Казанский федеральный УНИВЕРСИТЕТ

Казанский федеральный УНИВЕРСИТЕТ
ВЫСШАЯ ШКОЛА
информационных технологий
и интеллектуальных систем

03. Технологии WEB

Лекции по информатике
для студентов второго курса Высшей школы ИТИС КФУ
2020

Ференец Александр Андреевич
старший преподаватель кафедры программной инженерии

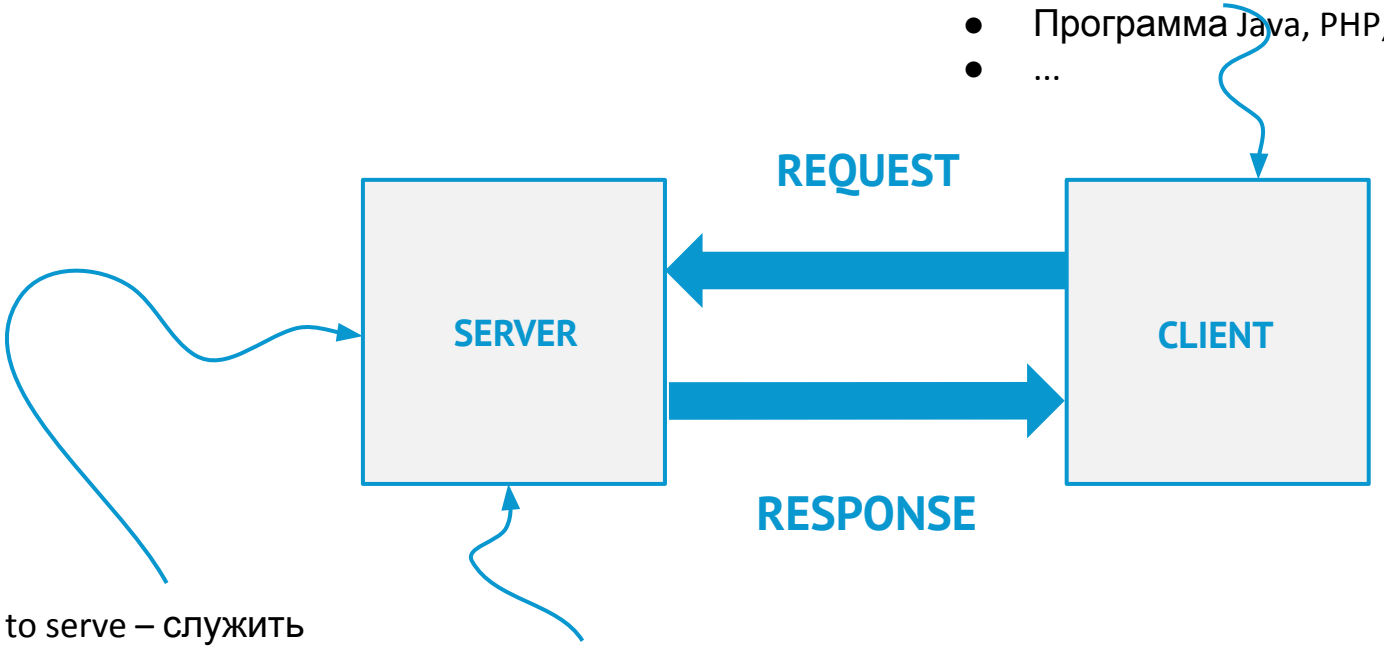
С использованием материалов
к. т. н., доцента кафедры программной инженерии Абрамского М.М.

aferenets@it.kfu.ru

V. 1.0

HTTP. В очередной раз про архитектуру

- Браузер
- Мобильное приложение
- Программа Java, PHP, ...
- ...



WEB-Server, но бывают другие для других протоколов

HTTP. Stateless/statefull?

State – состояние.

Statefull. Запустили приложение, оно хранит переменные запустившего его пользователя в ОЗУ. Есть хранение состояния.

Stateless. Попросили приложение выполнить какие-то операции, оно нам вернуло результат работы. Разошлись. Нет хранения состояния.

HTTP – ???

HTTP. Stateless!

State – состояние.

