd.net 3 файла cookie	▶	
lmi.sc.omtrdc.net 2 файла cookie	▶	
loadercdn.net 1 файл cookie	▶	
localhost Хранилище для базы данных, Локальное хранилище, 3 файла cookie	▶	

## ← Локальные данные сайта aj2021.online

UUID

Название

UUID

Контент

d607b150-770d-11ea-8fd0-02421b1ecf45

Домен

.aj2021.online

Путь

/

Отправка

только при защищенном подключении

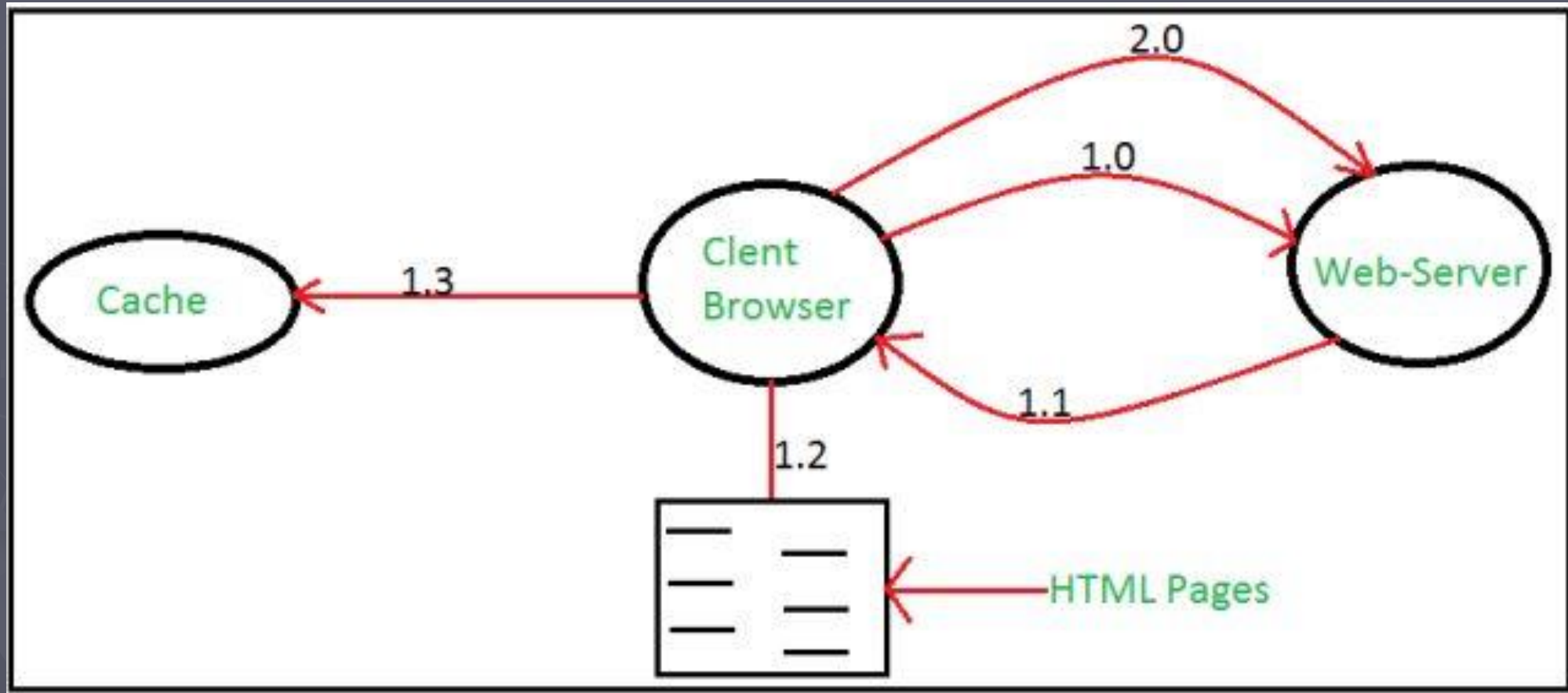
Доступно для скрипта

Да

Создано

воскресенье, 5 апреля 2020 г., 10:21:45

# Working process



# Файлы Cookie

- ▶ Класс `javax.servlet.http.Cookie`
- ▶ Cookie — это небольшие блоки текстовой информации, которые сервер посылает клиенту для сохранения в файлах cookies
- ▶ Можно запрещать
- ▶ Возврат на сервер как часть заголовка HTTP, когда клиент повторно заходит на тот же веб-ресурс
- ▶ Ассоциация с доменом или сервером

## Создание

```
Cookie cookie = new Cookie("user", "Bstu 1234");  
response.addCookie(cookie);
```

## Извлечение

```
Cookie[ ] cookies = request.getCookies();
```

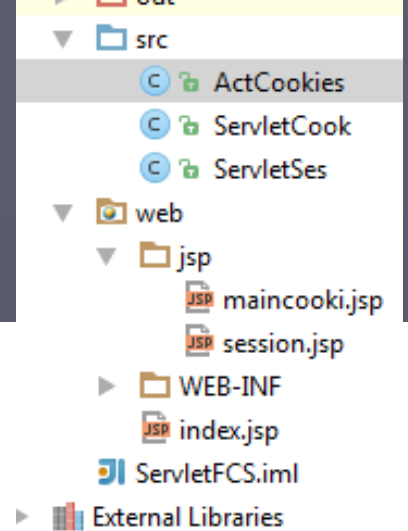
getValue()

Параметры:

путь, домен, номер версии, время жизни,  
комментарий.

setMaxAge(long sec)

```
public class ActCookies {  
    private static int number = 1;  
  
    public static void setCookie(HttpServletResponse resp) {  
        String name = "PNV";  
        String role = "moderator" + number++;  
        Cookie c = new Cookie(name, role);  
        c.setMaxAge(60 * 60); // время жизни файла cookie  
        resp.addCookie(c); // добавление cookie к объекту-ответу  
        String value = resp.getLocale().toString();  
        Cookie loc = new Cookie("locale", value);  
  
        resp.addCookie(loc);  
