25.03.2020
Servlets API

Намиот Д.Е.
dnamiot@gmail.com

HTTP сессии

Идентификация сессии присутствует в запросе. Соответственно, это все методы класса HttpServletRequest:

- String getRequestedSessionId()

возвращает идентификатор текущей сессии или null

- HttpSession getSession()

возвращает объект HttpSession, связанный с идентификатором сессии из запроса. Если нет идентификатора или сессии нет, то создается новая. Важно – объект на сервере. В запросе – только идентификатор !

- HttpSession getSession(boolean create)

возвращает объект HttpSession, связанный с идентификатором сессии из запроса. Если нет идентификатора или сессии нет, то, в зависимости от логического параметра создается (не создается) новая. Важно – объект на сервере. В запросе – только идентификатор !

HTTP сессии

- boolean isRequestedSessionIdFromCookie()

Проверяет, что ID сессии передавался с помощью cookie

- boolean isRequestedSessionIdFromURL()

Проверяет, что ID сессии передавался в URL.

- boolean isRequestedSessionIdValid()

Проверяет, что ID сессии еще может использоваться (соответствует некоторому объекту на сервере)

HTTP сессии

Методы класса HttpSession

- long getCreationTime()

Время создания в миллисекундах аосле January 1, 1970 GMT.

- String getId()

Возвращает идентификатор сессии

- long getLastAccessedTime()

Время последнего запроса клиента с таким идентификатором сессии в миллисекундах после January 1, 1970 GMT. Время доступа отмечает контейнер.

- int getMaxInactiveInterval()

Максимальный интервал в секундах между запросами. Время неактивности. Если интервал между запросами больше, сессия не может быть использована (isRequestedSessionIdValid()).

HTTP сессии

- void invalidate()

Помечает текущую сессию как недоступную. Объекты, содержащиеся в сессии могут быть удалены.

- boolean isNew()

Новая сессия или нет. После вызова getSession() или getSession (true) можно проверить – существовала ли сессия ранее или была создана заново.

- void setMaxInactiveInterval(int interval)

Задание интервала неактивности в секундах.

HTTP сессии

Сам объект HttpSession организован как база ключ-значение. Объекты хранятся, по крайней мере, на все время существования сессии

- void setAttribute(String name, Object value)

Сохраняет объект в сессии по заданному ключу (имени)

- Object getAttribute(String name)

Получает объект из сессии по ключу (имени) (null, если по заданному ключу ничего нет)

- Enumeration<String> getAttributeNames()

Получает список ключей (имен объектов) из сессии.

- void removeAttribute(String name)

Удаляет из сессии объект по ключу

Схема использования

- В методе doPost() или doGet(), который обрабатывает запрос пользователя получаем экземпляр объекта HttpSession
  HttpSession sess = request.getSession();

- Если это первое обращение – то сессия создается

- При необходимости сохранить что-то “в сессии”:
  String code = request.getParameter(“code”);
  sess.setAttribute (“mykey”, code);

  получили код товара и записали в сессию. Сохранили товар в корзине

- Для получения объектов из сессии (checkout)
  String code = (String)sess.getAttribute(“mykey”);

Примеры

```
<web-app>
 <servlet>
    <servlet-name>Servlet1</servlet-name>
    <servlet-class>MyServlet1</servlet-class>
 </servlet>
<servlet>
   <servlet-name>Servlet2</servlet-name>
   <servlet-class>MyServlet2</servlet-class>
</servlet>

<servlet-mapping>
   <servlet-name>Servlet1</servlet-name>
   <url-pattern>/login</url-pattern>
</servlet-mapping>
<servlet-mapping>
   <servlet-name>Servlet2</servlet-name>
   <url-pattern>/welcome</url-pattern>
</servlet-mapping>
</web-app>
```

Примеры

Главная страница (index.htm)

```
<form action="login">
  User Name:<input type="text" name="userName"/><p/>
  Password:<input type="password" name="userPassword"/><br/>
  <br/>
  <input type="submit" value="Ok"/>
</form>
```

