

ПМЗ Разработка модулей ПО.

РО 3.1 Понимать и применять принципы объектно-ориентированного и асинхронного программирования.

Тема 2. Разрабатывать модули с применением DOM API Regexp HTTP.

Лекция 3. Cookies, сессии и базовая аутентификация.

Цель занятия:

**Понять, как в «безсостоятельном»
протоколе HTTP реализуется
хранение состояния, авторизация
пользователей и работа с сессиями.**

Учебные вопросы:

1. Cookies

2. Сессии

3. Базовая аутентификация

1. Cookies

Ключевые понятия: **Stateful vs. Stateless**

По своей сути, HTTP является stateless-протоколом, что означает отсутствие состояния.

Каждый запрос, который браузер отправляет на сервер, абсолютно независим от предыдущих.

Сервер не «помнит» вас или ваши предыдущие действия.

Stateless (Без состояния):

Сервер обрабатывает каждый HTTP-запрос как новый, не имея никакой информации о предыдущих запросах от того же клиента.

Аналогия: Представьте, что вы звоните в справочную, и каждый раз, когда вы задаете новый вопрос, вам отвечает новый оператор, который не знает, о чем вы говорили с предыдущим.

Stateful (С состоянием):

Сервер сохраняет информацию о клиенте и его взаимодействиях в течение определенного периода.

Аналогия: Тот же звонок в справочную, но теперь у оператора есть ваша "карта клиента", и он видит всю историю ваших обращений.

Именно из-за того, что HTTP по умолчанию stateless, нам нужны дополнительные механизмы, такие как cookies и сессии.

Они создают «мост» между независимыми запросами, позволяя нам сохранять информацию о пользователе и его действиях.

Куки и сессии позволяют имитировать stateful-поведение поверх stateless-протокола, что критически важно для создания интерактивных и персонализированных веб-приложений.

Куки (cookies) — это небольшие текстовые файлы, которые сервер отправляет в браузер пользователя. Браузер сохраняет их и при каждом последующем запросе к тому же серверу автоматически отправляет их обратно.

🍪 Главная роль куки — отслеживание состояния пользователя.

Они позволяют серверу "узнать" вас, запомнить ваши предпочтения, содержимое корзины в интернет-магазине или статус входа в систему, что делает веб-серфинг более персонализированным и удобным.

Структура куки:

Каждый куки имеет набор атрибутов, которые определяют его поведение:

- **name**: Имя куки (например, `user_id`).
- **value**: Значение куки (например, `12345`).
- **expires** или **Max-Age**: Устанавливает срок жизни куки. Если этот атрибут не задан, куки будет сессионным и удалится, когда вы закроете браузер. **Max-Age** задает время в секундах, а **expires** — конкретную дату.
- **path**: Определяет путь на сервере, для которого куки действителен. Например, если `path=/docs`, куки будет отправляться только для запросов к `/docs` и его поддиректориям (e.g., `/docs/info`).

- **domain:** Указывает домен, для которого куки доступен. Например, если `domain=example.com`, куки будет отправляться и на `www.example.com`, и на `blog.example.com`.
- **Secure:** Флаг, который требует, чтобы куки отправлялись только по HTTPS. Это предотвращает перехват данных при использовании небезопасного соединения.
- **HttpOnly:** Флаг, который запрещает доступ к куки через JavaScript (`document.cookie`). Это важная мера безопасности, которая помогает предотвратить XSS-атаки, когда злоумышленник может получить доступ к кукам через инъекцию вредоносного скрипта.

Жизненный цикл куки:

- **Создание:** Сервер отправляет HTTP-заголовок Set-Cookie в ответ на запрос. Например: Set-Cookie: user_id=12345; Max-Age=3600; HttpOnly.
- **Хранение:** Браузер получает этот заголовок и сохраняет куки на компьютере пользователя.
- **Отправка:** При каждом последующем запросе к тому же домену и пути браузер автоматически добавляет заголовок Cookie с сохраненными куками.
- **Обработка:** Сервер получает куки, обрабатывает их и может изменить или удалить.

2. Сессии.

Сессия — это механизм для хранения информации о пользователе на стороне сервера.

В отличие от куки, которые хранятся прямо в браузере пользователя, сессия позволяет серверу "запомнить" вас, даже если вы переходите со страницы на страницу.

Она создает временное состояние для каждого пользователя, пока он взаимодействует с приложением.

Ключевое отличие: куки хранит данные на клиенте, а сессия — на сервере.

Для того чтобы связать клиента с его сессией на сервере, используется идентификатор сессии (Session ID).

Этот Session ID — это уникальный ключ, который сервер создает и отправляет браузеру пользователя. Обычно он хранится в куки или передается как часть URL.

Механизм работы

Работа сессии — это последовательность простых шагов:

- Первый запрос. Клиент (браузер) отправляет первый запрос на веб-сервер.
- Создание сессии. Сервер получает запрос, создает уникальный Session ID и сохраняет его вместе с данными сессии (например, имя пользователя: 'Иван') в своей памяти или базе данных.
- Отправка Session ID. Сервер отправляет ответ клиенту, включая в него заголовок Set-Cookie с нашим Session ID. Браузер сохраняет этот Session ID как обычный куки.
- Последующие запросы. При каждом следующем запросе к этому же серверу браузер автоматически отправляет сохраненный Session ID. Сервер получает Session ID, ищет по нему соответствующие данные сессии и "узнаёт" пользователя.