    }  
}
```



```
public static ArrayList<String> addToRequest(HttpServletRequest request)
{
    ArrayList<String> messages = new ArrayList();
    Cookie[] cookies = request.getCookies();
    if (cookies != null) {
        messages.add("Number cookies : " + cookies.length);
        for (int i = 0; i < cookies.length; i++) {
            Cookie c = cookies[i];
            messages.add(c.getName() + " = " + c.getValue());
        } // end for
    } // end if
    return messages;
}
}
```



```

@WebServlet(name = "ServletCook" , urlPatterns = "/ServletCookie")
public class ServletCook extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

    }

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

        ActCookies.setCookie(response); // добавление cookie
        // извлечение cookie и добавление информации к request
        request.setAttribute("messages",
                               ActCookies.addToRequest(request));
        request.getRequestDispatcher("/jsp/maincooki.jsp").
            forward(request, response);
    }
}

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<html><head><title>from Cookie </title></head>
<body>
    ${messages}      [Number cookies : 5, PNV = moderator1, locale = ru_RU, JSESSIONID = bcb48a8503f6c323c5e725d44ead, __AntiXsrfToken = 5027de3aacc84313bb4f3f50df1ae47652e52dae609b6f11c1c7765ee6]
</body></html>

```

# Listener

- ▶ Отслеживание изменений состояния некоторых объектов приложения  
(сеанс, контекст и запрос сервлета, генерируемыми во время жизненного цикла веб-приложения)
- ▶ Интерфейсы - позволяют следить за событиями, связанными с сеансом, контекстом и запросом сервлета

