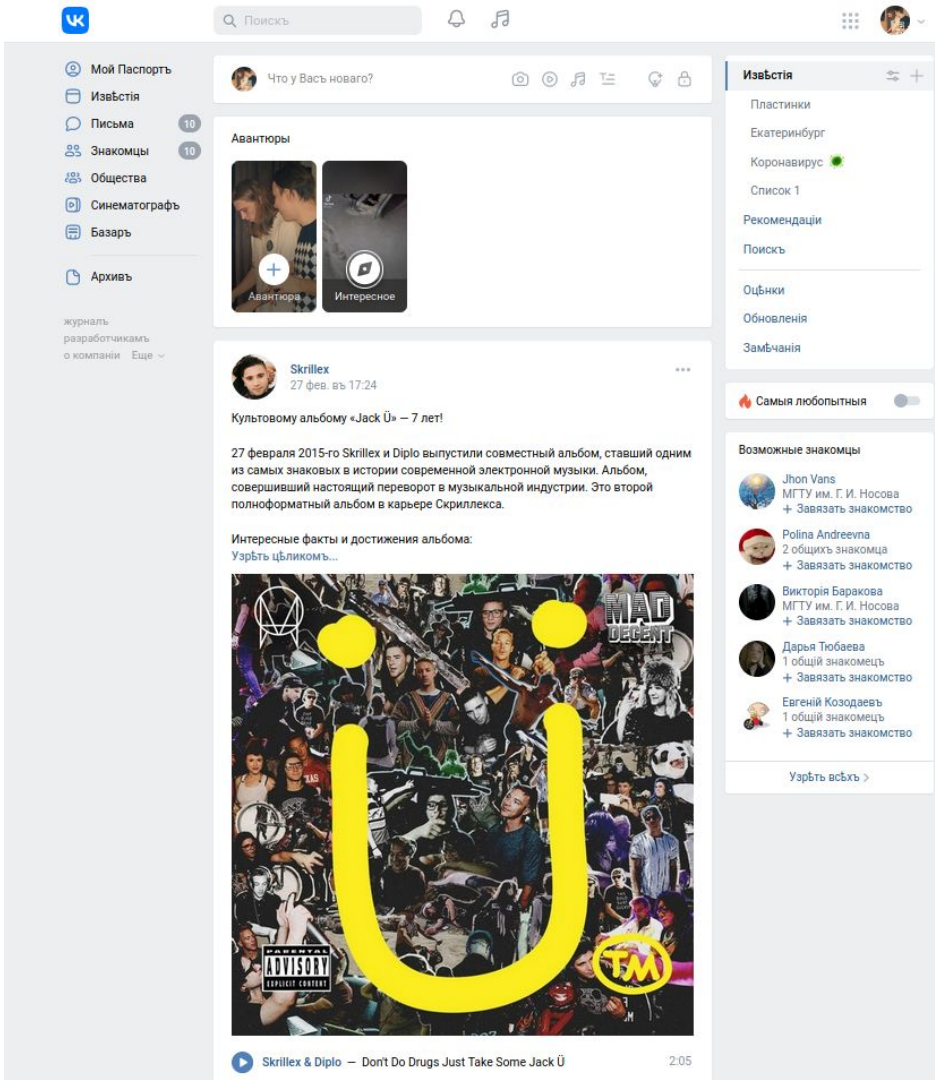
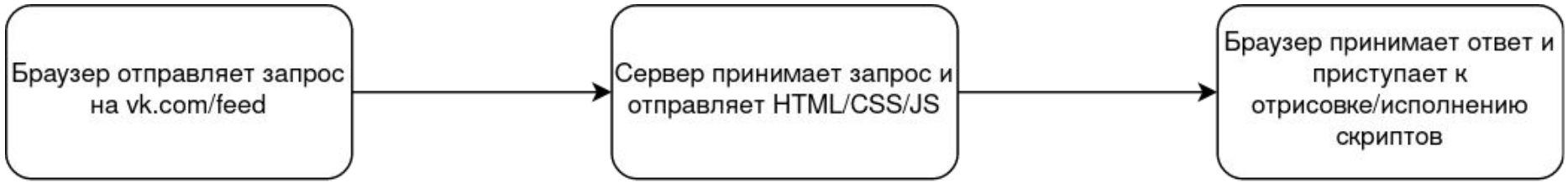
Spring framework
или web на Java

Что происходит когда мы заходим на сайт?