Statefull. Запустили приложение, оно хранит переменные запустившего его пользователя в ОЗУ. Есть хранение состояния.

Stateless. Попросили приложение выполнить какие-то операции, оно нам вернуло результат работы. Разошлись. Нет хранения состояния.

HTTP – stateless!

Даже HTTP/2 со всеми сохранения соединений и проч.

HTTP. Stateless для сайтов?

Открываем сайт, вводим логин-пароль в форму авторизации, пользуемся от своего имени. То есть **сайт помнит нас**! То есть сайт на стороне сервера узнаёт нас без ввода логина-пароля каждый раз.

Он должен узнавать нас либо по IP и прочим косвенным признакам, либо должен помнить состояние (для каждого пользователя отдельно).

Сохранение состояния HTTP. Проблема

Вариант: хранить данные в БД на сервере (SQL/noSQL не важно, хоть CSV).

А как поймём, что запрос №1273612 – запрос от того же пользователя, что и запрос №1273522 (поступил 5 минут назад), чтобы достать из огромной базы запросов/соединений инфу?

СОХРАНЕНИЕ СОСТОЯНИЕ HTTP. Решение №1

Вариант: хранить данные у пользователя.

!

Если после первого запроса пользователя попросить его сохранить и при всех следующих запросах высылать определённую инфу. То мы можем ничего не хранить сами.

!

COOKIES. Основные идеи

Любой сайт
(веб-приложение на конкретном доменном имени)

Коллекция из
(ключ, значение, …)

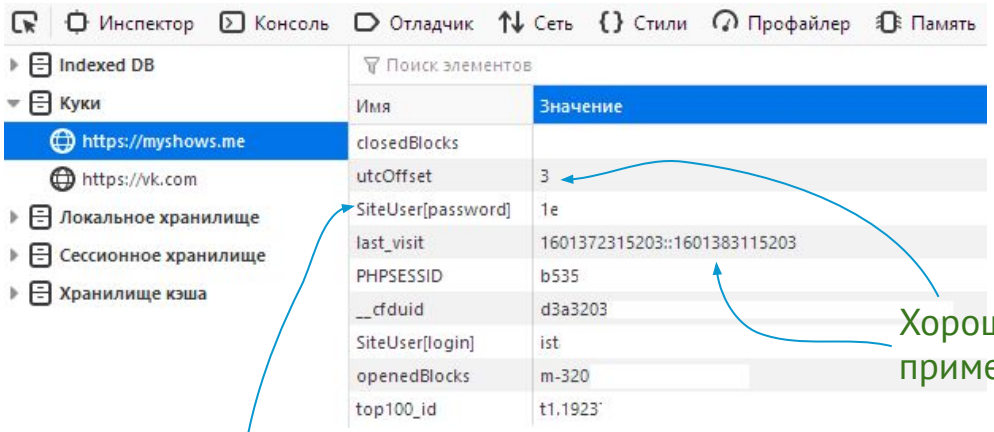
→ **БРАУЗЕР**

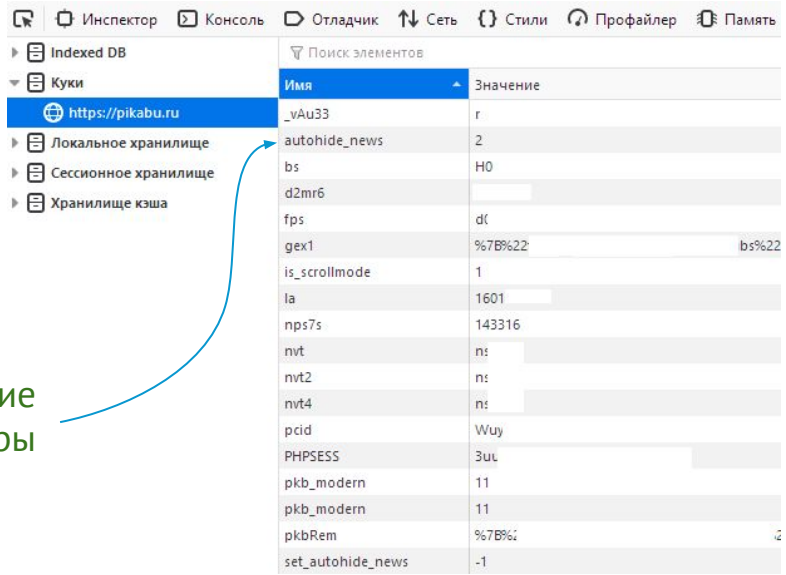
А что браузеру нужно сохранить в итоге?

