ПМ3 Разработка модулей ПО.

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

Тема 2. Протокол HTTP .

Лекция 15. Тело запроса, query string и отправка форм.

Цель занятия:

Понять, как данные передаются в HTTP-запросах, чем отличаются query string и тело запроса, какие существуют способы кодирования данных при отправке форм.

Учебные вопросы:

1. Query string.
2. Тело HTTP-запроса.
3. Сравнение GET и POST.
4. Кодировка параметров.

1. Query string.

Query string (строка запроса) — это часть URL, которая содержит пары «**ключ=значение**» и используется для передачи параметров от клиента к серверу.

◇ Структура URL с query string

URL может состоять из нескольких частей:

[протокол]://[домен]/[путь]?[параметры]

- **протокол** → http:// или https:// (какой протокол используется).
- **Домен** (host) → example.com (куда идёт запрос).
- **Путь** (path) → /resource (какой ресурс на сервере).
- **Query string** (строка запроса) → после знака ? идут параметры.

◇ Как устроены параметры

После **?** записываются пары ключ=значение.

- Несколько параметров соединяются с помощью **&**.
- Пробелы и спецсимволы кодируются (%20 вместо пробела).

Пример:

/search?user=neo&city=zion

Разбор:

- Путь → /search
- Query string → user=neo&city=zion
- Параметры:
  - user = neo
  - city = zion

◇ Когда используется query string

- Поиск:

  /search?q=matrix

- Фильтрация:

  /products?category=books&sort=price

- Пагинация (разбиение на страницы):

  /articles?page=2&limit=10

🔷 Особенности

- Параметры видны в адресной строке → удобно для ссылок, но нельзя хранить пароли.
- Длина ограничена (обычно до ~2000 символов).
- Порядок параметров обычно неважен (a=1&b=2 = b=2&a=1).
- Можно передавать несколько значений для одного ключа:

  /search?tag=js&tag=html&tag=css

👉 То есть, query string — это просто «список параметров», прикреплённый к адресу.

Сервер при получении URL разбирает ключи и значения и использует их для обработки запроса.

Примеры:

🔷 1. Фильтрация и сортировка

DummyJSON (товары)

https://dummyjson.com/products/category/smartphones?limit=3&sort=price

👉 вернёт товары из категории smartphones, ограничит 3 штуками, отсортирует по цене.

🔷 2. Поиск

JSONPlaceholder (фейковые посты)

https://jsonplaceholder.typicode.com/posts?userId=1&q=sunt

👉 вернёт посты пользователя с userId=1, где встречается sunt.

Open Library (книги)

https://openlibrary.org/search.json?q=matrix

👉 поиск книг по слову «matrix».

🔷 3. Передача небольших данных (пагинация, лимит)

Reqres (пользователи)

https://reqres.in/api/users?page=2&per_page=3

👉 в ответе будут пользователи со второй страницы, по 3 на страницу.

PokeAPI

https://pokeapi.co/api/v2/pokemon?limit=5&offset=10

👉 вернёт 5 покемонов, начиная с 11-го.

🔷 4. Использование в GET и POST

Иногда параметры передаются и в query string, и в теле POST.

Пример (httpbin):

POST https://httpbin.org/post?token=123

Body: { "username": "neo" }

👉 token=123 идёт в query string, а username=neo — в теле запроса.

2. Тело HTTP-запроса.

Тело HTTP-запроса (request body) — часть запроса, которая передаёт данные от клиента к серверу.

В отличие от query string, данные не вставляются в URL, а отправляются «внутри» запроса.

◆ Отличие от query string:

- Query string → параметры видны в URL (/search?q=matrix).
- Request body → данные передаются в теле, скрыты от адресной строки.

◇ Когда используется:

- POST — создание ресурса (регистрация, добавление записи).
- PUT — полное обновление ресурса.
- PATCH — частичное обновление.
- DELETE — иногда поддерживает тело

◇ Основные форматы тела запроса

1. application/x-www-form-urlencoded

Классические формы (HTML).

Данные кодируются как ключ=значение&ключ2=значение2.