Последовательность действий при входе на страничку



Роль браузера

Браузер, или веб-обозреватель — прикладное программное обеспечение для просмотра страниц, содержания веб-документов, компьютерных файлов и их каталогов; управления веб-приложениями; а также для решения других задач.

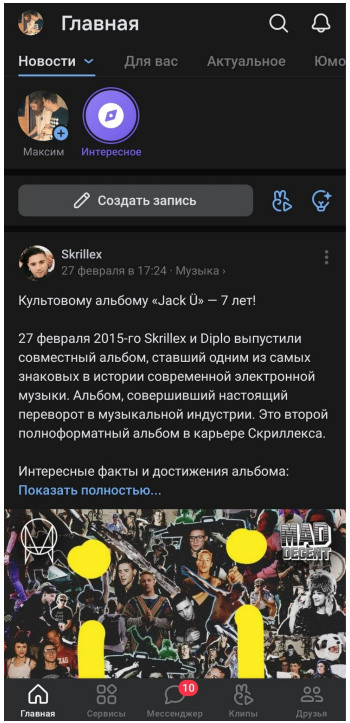
Но если кратко, то:

- Обмен сообщениями по HTTP протоколу
- Рендеринг HTML/CSS
- Выполнение скриптов
- Дополнительные функции*

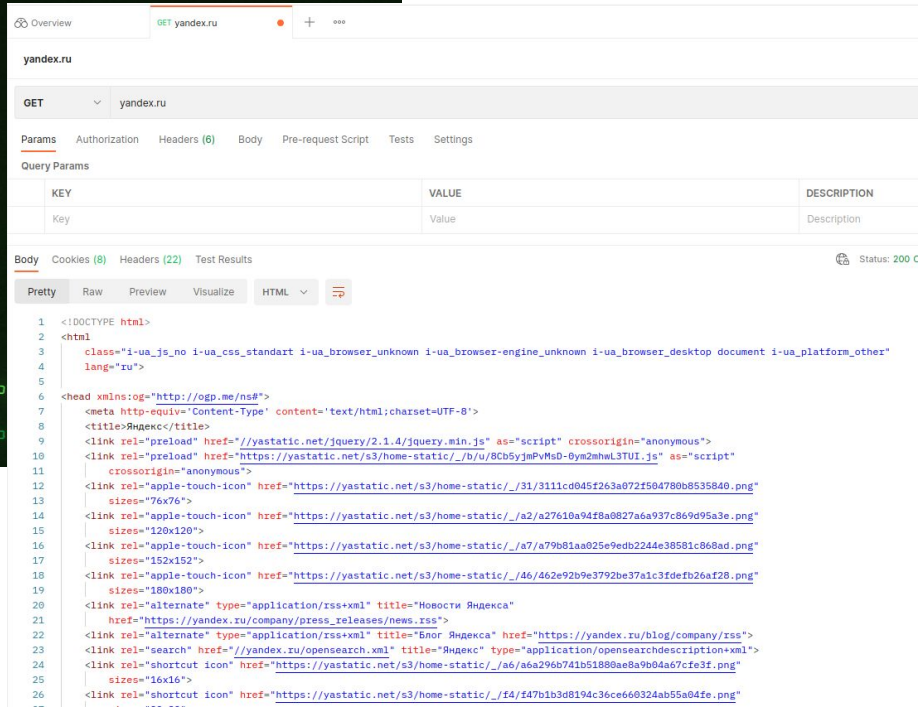
* - Вкладки, расширения, история, настройки и т.д.

Кто может общаться с сервером?