1. Name
2. Value
3. Domain (может быть IP-адрес)
4. Path (для контекста и прочих “подсайтов”)
5. Expires/Max-age
6. + secure, httponly, samesite

Чтобы знать, кому высылать

COOKIES. Примеры



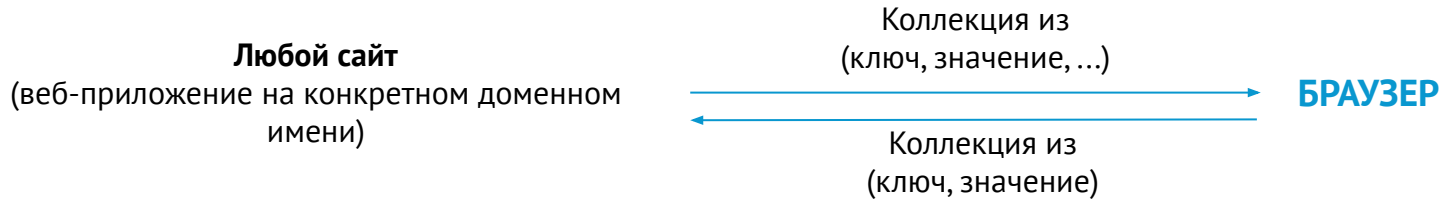


Хорошие примеры

Плохой пример

Не всё следует хранить в cookies (пояснение будет дальше)

COOKIES. Передача клиенту и приём сервером



Пример заголовка, который посылает сервер

```
Set-Cookie: name=value; Max-age=31536000; Domain=example.com; Path=/; Secure; Httponly;
```

Пример заголовка, который посылает клиент

```
Cookie: name=value; name2=value2
```

+ Можно устанавливать cookie сразу в браузере через JavaScript

COOKIES. Управление в Java

Нужно послать клиенту данные. Как это делается в сервлете?

COOKIES. Управление в Java

Нужно послать клиенту данные. Как это делается в сервлете?

```
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
    ...
}
```

addCookie

```
void addCookie(Cookie cookie)
```

Adds the specified cookie to the response. This method can be called multiple times to set more than one cookie.

Parameters:

`cookie - the Cookie to return to the client`

А как удалить?
Спойлер: у HttpServletResponse нет метода deleteCookie.

COOKIES. Управление в Java

Нужно послать клиенту данные. Как это делается в сервлете?

```
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
    ...
}
```

addCookie

```
void addCookie(Cookie cookie)
```

Adds the specified cookie to the response. This method can be called multiple times to set more than one cookie.

Parameters:

`cookie - the Cookie to return to the client`

А как удалить?
Спойлер: у HttpServletResponse нет метода deleteCookie.

Установить cookie с нужным именем и прошедшим Expires или нулевым MaxAge.

COOKIES. Управление в Java

Как получить cookies? Их нам присылает клиент каждый раз, значит….

```
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
    …
}
```

getCookies

```
Cookie[] getCookies()
```

Returns an array containing all of the `Cookie` objects the client sent with this request. This method returns `null` if no cookies were sent.

Returns:

an array of all the Cookies included with this request, or null if the request has no cookies

А ещё есть HTTP-авторизация. Но Cookie универсальнее.

СОХРАНЕНИЕ СОСТОЯНИЕ HTTP. Решение №2 (странное)

Добавлять ко всем-всем ссылкам (и формам) нужные параметры. Пользователь при пользовании сайтом кликает по ссылкам -> шлёт на сервер GET-параметры.

http://google.com/someAction?name=value&name2=value2

СОХРАНЕНИЕ СОСТОЯНИЕ HTTP. Решение №1+3

Храним данные на сервере. А в Cookies пишем идентификатор клиента.