Главное преимущество сессий — это безопасность.
Поскольку вся важная информация хранится на сервере, нет риска, что злоумышленник получит доступ к ней, перехватив или изменив данные на компьютере пользователя.

3. Базовая аутентификация.

Процесс базовой аутентификации состоит из нескольких шагов:

- Запрос на защищенный ресурс. Пользователь пытается получить доступ к ресурсу, который требует аутентификации.
- Вызов аутентификации (401 Unauthorized). Сервер, обнаружив отсутствие учетных данных, немедленно возвращает ответ со статусом 401 Unauthorized. В этот ответ сервер добавляет специальный заголовок WWW-Authenticate: Basic realm="...". Параметр realm (область) — это текст, который браузер покажет пользователю в окне для входа.
- Ввод учетных данных. Браузер, получив ответ с кодом 401 и заголовком WWW-Authenticate, автоматически выводит стандартное окно для ввода логина и пароля.

- Кодирование и повторный запрос. После того как пользователь ввел данные, браузер:
 - Объединяет логин и пароль через двоеточие: логин:пароль.
 - Кодирует эту строку в формате Base64. Например, admin:123 превращается в YWRtaW46MTIz.
 - Отправляет новый запрос на тот же ресурс, добавляя в него заголовок Authorization: Basic <закодированная_строка>.
- Проверка на сервере. Сервер получает этот заголовок, декодирует строку из Base64, проверяет учетные данные и, если они верны, предоставляет доступ к ресурсу.

Безопасность и недостатки:

Основной недостаток базовой аутентификации — это её небезопасность.

Несмотря на кодирование в Base64, данные передаются практически в открытом виде.


Base64 — это не шифрование, а простой способ преобразования двоичных данных в текст.

Любой, кто перехватит сетевой трафик (например, через публичный Wi-Fi), может легко декодировать строку и получить доступ к логину и паролю.

Вывод: Базовую аутентификацию обязательно использовать вместе с протоколом HTTPS. HTTPS шифрует весь трафик между браузером и сервером, включая заголовок Authorization, что делает передачу данных безопасной и защищает их от перехвата.

Куки и сессии — это разные механизмы, но они работают вместе, как ключ и замок, чтобы помочь веб-серверу запомнить вас.

Куки: Ключ в вашем кармане

Куки — это небольшой текстовый файл, который сервер отправляет в ваш браузер. Браузер хранит его на вашем компьютере. Представьте, что это ключ , который выдали вам на входе в здание. Каждый раз, когда вы возвращаетесь, вы показываете этот ключ, и вас узнают.

Где хранятся: на вашем компьютере.

Что хранят: имя пользователя, настройки темы сайта, содержимое корзины. Обычно это небольшие, нечувствительные данные.

Уязвимость: если в куки хранить конфиденциальные данные (пароль), они могут быть перехвачены.

Сессии: Информация у охранника

Сессия — это информация о вас, которая хранится на сервере. Это как карточка посетителя, которая лежит у охранника 🧑🔒. На этой карточке записано, кто вы, вошли ли вы в систему и к каким ресурсам у вас есть доступ. Сам по себе браузер не знает, что это за карточка.

Где хранятся: на сервере.

Что хранят: ваш статус входа в систему, права доступа, важные данные, которые не должны быть видны клиенту.

Как это работает: Чтобы "связать" ваш браузер с вашей сессией на сервере, используется Session ID (идентификатор сессии). Этот Session ID — это просто уникальный код, который сервер отправляет вам в виде куки.

Как они работают вместе?

Вы заходите на сайт. Сервер создает для вас сессию, генерирует уникальный Session ID и записывает его в свой файл (или базу данных).

Сервер отправляет этот Session ID в ваш браузер, упаковав его в куки.

Когда вы переходите на другую страницу, ваш браузер автоматически отправляет этот куки с Session ID обратно на сервер.

Сервер получает куки, находит по Session ID нужную сессию и "вспоминает" все данные о вас.

Итог: Куки — это просто место для хранения (на клиенте).
Сессия — это набор данных о пользователе (на сервере).
Куки часто используется как инструмент для передачи Session ID, чтобы сессия могла работать.

Итог:

- используйте куки для простых данных
- сессии для безопасности и конфиденциальности
- базовую аутентификацию только в связке с HTTPS.

Контрольные вопросы:

- Что такое cookie и для чего они нужны в HTTP?
- Каким заголовком сервер устанавливает cookie у клиента?
- Как браузер отправляет cookie обратно на сервер?
- Что означают атрибуты cookie:
 - Secure
 - HttpOnly
 - SameSite
- Почему HTTP называют «безсостоятельным» протоколом?
- Как cookie помогают реализовать сессию?
- Где хранятся данные сессии — в браузере или на сервере?
- Как работает базовая аутентификация (Basic Auth)?
- Чем опасна Basic Auth при использовании без HTTPS?
- Какие альтернативные способы аутентификации используются в современных веб-приложениях?

Домашнее задание:

1. https://ru.hexlet.io/courses/http_protocol

Материалы лекций:

<https://github.com/ShViktor72/Education2025>

Обратная связь:

colledge20education23@gmail.com