Примеры

```
// Servlet1
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class MyServlet1 extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse response){
  try{
   response.setContentType("text/html");
   PrintWriter pw = response.getWriter();

  String name = request.getParameter("userName");
  String password = request.getParameter("userPassword");
  pw.print("Hello "+name);
  pw.print("Your Password is: "+password);
  HttpSession sess=request.getSession();
  sess.setAttribute("uname",name);
  sess.setAttribute("upass",password);
  pw.print("<a href='welcome'>view details</a>");
  pw.close();
   }catch(Exception exp){
  System.out.println(exp);      }   } }
```

Примеры

Выдача сервлета:

View details

View details

Примеры

```
// Servlet2
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class MyServlet2 extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse response){
  try{
      response.setContentType("text/html");
      PrintWriter pw = response.getWriter();
      HttpSession sess=request.getSession(false);
      String myName=(String)sess.getAttribute("uname");
      String myPass=(String)sess.getAttribute("upass");
      pw.print("Name: "+myName+" Pass: "+myPass);
      pw.close();
  }catch(Exception exp){
      System.out.println(exp);
   }
  }
}
```

Корзина (эл. коммерция)

```
class Cart{
    String name;
    int qty;
// Getter & Setter
   public void setName(String name) { this.name =
name; }
 public String getName() { return this.name; }

 public void setQty(int qty) { this.qty = qty; }
 public int getQty() { return this.qty; }
}
```

Корзина (эл. коммерция)

```
String name=req.getParameter("n");
String qty=req.getParameter(“q");

HttpSession s=req.getSession();

List<Cart> list= (List<Cart>) s.getAttribute("list");

 if(list==null){  list =new ArrayList<>();  }
Cart c= new Cart();
c.setName(name);
c.setQty(Integer.parseInt(qty));
// Add the name & cost to List
list.add(c);
s.setAttribute("list",list);
```

Валидация данных

- Проверки на стороне клиента

```
<input type="number" name=“qty" />
<input type="text" name="name" pattern="[0-9]" />
```

- Проверки на стороне сервера

```
int q;
If (qty != null)
{
   try
   {
      q = Integer.parseInt(qty);
   }
   catch (Exception ex { … }
}
```

RequestDispatcher

//RequestDispatcher

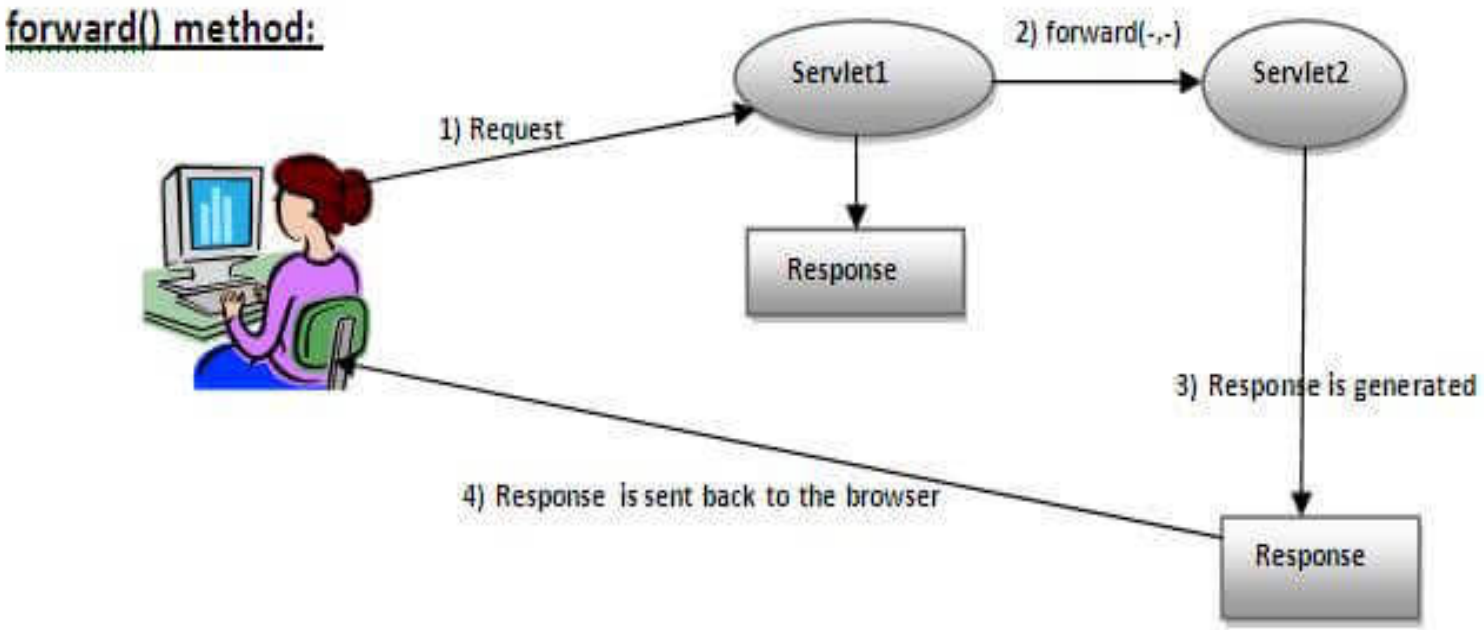
getRequestDispatcher(String path)
Возвращает объект *RequestDispatcher,* который содержит ссылку на указанный ресурс.