Алгоритм поведения.
Поступил запрос. В Cookies есть индентификатор?

a. Есть. Достаём из базы связанные данные. Используем. Возможно добавляем новые данные в базу для этого идентификатора.

b. Нет. Выделяем место в базе. Просим клиент запомнить идентификатор в надежде, что при следующем запросе нам его пришлют.

СЕССИЯ. Основные идеи

1. Выделяется cookie со специальным именем
   a. JSESSIONID
   b. PHPSESSID
   c. …
2. Генерируется уникальное значение и записываемся в cookie с именем из п.1
3. Выделяется место на сервере, привязывается уже сгенерированное уникальное значение.
4. Если пришёл пользователь с Cookie с соответствующим именем, ищем сессию у себя на сервере. Если нашли, то используем/обновляем данные. Если нет, то начинаем с п.1.

У сессии обычно устанавливается время жизни порядка 20-30 минут.

СЕССИЯ. Управление в Java

Для программиста обычно нет разницы, как хранятся данные сессии.
Главное: что можно положить почти любой объект и он сериализуется.

getSession

`HttpSession getSession()`

Returns the current session associated with this request, or if the request does not have a session, creates one.

Returns:

the HttpSession associated with this request

See Also:

`getSession(boolean)`

Основные методы:

- `Object getAttribute(String name)`
- `void removeAttribute(String name)`
- `void setAttribute(String name, Object value)`

С осторожностью:

- `void invalidate()`
- `void setMaxInactiveInterval(int interval)`

В известном всем файле настройки сайта:

```
<session-config>
    <session-timeout>10</session-timeout>
    <cookie-config>
        <name>MY_JSESSIONID_</name>
    </cookie-config>
</session-config>
```

Если мы возьмём чужие все Cookie или JSESSIONID, для сайта мы будем выглядеть этим другим человеком. Мы же фактически забрали его статус.

Это не повод отключать работу с ними в браузере!

Как можно достать чужие Cookie?

REDIRECT. Общая идея

Что такое редирект? Попросить клиент сделать запрос куда-нибудь ещё.
Зачем? Переезд сайта или временная смена обрабатывающего скрипта. Есть другие причины.

Что такое редирект для HTTP?

REDIRECT. Общая идея

Что такое редирект? Попросить клиент сделать запрос куда-нибудь ещё.
Зачем? Переезд сайта или временная смена обрабатывающего скрипта. Есть другие причины.

Что такое редирект для HTTP?

Ответить клиенту со статусным кодом 301 (Moved Permanently) или 302 (Found).

А как серверу указать клиенту, куда “переехал” документ?

REDIRECT. Общая идея

Что такое редирект? Попросить клиент сделать запрос куда-нибудь ещё.
Зачем? Переезд сайта или временная смена обрабатывающего скрипта. Есть другие причины.

Что такое редирект для HTTP?

Ответить клиенту со статусным кодом 301 (Moved Permanently) или 302 (Found).

А как серверу указать клиенту, куда “переехал” документ?

Добавить заголовок Location.

```
HTTP/1.1 302 Found
Date: Wed, 27 JUL 2016 11:20:59 GMT
Location: http://example.com/another
```

REDIRECT. Использование в Java

Сделать редирект – отправить клиенту информацию, значит, в сервлете это можно сделать через объект HTTPServletResponse:

```
void  sendRedirect(String location)
```

Либо эквивалентно:

```
void setStatus(int sc)
        +
void setHeader(String name, String value)
```

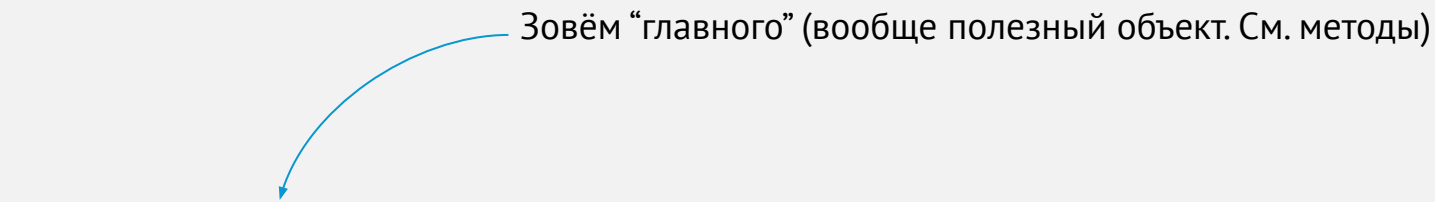
FORWARD. Основная идея

Зачем, когда?
Если один обработчик (например, сервлет для Java) начал обрабатывать запрос, а продолжить должен другой, надо передать выполнение.