Интерфейсы и их методы
<b>ServletContextListener</b> contextInitialized(ServletContextEvent e) contextDestroyed(ServletContextEvent e)
<b>HttpSessionListener</b> sessionCreated(HttpSessionEvent e) sessionDestroyed(HttpSessionEvent e)
<b>ServletContextAttributeListener</b> attributeAdded(ServletContextAttributeEvent e) attributeRemoved(ServletContextAttributeEvent e) attributeReplaced(ServletContextAttributeEvent e)
<b>HttpSessionAttributeListener</b> attributeAdded(HttpSessionBindingEvent e) attributeRemoved(HttpSessionBindingEvent e) attributeReplaced(HttpSessionBindingEvent e)
<b>HttpSessionBindingListener</b> valueBound(HttpSessionBindingEvent e) valueUnbound(HttpSessionBindingEvent e)
<b>HttpSessionActivationListener</b> sessionWillPassivate(HttpSessionEvent e) sessionDidActivate(HttpSessionEvent e)
<b>ServletRequestListener</b> requestDestroyed(ServletRequestEvent e) requestInitialized(ServletRequestEvent e)
<b>ServletRequestAttributeListener</b> attributeAdded(ServletRequestAttributeEvent e) attributeRemoved(ServletRequestAttributeEvent e) attributeReplaced(ServletRequestAttributeEvent e)

## Регистрация блока прослушивания

@WebListener

```
public class SessionListenerDemo
    implements HttpSessionAttributeListener {
    public void attributeRemoved(HttpSessionBindingEvent ev) {
    }
    public void attributeAdded(HttpSessionBindingEvent ev) {
        // запись в log-файл или иные действия
        System.out.println("add: " + ev.getClass().getSimpleName()
            + " : " + ev.getName()
            + " : " + ev.getValue());
    }
    public void attributeReplaced(HttpSessionBindingEvent ev) {
        // запись в log-файл или иные действия
        System.out.println("replace: " + ev.getClass().getSimpleName()
            + " : " + ev.getName()
            + " : " + ev.getValue());
    }
}
```

# Зарегистрировать обработку событий МОЖНО в элементе `<listener>`

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/
  version="3.1">

  <context-param>
    <param-name>administrator</param-name>
    <param-value>Jaroslav Juk</param-value>
  </context-param>

  <listener>
    <listener-class>listener.SessionListenerDEmo</listener-class>
  </listener>

</web-app>
```

```
@WebListener()
public class RequestListenerDemo implements ServletRequestListener {

    public void requestInitialized(ServletRequestEvent ev) {
        // будет использован для получения информации о запросе
        HttpServletRequest req = (HttpServletRequest) ev.getServletRequest();
        String uri = "Request Initialized for " + req.getRequestURI();
        String id = "Request Initialized with ID=" + req.getRequestedSessionId();
        System.out.println(uri + "\n" + id);
        ServletContext context = ev.getServletContext();
        // счетчик числа созданных запросов
        Integer reqCount = (Integer) req.getSession().getAttribute("counter");
        if (reqCount == null) {
            reqCount = 0;
        }
        context.log(uri + "\n" + id + ", Request Counter =" + reqCount);
    }

    public void requestDestroyed(ServletRequestEvent ev) {
        System.out.println("Request Destroyed: "
            + ev.getServletRequest().getAttribute("lifecycle"));
    }
}
```

```
[2018-05-20T21:08:19.044+0300] [glassfish 5.0] [INFO] [] [javax.enterprise.web] [tid: _ThreadID=29 _ThreadName=http-listener-1(4)
WebModule[/jspxml_war_exploded] ServletContext.log():Request Initialized for /jspxml_war_exploded/FirstServletTest
Request Initialized with ID=bd8bfccce0ecf7e82cece893333d, Request Counter =0]]
```

```
[2018-05-20T21:08:19.051+0300] [glassfish 5.0] [INFO] [] [javax.enterprise.web] [tid: _ThreadID=29 _ThreadName=http-listener-1(4)
WebModule[/jspxml_war_exploded] ServletContext.log():This is message from first servlet]]
```