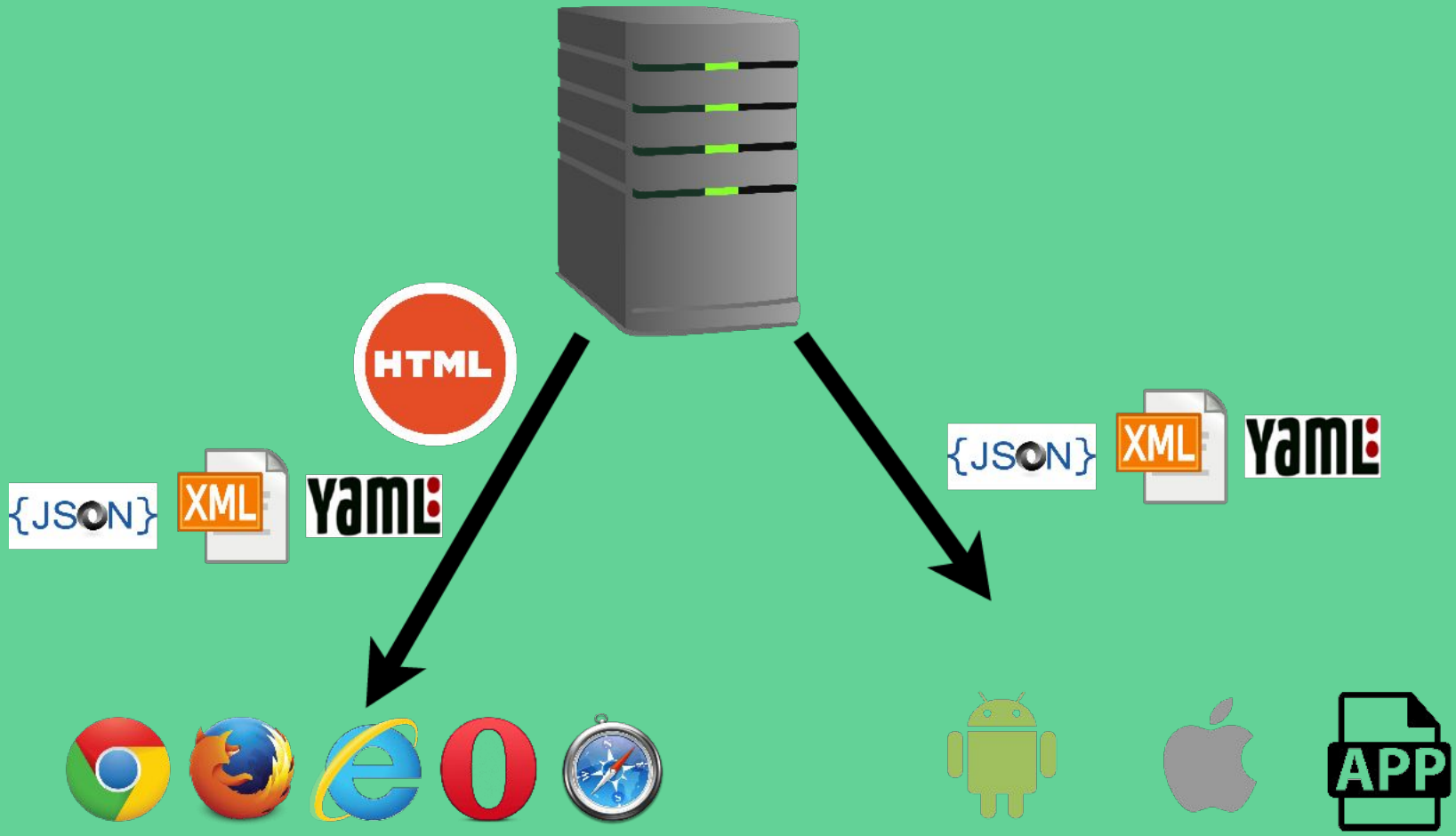
Любое приложение может посылать запросы




```
maxim@Asus:~$ http localhost:8080
HTTP/1.1 200
Connection: keep-alive
Content-Language: en-US
Content-Type: text/html;charset=UTF-8
Date: Wed, 02 Mar 2022 21:31:45 GMT
Keep-Alive: timeout=60
Transfer-Encoding: chunked

<!DOCTYPE HTML>
<html>
<link href="/style/style.css" rel="stylesheet">

<head>
</head>

<body>
    <div class="mainContainer">
        <img src="/images/logo.gif">
        <div style="margin-bottom: 10px;">
            ПриВет, сегодня <b>03.03</b>. А это о
        </div>
        <input type="file" id="imageLoader" accep
        <div class="imagesView">
```