Банальное решение: вызвать другой обработчик. Ну… функцию вызвать.

Почему неидеальное решение:
Надо передать объект запроса/ответа, надо корректно инициировать, надо зарегистрировать новую итерацию обработки (вдруг будет рекурсия без условия выхода).

FORWARD. Использование в Java




```
getServletContext()
    .getRequestDispatcher("/path")
    .forward(req, resp);
```

В других языках/фреймворках аналогично: позвать главного, вызвать местный forward.

```
$subRequest = $request->duplicate($query, null, $path);
$this->getContainer()->get('http_kernel')->handle($subRequest, HttpKernelInterface::SUB_REQUEST);
```

Symfony, PHP

Пре- или пост-обработка запроса

Чаще после фильтра запрос передается сервлету.

```
@WebFilter(“/admin/*”)
public class FilterConnect implements Filter {
     public void doFilter (ServletRequest request,
                           ServletResponse response,
                           FilterChain chain) throws IOException, ServletException{
          if (request.getSession.getAttribute("username") != null){
               // Идем дальше
               chain.doFilter(request, response);
          }
     }
}
```

В других языках/фреймворках встретите скорее более обобщённое понятие слушателей событий (Listener).

Соответственно “зоопарк” событий будет гораздо шире, а не только “пришёл запрос”, “готов ответ”.

ПОЛЬЗОВАТЕЛЬ. Процессы

- Регистрация – процесс первоначальной записи данных о пользователе.

Ввод логина и пароля пользователя на сайт – ?

ПОЛЬЗОВАТЕЛЬ. Процессы

- Регистрация – процесс первоначальной записи данных о пользователе.
- Аутентификация – процесс подтверждения пользователя того, что он тот, за кого себя выдаёт.
- Авторизация – процесс выдачи прав на определённые действия.
- Идентификация – ??? (ещё фильм про спецагента есть)

ПОЛЬЗОВАТЕЛЬ. Процессы

- Регистрация – первоначальной записи данных о пользователе.
- Аутентификация – процесс подтверждения пользователя того, что он тот, за кого себя выдаёт.
- Авторизация – процесс выдачи прав на определённые действия.
- Идентификация – процесс связи объекта с ранее известной информации (установление тождественности известному).

СЕРВЛЕТЫ. Проблема спагетти-кода

Чего больше нужного в этом коде – Java или HTML?

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) … {
    ….
    PrintWriter out = response.getWriter();
    out.println("<html>");
    out.println("<head>");
    out.println("<title>Братиш, сколько время?</title>");
    out.println("</head>");
    out.println("<body>");
    out.println("<h1>Время: " + (new java.util.Date()) + "</h1>");
    out.println("</body>");
    out.println("</html>");
    out.close();
}
```

JSP. Вывоз мусора или элегантное решение?

Jakarta Server Pages

Содержит:
Статические элементы (html, css, js),
JSP-элементы, генерирующие содержимое.

JSP-страница превращается в сервлет с помощью компилятора Jasper, входящего в Tomcat (а дальше понятно). То есть закинув на сайт jsp-файл, сделав запрос к серверу на соот.ресурс, потом можно в недрах сервера найти сервлет с кучей print'ов.

JSP. Перемещаем мусор из сервлетов (плохой код)


```
<html>
<head>
    <title>Э, парень, сколько время?</title>
</head>
<body>
    <h1>Время: <%= new java.util.Date() %> </h1>
</body>
</html>
```

Чем это лучше PHP? =\

JSP. Пример плохого PHP-скрипта