```
[2018-05-20T21:08:19.053+0300] [glassfish 5.0] [INFO] [] [] [tid: _ThreadID=29 _ThreadName=Thread-8] [timeMillis: 1526839699053]
```

# Filter

## ► интерфейс Filter

Позволяет создать объект, который перехватывает запрос, может трансформировать заголовок и содержимое запроса клиента.

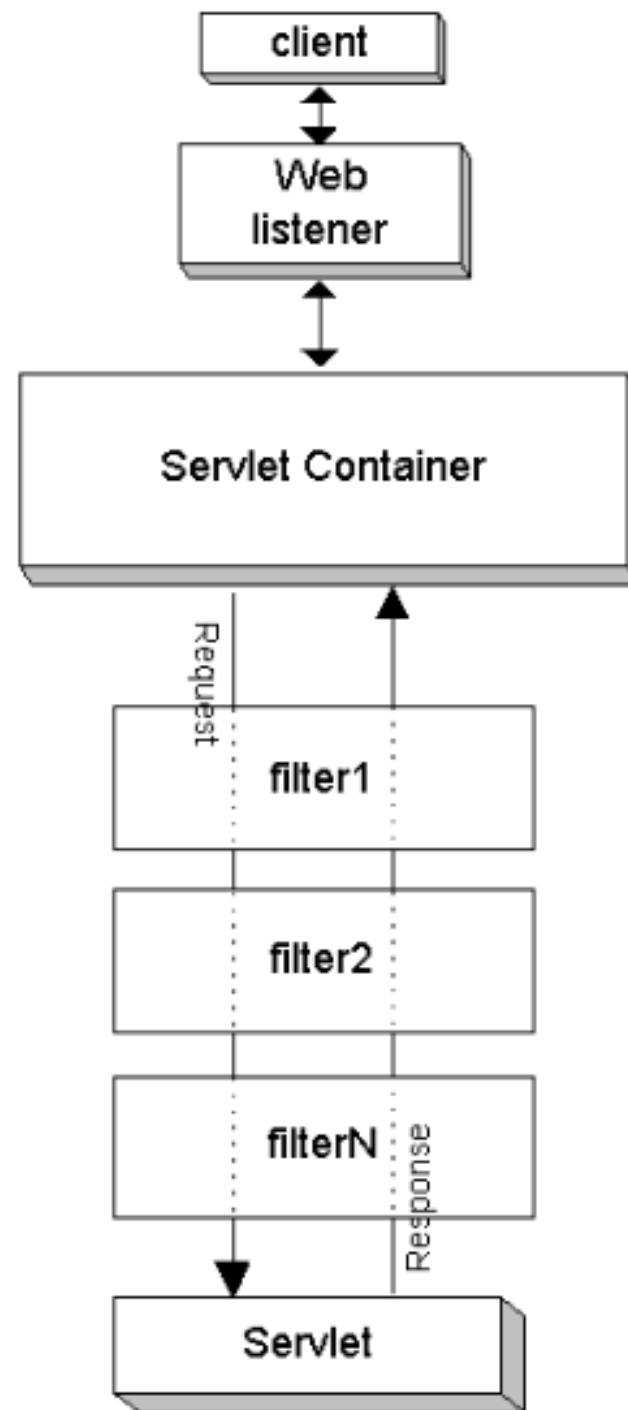
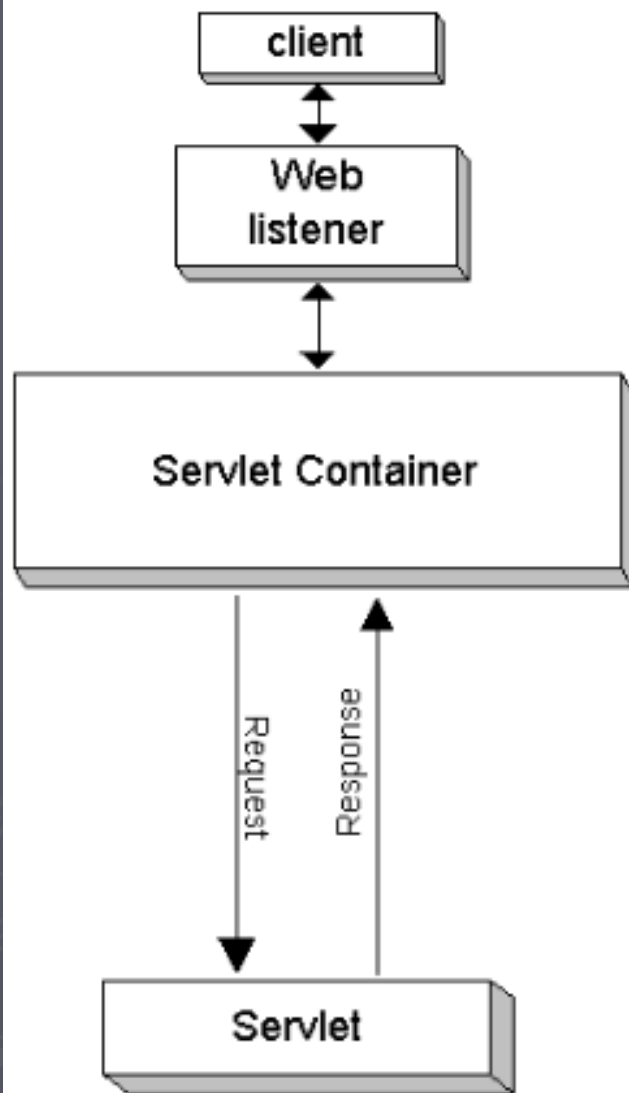
- 1) Предварительная обработка
- 2) Модификация запроса/ответа клиента