Когда пользователь заполняет поля **<input>** и нажимает **Submit**, браузер кодирует данные в формате:

ключ1=значение1&ключ2=значение2

Пример формы:

```
<form action="/login" method="post">
  <input type="text" name="username">
  <input type="password" name="password">
  <button type="submit">Войти</button>
</form>
```

curl:

curl -k -X POST -d "username=neo&password=trinity" https://httpbin.org/post

Hoppscotch

Метод: POST

URL: https://httpbingo.org/post

Body → x-www-form-urlencoded:

- username = neo
- password = trinity

2. multipart/form-data

Используется для загрузки файлов + текстовых данных.

curl:

curl -k -X POST -d "username=neo&password=trinity" https://httpbin.org/post

Hoppscotch

Метод: POST

URL: https://httpbin.org/post

Вкладка Body → Form-Data

Поле: username = neo

Файл: file = выберите файл

3. application/json

📌 Самый популярный формат для REST API.

Hoppscotch

Метод: POST

URL: https://jsonplaceholder.typicode.com/posts

Вкладка Body → JSON

```
{
  "title": "Matrix",
  "body": "There is no spoon",
  "userId": 1
}
```

◇ Итог:

- Query string подходит для поиска и фильтров.
- Request body нужен для форм, JSON и файлов.
- Тип определяется заголовком Content-Type.

3. Сравнение GET и POST

Характеристика	GET	POST
Где данные	В URL (query string)	В теле запроса
Объём данных	Ограничен длиной URL	Практически без ограничений
Видимость	Видно в адресной строке	Скрыто (но доступно в сетевом трафике)
Кэширование	Поддерживается	Обычно нет
Основное назначение	Получение (чтение) данных	Отправка, создание, изменение
Пример использования	Поиск, фильтр	Регистрация, загрузка файлов

GET → для безопасного «чтения» (без изменения данных).

POST → для «записи» и передачи чувствительных данных.

4. Кодировка параметров.

◆ Зачем нужна кодировка

В URL могут использоваться только ограниченные символы (латиница, цифры и некоторые спецсимволы - _ . ~).

Пробелы, кириллица, спецсимволы (&, ?, =, /) должны кодироваться, чтобы браузер и сервер правильно поняли запрос.

◆ URL-encoding (percent-encoding)

Каждый «запрещённый» символ заменяется на %XX, где XX — это его код в UTF-8 (в шестнадцатеричном виде).

Примеры:

hello world → hello%20world

нео → %D0%BD%D0%B5%D0%BE

a&b=c → a%26b%3Dc

◇ Пример с query string

GET /search?query=hello%20world&city=%D0%97%D0%B8%D0%BE%D0%BD HTTP/1.1

Host: example.com

hello world → hello%20world

Зион (кириллица) закодирована в UTF-8.

◇ Кодировка в HTML-формах

При отправке формы с application/x-www-form-urlencoded:

- пробелы заменяются на **+** (а не на **%20**),
- остальные символы кодируются так же, как в URL-encoding.

Пример:

POST /login HTTP/1.1

Content-Type: application/x-www-form-urlencoded

username=neo+smith&city=%D0%97%D0%B8%D0%BE%D0%BD

🔷 Важно помнить

- Браузер обычно автоматически кодирует параметры формы и URL.
- Разработчику важно знать, как именно будет выглядеть «сырой» запрос, чтобы понимать поведение серверной части.
- Ошибки в кодировке → неправильное отображение кириллицы или спецсимволов.

👉 Итог:

- URL-encoding нужен для безопасной передачи данных в URL.
- В формах пробелы превращаются в +.
- Лучше всегда тестировать запросы с кириллицей и спецсимволами (например, через curl или Hoppscotch).

Итоги лекции:

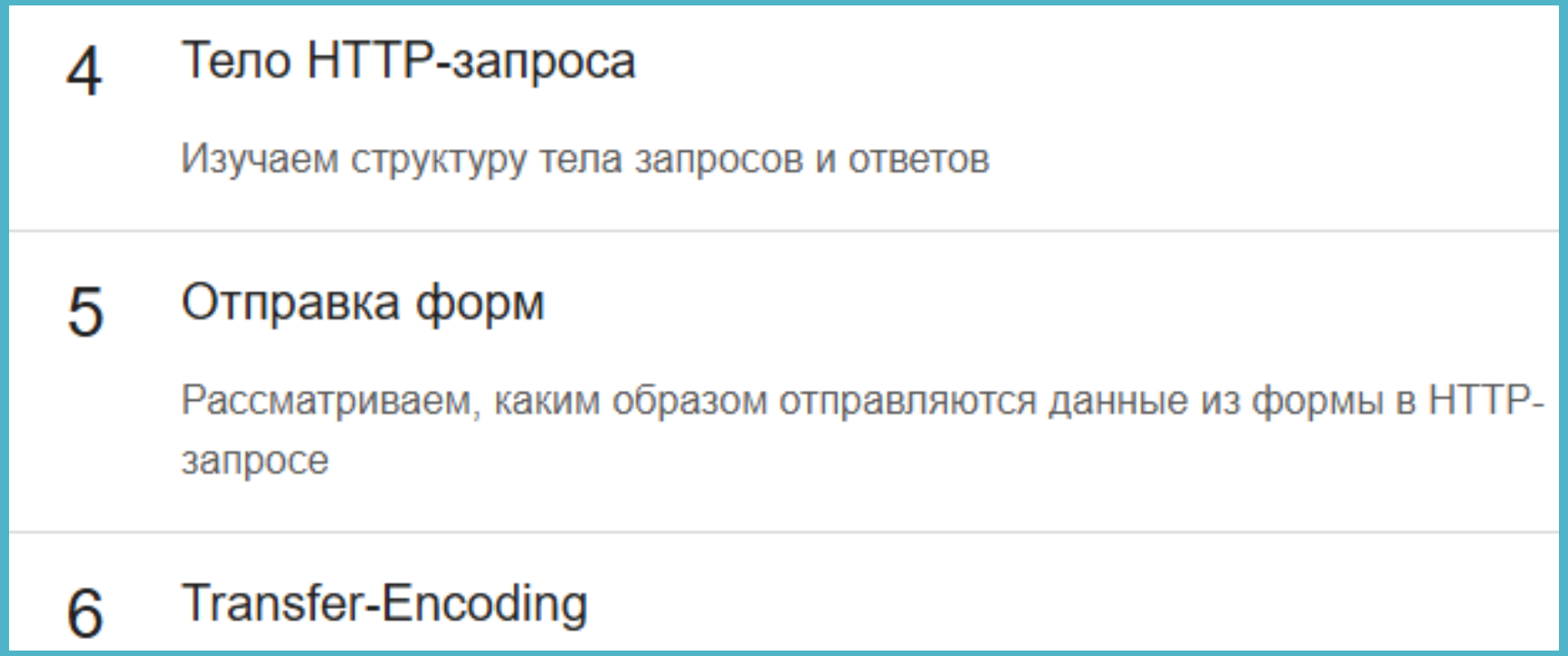
- Query string: параметры в URL (?q=matrix&page=2), удобно для поиска и фильтрации, но видно в адресе и ограничено по длине.
- Тело запроса: используется в POST/PUT/PATCH, данные не видны в URL.
- Форматы тела:
  - x-www-form-urlencoded → обычные формы,
  - multipart/form-data → загрузка файлов,
  - application/json → API-запросы.
- Кодировка: спецсимволы и кириллица кодируются (пробел → %20 или +, «имя» → %D0%B8%D0%BC%D1%8F).
- GET vs POST:
  - GET → чтение, кэшируется, параметры в URL.
  - POST → отправка/создание, данные в теле.
- Безопасность: всегда HTTPS, не передавать пароли в URL, использовать токены или защищённые cookie.

Контрольные вопросы:

- Что такое query string? Как он выглядит в URL?
- В чём разница между передачей данных через query string и через тело запроса?
- Какой метод HTTP обычно используется для передачи query string?
- Какие методы HTTP передают данные в теле запроса?
- Чем отличаются методы GET и POST при передаче данных?
- В каких случаях рекомендуется использовать GET, а в каких — POST?
- Какой формат кодирования данных используется по умолчанию в HTML-формах?
- Для чего применяется multipart/form-data?
- Как передаётся JSON в теле запроса?
- Что означает URL-encoding? Приведите пример преобразования.
- Как кодируются пробелы и кириллица в query string?

Домашнее задание:

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



Материалы лекций:

https://github.com/ShViktor72/Education2025

Обратная связь:

colledge20education23@gmail.com