Ресурс может быть статическим или динамическим. Два метода

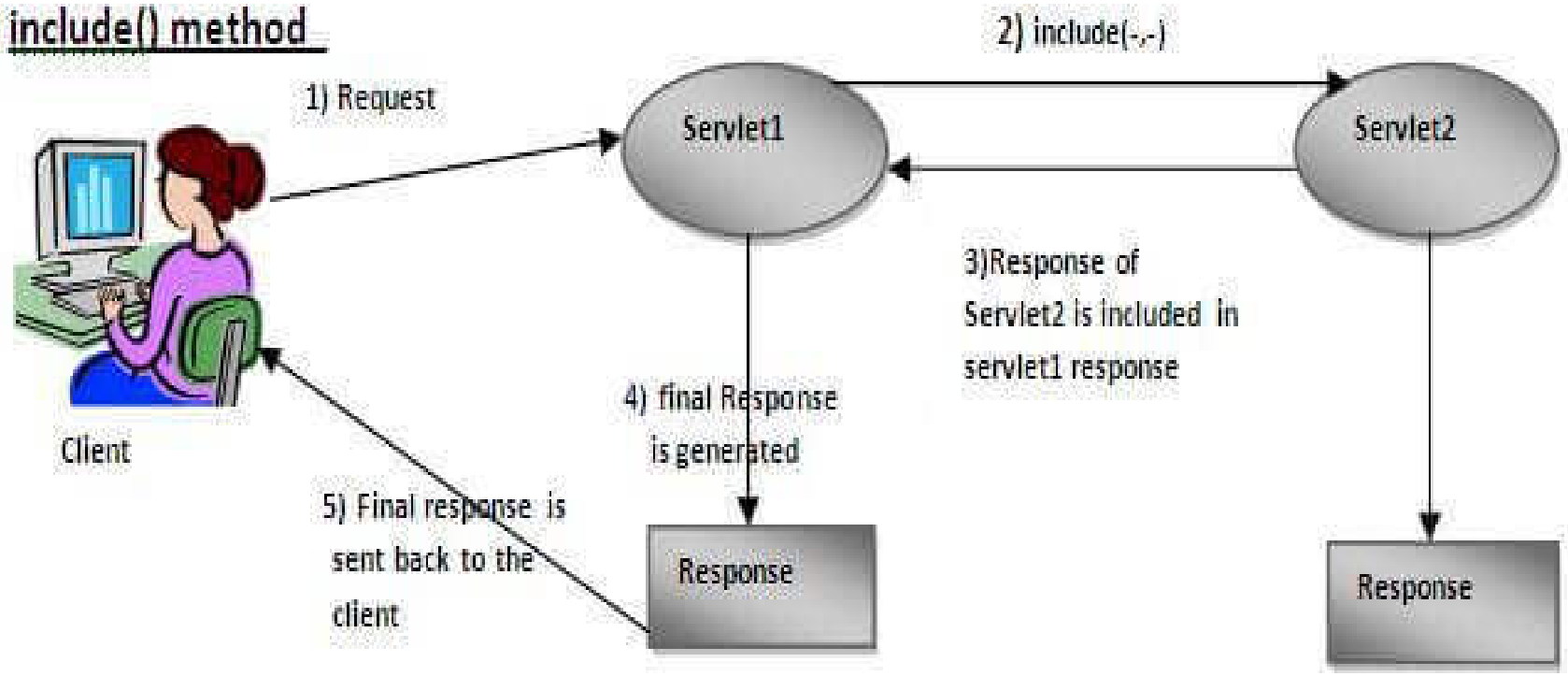
void forward(ServletRequest request, ServletResponse response)
Переправляет запрос другому ресурсу (servlet, JSP file, or HTML file) на сервере.