# Примеры фильтров

- ▶ Authentication filters
- ▶ Logging and auditing filters
- ▶ Image conversion filters
- ▶ Data compression filters
- ▶ Encryption filters
- ▶ Tokenizing filters
- ▶ Filters that trigger resource access events
- ▶ XSL/T filters that transform XML content
- ▶ MIME-type chain filters
- ▶ Caching filters





Browser

Web App

page2

page1



redirect to page2

page2



page2



filter 1

filter 2

filter 3

page1

# Основные действия

- перехват инициализации сервлета или jsp 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 и/или jsp.

пакет javax.servlet

Интерфейсы Filter, FilterChain и FilterConfig

### *Жизненный цикл фильтра*

- 1) void init(FilterConfig config)
- 2) doFilter(ServletRequest request,  
ServletResponse response, FilterChain chain)  
chain.doFilter()
- 3) destroy()

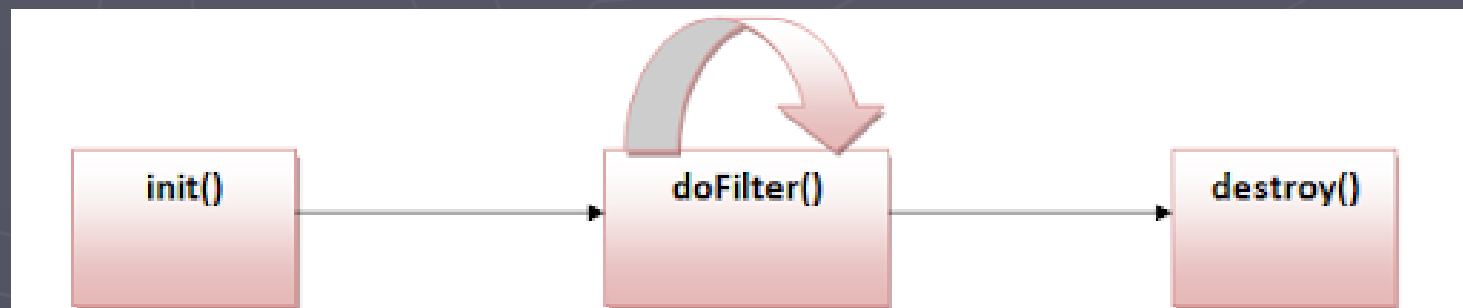
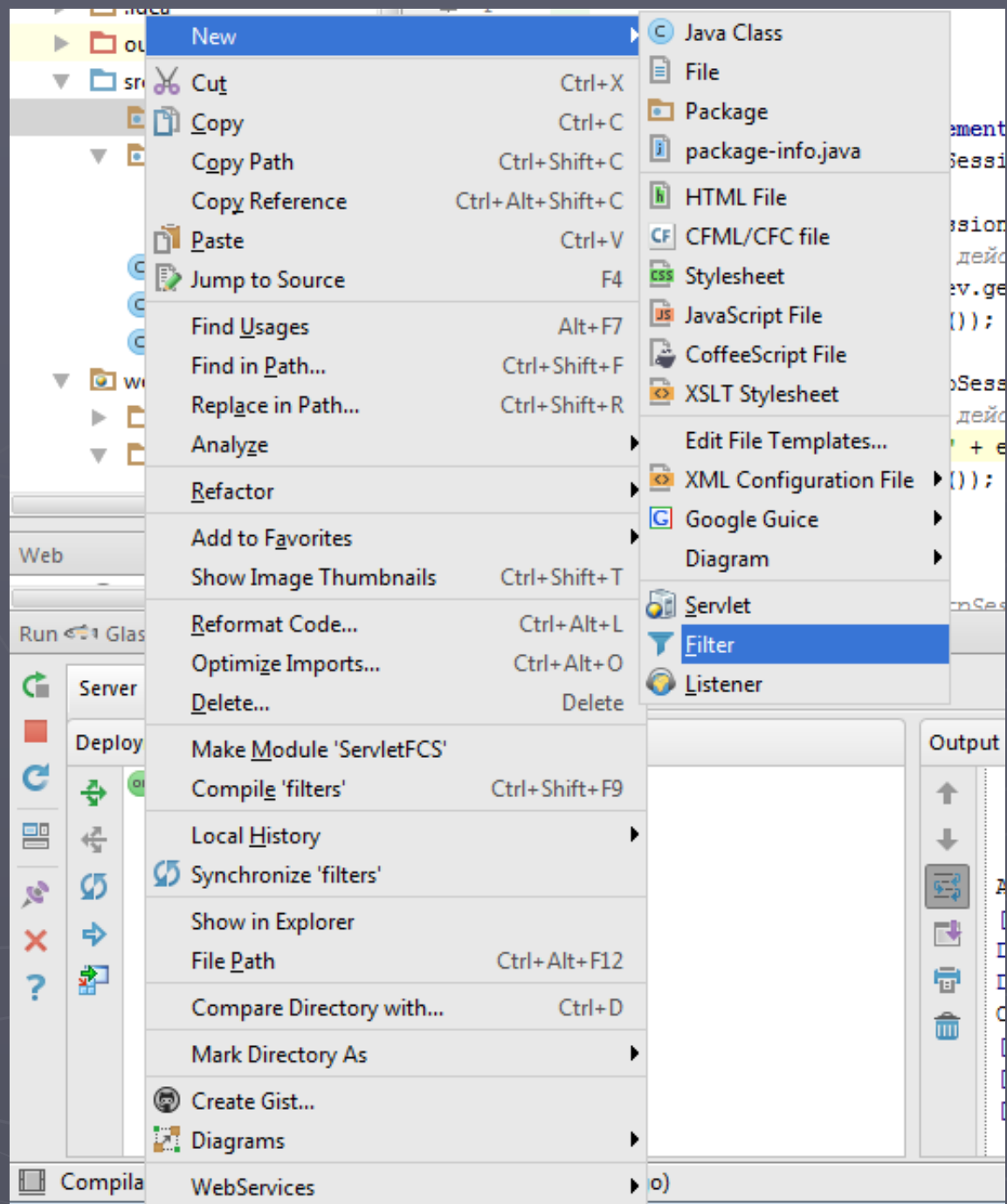


Fig - Filter Lifecycle

# Фильтр установки кодировки