```
1   ...
2   <body>
3   <?php
4       $friends = array(…);
5       if (!empty($friends)) {
6           echo "<ul>";
7           foreach($friends as $index => $friend) {
8               echo "<li><a href='".$friend->page_url."'>".$friend->name."</a></li>";
9           }
10          echo "</ul>";
11      } else {
12         echo "<h1>You have no friends.</h1>"
13      }
14  ?>
15  </body>
```

JSP. Уже шаблонизатор


```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

...
<c:if test = "${not empty users}">
    <c:forEach items="${users}" var="user">
    <li>
        <c:out value="${user.username}"></c:out>
    </li>
    </c:forEach>
</c:if>
...
```

ШАБЛОНИЗАТОРЫ. Пример – Twig


```
<!DOCTYPE html>
<html>
    <head>
        <title>My Webpage</title>
    </head>
    <body>
        <ul id="navigation">
        {% for item in navigation %}
            <li><a href="{{ item.href }}">{{ item.caption }}</a></li>
        {% endfor %}
        </ul>

        <h1>My Webpage</h1>
        {{ a_variable }}
    </body>
</html>
```

ШАБЛОНИЗАТОРЫ. Что это и зачем нужны?

Шаблонизатор – система генерации вывода программы обычно со своим индивидуальным языком описания управляющих конструкций, функциями обработки выводимых данных.

Особенности:

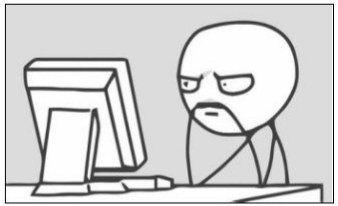
- Свой ограниченный язык. Нельзя напортачить на уровне возможностей Java.
- Ограниченный набор переменных. Нельзя получить доступ к ядру системы или к БД. Обработчик запроса предоставит только необходимые данные.
- Набор полезных функций, “хелперов” для обработки данных (дат, чисел, строк).
- Упрощённый синтаксис. Синтаксический сахар. Хорошо для верстальщиков (точнее тех, кто между ними и бэкэндером, если там кто-то есть =)).

Минусы:

- Работает медленнее, так как нужен ещё один интерпретатор.
- Другие минусы, следующие из особенностей.

JSP+JSTL (почти шаблонизатор), Freemarker, Twig, Slim, Jinja

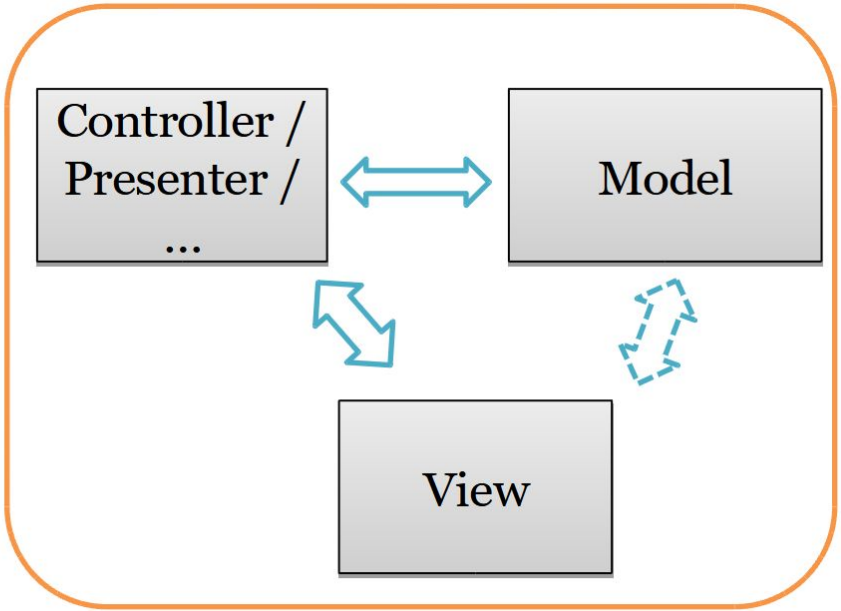
Архитектура. Первые мысли



Клиент

request

response



Сервер

SESSION,COOKIE. Демонстрация

<CODE>

Демонстрация использования сессии. Обзор создаваемых cookie. Пример сайта со авторизацией.