void include(ServletRequest request, ServletResponse response)
Включает содержимое другого ресурса (servlet, JSP page, HTML file) в ответ (отклик)

RequestDispatcher



RequestDispatcher



RequestDispatcher

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class Validation extends HttpServlet
{
   public void doPost(HttpServletRequest request,
     HttpServletResponse response)
       throws ServletException, IOException
   {
       response.setContentType("text/html");
       PrintWriter pw = response.getWriter();
       String name=request.getParameter("name");
       String pass=request.getParameter("pass");
       if(“Guest”.equals(name) && “12345”.equals(“pass"))
       {
          RequestDispatcher dis=request.getRequestDispatcher("welcome");
          dis.forward(request, response);         }
       else
       {
          pw.print("User name or password is incorrect!");
          RequestDispatcher dis=request.getRequestDispatcher("index.html");
          dis.include(request, response);
       }
   }    }
```

RequestDispatcher

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class Validation extends HttpServlet
{
   public void doPost(HttpServletRequest request,
     HttpServletResponse response)
       throws ServletException, IOException
   {
       response.setContentType("text/html");
       PrintWriter pw = response.getWriter();
       String name=request.getParameter("name");
       String pass=request.getParameter("pass");
       if(“Guest”.equals(name) && “12345”.equals(“pass"))
       {
          request.setAttribute (“name”, “User: ”+name);
          RequestDispatcher dis=request.getRequestDispatcher("welcome");
          dis.forward(request, response);         }
       else
       {
          pw.print("User name or password is incorrect!");
          RequestDispatcher dis=request.getRequestDispatcher("index.html");
          dis.include(request, response);
       }
   }    }
```