```
@WebFilter(urlPatterns = { "/"* } ,
    initParams = {
        @WebInitParam(name = "encoding",
            value = "UTF-8",
            description = "Encoding Param") })

public class EncodingFilter implements Filter {
    private String code;

    public void init(FilterConfig fConfig) throws ServletException {
        code = fConfig.getInitParameter("encoding");
    }

    public void doFilter(ServletRequest request,
        ServletResponse response,
        FilterChain chain)
        throws IOException, ServletException {
        String codeRequest = request.getCharacterEncoding();
        // установка кодировки из параметров фильтра, если не установлена
        if (code != null && !code.equalsIgnoreCase(codeRequest)) {
            request.setCharacterEncoding(code);
            response.setCharacterEncoding(code);
        }
        chain.doFilter(request, response);
    }

    public void destroy() {
        code = null;
    }
}
```

# Можно настраивать в web.xml

```
<filter>
  <filter-name>encodingfilter</filter-name>
  <filter-class>filters.EncodingFilter</filter-class>
  <init-param>
    <param-name>encoding</param-name>
    <param-value>UTF-8</param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>encodingfilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

```
WebFilter(urlPatterns = { "/"* } ,  
    initParams = {  
        @WebInitParam(name = "encoding",  
            value = "UTF-8",  
            description = "Encoding Param") })
```

## ► Шаблоны

- |                       |                |
|-----------------------|----------------|
| 1) все сервлеты и JSP | "/*"           |
| 2) только для JSP     | "*.jsp"        |
| 3) для каталога       | "/jsp/admin/*" |
| 4) для конкретной jsp | "/index.jsp"   |
| 5) для сервлета       | "/controller"  |



## ► Пример - Запрещение несанкционированного прямого обращения к JSP

```
@WebFilter(filterName = "PageRedirectFilter",
    urlPatterns = { "/jsp/*" },
    initParams = { @WebInitParam(name = "INDEX_PATH", value = "/index.jsp") }
)

public class PageRedirectFilter implements Filter {
    private String indexPath;
    public void init(FilterConfig fConfig) throws ServletException {
        indexPath = fConfig.getInitParameter("INDEX_PATH");
    }

    public void doFilter(ServletRequest request, ServletResponse response,
        FilterChain chain) throws IOException, ServletException {
        HttpServletRequest httpRequest = (HttpServletRequest) request;
        HttpServletResponse httpResponse = (HttpServletResponse) response;
        // переход на заданную страницу
        httpResponse.sendRedirect(httpRequest.getContextPath() + indexPath);
        chain.doFilter(request, response);
    }
    public void destroy() {
    }
}
```

# Перехват обращений к сервлету

```
public enum ClientType {  
  
    GUEST, USER, ADMINISTRATOR  
  
}
```

```
@WebFilter(urlPatterns = { "/controller" }, servletNames = { "MainServlet" })  
public class SecurityFilter implements Filter {  
    public void destroy() {  
    }  
    public void doFilter(ServletRequest request, ServletResponse response,  
                        FilterChain chain) throws IOException, ServletException {  
        HttpServletRequest req = (HttpServletRequest) request;  
        HttpServletResponse resp = (HttpServletResponse) response;  
        HttpSession session = req.getSession();  
        ClientType type = (ClientType) session.getAttribute("userType");  
        if (type == null) {  
            type = ClientType.GUEST;  
            session.setAttribute("userType", type);  
            RequestDispatcher dispatcher = request.getServletContext()  
                .getRequestDispatcher("/jsp/guest.jsp");  
            dispatcher.forward(req, resp);  
            return;  
        }  
        // pass the request along the filter chain  
        chain.doFilter(request, response);  
    }  
    public void init(FilterConfig fConfig) throws ServletException {  
    }  
}
```