Данные


HTML
{JSON}
XML
YAML
{JSON}
XML
YAML
APP

```
<body>
    <div class="mainContainer">
        <img th:src="@{/images/logo.gif}">
        <div style="margin-bottom: 10px;">
            Привет, сегодня <b th:text="${time}"></b>. А это отличное время, чтобы сделать
        </div>
        <input type="file" id="imageLoader" accept="image/png, image/jpeg">
        <div class="imagesView">
            <div class="imageContainer">
                <img class="imageInContainer" id="loadedImage">
            </div>
            <div id="spinner">
                <div id="spinner-item"></div>
            </div>
            <div class="imageContainer" id="processedImageContainer">
                <img class="imageInContainer" id="processedImage">
            </div>
        </div>

        <div class="settingsContainer">
            <select name="effect" id="effectSelector" style="width: 100%;">
                <option value="" selected disabled hidden>Выбирете эффект</option>
                <option value="SeamCarving">Seam carving</option>
                <option value="Monochrome">Monochrome</option>
            </select>
            <div>
                <p id="widthTextVal">Ширина изображения: </p>
                <input type="range" max="0" name="Width" id="widthRange">
            </div>
            <div>
                <p id="heightTextVal">Высота изображения:</p>
                <input type="range" max="0" name="Width" id="heightRange">
            </div>
            <div>
                <span>Сохранить размер изображения:</span>
                <input type="checkbox" id="saveImageScale">
            </div>
            <button name="Start" id="submitButton" disabled>Обработать изображение</button>
        </div>
    </div>
    <script th:src="@{/script/script.js}"></script>
```

HTML

Теги

Позволяют задать способ отображения контента на веб-страницах.

Атрибуты

Используется для определения характеристик html-элемента и помещается внутри открытого тега элемента.

Содержимое

Обычно представляет собой отображаемое тегом содержимое (текст, внутренние теги и т.д.).

CSS

Свойства

Определяют графические характеристики элемента.

Селекторы

Определяют элементы, которым будут применены свойства.

```css
html,
body {
    width: 100%;
    height: 100%;
    margin: 0;
}

.mainContainer {
    display: flex;
    align-items: center;
    height: 100%;
    width: 100%;
    flex-direction: column;
}

.imagesView {
    display: flex;
    max-width: 70%;
    max-height: 50%;
    justify-content: center;
    margin: 20px 0;
}

.imageContainer {
    margin: 0 10px;
    max-width: 50%;
    max-height: 100%;
}

.imageInContainer {
    width: 100%;
    height: 100%;
    object-fit: contain;
}

.imagePlaceHolder {
    width: 40%;
    margin: 0 10px;
    border-width: 3px;
    border-style: dotted;
    border-color: black;
}
```

```
let loadedImage;
init();

function init() {
    let imageLoader = document.getElementById('imageLoader');
    loadedImage = document.getElementById('loadedImage');

    document.getElementById('submitButton').addEventListener('click', () => {
        let postSended = sendPost(loadedImage.src);
        if (postSended)
            document.getElementById('spinner').style.display = 'flex';
    });

    imageLoader.addEventListener('change', event => {
        const files = event.target.files;
        const file = files[0];
        loadAndShowImage(file);
    });

    let widthRange = document.getElementById('widthRange');
    widthRange.addEventListener('input', () => {
        document.getElementById('widthTextVal').innerText = 'Ширина изображения:' +
    });

    let heightRange = document.getElementById('heightRange');
    heightRange.addEventListener('input', () => {
        document.getElementById('heightTextVal').innerText = 'Высота изображения:'
    });
}

function loadAndShowImage(file) {
    const reader = new FileReader();
    reader.readAsDataURL(file);
    reader.addEventListener('load', (event) => {
        loadedImage.src = event.target.result;
        loadedImage.style.display = 'block';
        loadedImage.addEventListener('load', () => {
            document.getElementById('submitButton').disabled = false;
            let wRange = document.getElementById('widthRange');
            let hRange = document.getElementById('heightRange');
            wRange.max = Math.ceil(loadedImage.naturalWidth);
            wRange.min = Math.ceil(loadedImage.naturalWidth * 0.1);
            wRange.value = Math.ceil(loadedImage.naturalWidth / 2);
            wRange.dispatchEvent(new Event('input'));
            hRange.max = Math.ceil(loadedImage.naturalHeight);
```

Java Script

Это

Мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Является реализацией спецификации ECMAScript.

Цель

Применяется для выполнения сценариев для придания интерактивности веб-страницам

Что умеет?

- Добавлять новый HTML-код на страницу, изменять существующее содержимое, модифицировать стили.
- Реагировать на действия пользователя, щелчки мыши, перемещения указателя, нажатия клавиш.
- Отправлять сетевые запросы на удалённые сервера, скачивать и загружать файлы

Чего не умеет?