RequestDispatcher

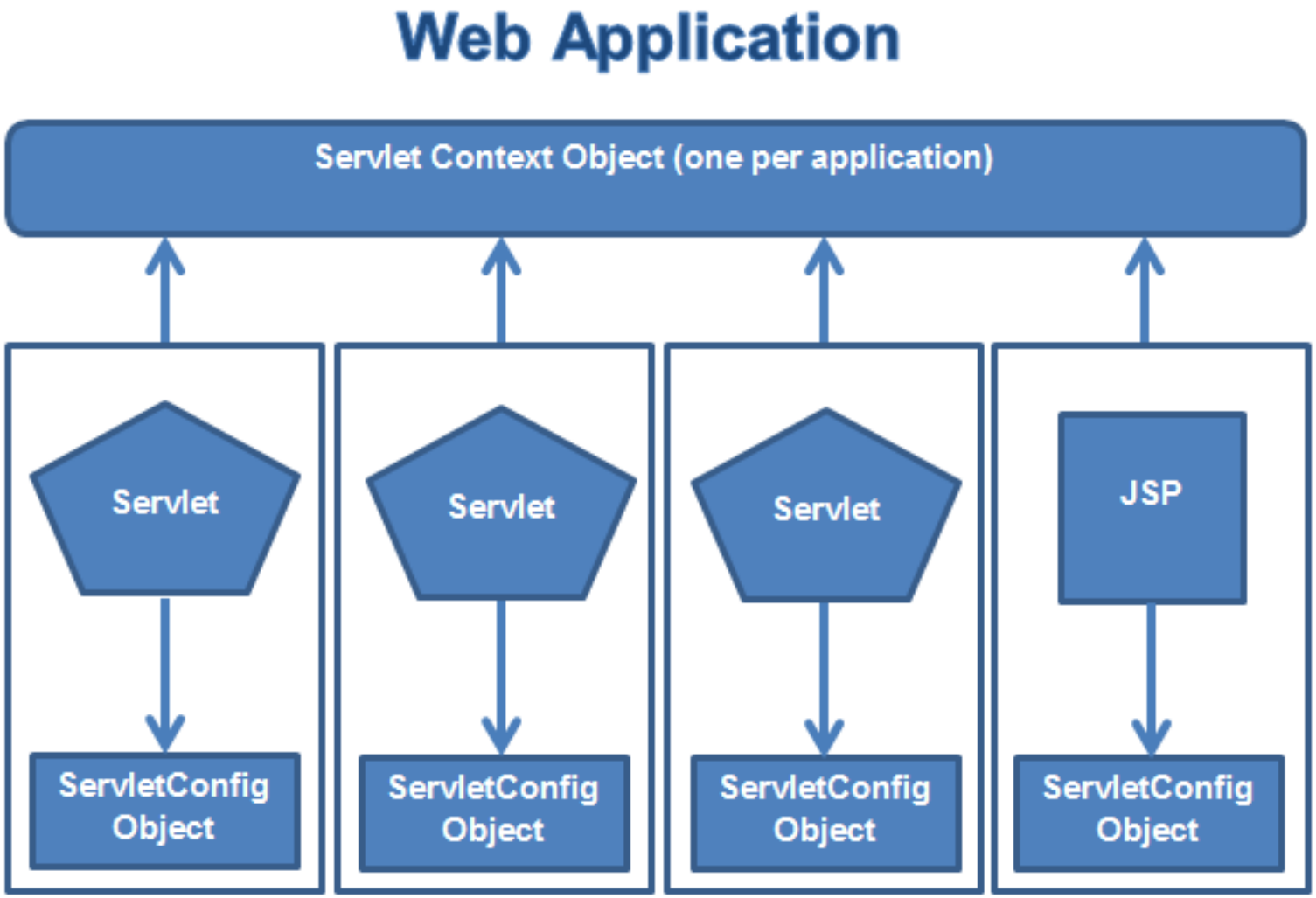
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class WelcomeUser extends HttpServlet {

 public void doPost(HttpServletRequest request,
 HttpServletResponse response)
 throws ServletException, IOException
 {
     response.setContentType("text/html");
    PrintWriter pw = response.getWriter();

