ПМ3 Разработка модулей ПО.

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

Тема 1. Протокол HTTP .

Лекция 4. 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. Сравнение GET и POST.
2. Кодировка параметров.

2. 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.

3. 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:

- 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