- Не имеет прямого доступа к системным функциям ОС
- Различные окна/вкладки не знают друг о друге

JSON (*JavaScript Object Notation*)

Это

Текстовый формат обмена данными, основанный на JavaScript. Но при этом формат независим от JS и может использоваться в любом языке программирования.

Описывает

- Объекты
- Массивы
- Примитивные типы
  - Строка
  - Число
  - Логическое значение
  - Null

```
{
    "orderID": 12345,
    "shopperName": "Ваня Иванов",
    "shopperEmail": "ivanov@example.com",
    "contents": [
        {
            "productID": 34,
            "productName": "Супер товар",
            "quantity": 1
        },
        {
            "productID": 56,
            "productName": "Чудо товар",
            "quantity": 3
        }
    ],
    "orderCompleted": true
}
```

```
function getData() {
    return {
        orderID: 12345,
        shopperName: "Ваня Иванов",
        shopperEmail: "ivanov@example.com",
        contents: [{
                "productID": 34,
                "productName": "Супер товар",
                "quantity": 1
            },
            {
                productID: 56,
                productName: "Чудо товар",
                quantity: 3
            }
        ],
        orderCompleted: true
    }
}
```

HTTP

Запрос

```
GET / HTTP/1.1
Accept: text/html
Accept-Charset: utf-8
Host: CoolSite.com
```

Ответ

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Content-Length: 98

<html>
  <head>
    <title>An Example Page</title>
  </head>
  <body>
    <p>Hello World</p>
  </body>
</html>
```

```
POST /user/create HTTP/1.1
Accept: application/json
Content-Type: application/json
Content-Length: 28
Accept-Charset: utf-8
Host: SomeWebSite.ru

{
  "Id": 12345,
  "User": "John"
}
```

Java servlet API

```
public class MyServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException {
        if (req.getRequestURI().equals("/")) {
            resp.setContentType("text/html");
            resp.getWriter().print("
                  <html>
                        <head>
                        </head>
                        <body>
                              <h1>Welcome!</h1>
                              <p>This is a very cool page!</p>
                        </body>
                  </html>
                  ");
        }
        else {
            throw new IllegalStateException("Help, I don't know what to do with this url");
        }
    }
}
```

```java
public class TomcatApplicationLauncher {

    public static void main(String[] args) throws LifecycleException {
        Tomcat tomcat = new Tomcat();
        tomcat.setPort(8080);
        tomcat.getConnector();

        Context ctx = tomcat.addContext("", null);
        Wrapper servlet = Tomcat.addServlet(ctx, "myServlet", new MyServlet());
        servlet.setLoadOnStartup(1);
        servlet.addMapping("/*");

        tomcat.start();
    }
}
```

```
http://CoolWebsite.com/api/users/{userId}
public class MyServlet extends HttpServlet {

   @Override
   protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException {
       if (req.getRequestURI().equals("/")) {
           resp.setContentType("text/html");
           resp.getWriter().print("<html><head></head><body><h1>Welcome!</h1><p>This is a very cool
page!</p></body></html>");
       } else if (req.getRequestURI().startsWith("/api/users/")) {

           Integer userId = Integer.valueOf(req.getRequestURI().lastIndexOf("/") + 1);

           resp.setContentType("application/json");

           User user = Database.findUser(userId);

           resp.getWriter().print("{\n" +
                   "  \"id\":" + userId + ",\n" +
                   "  \"age\": \"" + user.age + "\",\n" +
                   "  \"name\" : \"" + user.age + "\"\n"  +
                   "}");
           //resp.getWriter().print(user.toString());
       } else {
           throw new IllegalStateException("Help, I don't know what to do with this url");
       }
   }
}
```

Как это работает в Spring’е

```java
@SpringBootApplication
@EnableWebMvc
public class DemoApplication extends WebMvcAutoConfiguration {
   public static void main(String[] args) {
       SpringApplication.run(DemoApplication.class, args);
   }
}
```

```java
@Controller
public class WebController {
    @GetMapping("/")
    public String mainPage(Model model) {
        model.addAttribute("time", new SimpleDateFormat("dd.MM").format(System.currentTimeMillis()));
        return "index.html";
    }
}
```

```java
@Controller
public class WebController {
    @GetMapping("/api/user/{someID}")
    public User getUser(@PathVariable(value="someID") String id) {
        return Database.getUserById(id);
    }
}
```