    String name=request.getParameter("name");
    pw.print("Hello "+name+"!");
    pw.print(" Attribute: “ + request.getAttribute(“name”));  }
}
```

ServletContext



ServletContext

- voidsetAttribute(java.lang.String name, java.lang.Object object)

Сохранение объекта

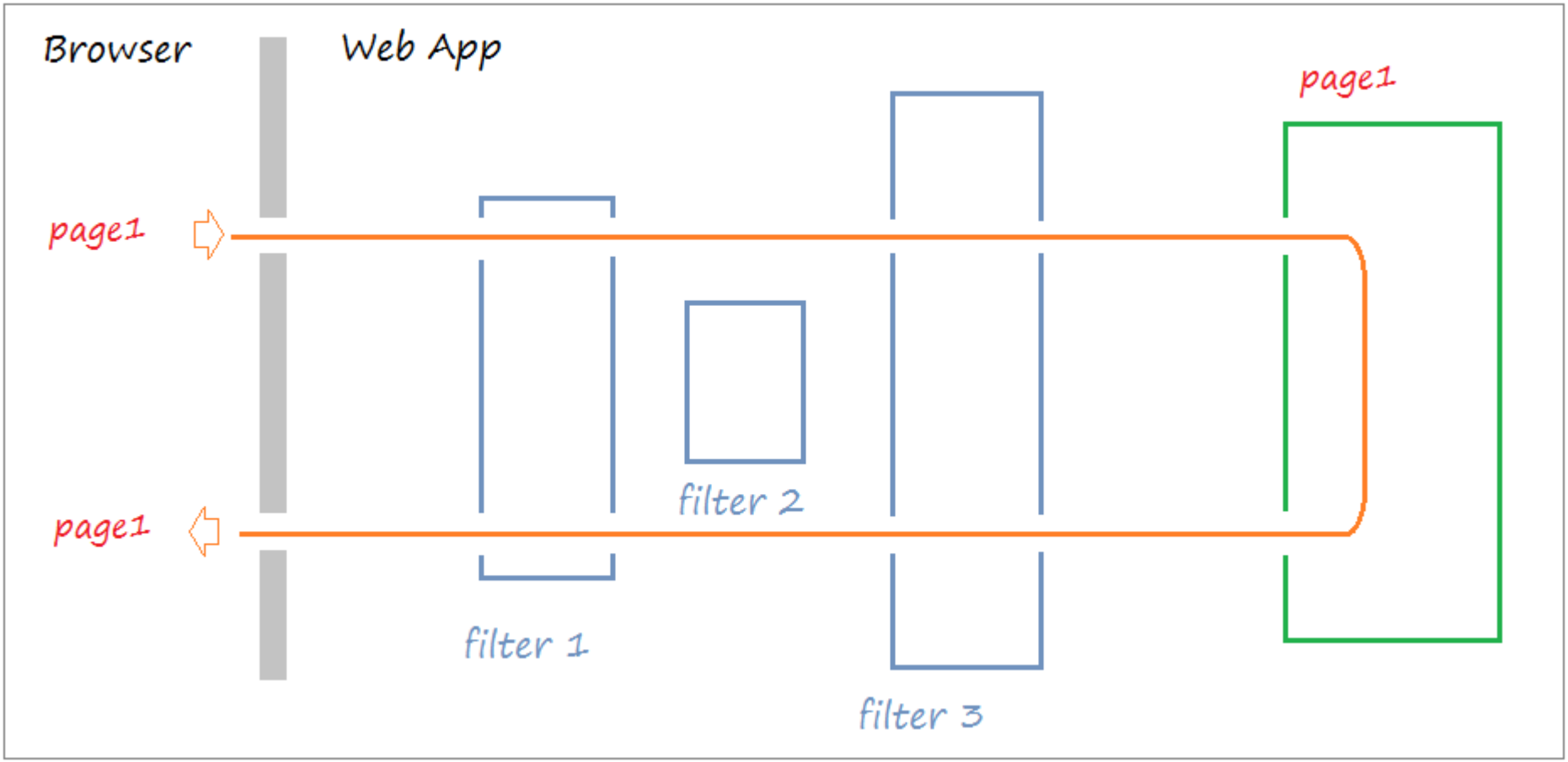
- void removeAttribute(java.lang.String name)

Удаление объекта из ServletContext.

- java.lang.Object getAttribute(java.lang.String name)

Извлечение объекта

Фильтры



Фильтры

```
public interface Filter
{
void      init(FilterConfig filterConfig)
//Инициализация

void      destroy()
//Выгрузка приложения

void      doFilter(ServletRequest request, ServletResponse response,
FilterChain chain)
//Бизнес-процесс
}
```

Фильтры

```
<filter>
  <filter-name>MyFilter</filter-name>
  <filter-class>com.MyFilter</filter-class>
  <init-param>
  <param-name>test</param-name>
  <param-value>testValue</param-value>
  </init-param>
</filter>
```

Фильтры

```
<filter-mapping>
  <filter-name>MyFilter</filter-name>
  <url-pattern>/*</url-pattern>
<!-- either url-pattern or servlet-name is mandatory -->
  <servlet-name>LoginServlet</servlet-name>
  <dispatcher>REQUEST</dispatcher>
</filter-mapping>
```

Может работать для клиентских запросов и/или RequestDispatcher

Фильтры

```
import javax.servlet.*;
@WebFilter("/*")
public class MyFilter 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 {
                // pre-processing
                chain.doFilter(request, response);
                // post-processing
        }
public void destroy() {
                //we can close resources here
        } }
```

Фильтры: примеры

```
public void doFilter(ServletRequest request, ServletResponse response,
FilterChain chain) throws IOException, ServletException {
                        if (request.getRemoteAddr().startsWith(“192.168”))
                        {
                          chain.doFilter(request, response);
                         }
                         else
                          response.sendRedirect(“not-allowed.html”);
        }
```

Фильтры: примеры

```
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()+"}");  }  }

// pass the request along the filter chain
chain.doFilter(request, response);  }
```