Thursday, 11 June 2020

# Изпитна тема No 17: Интернет програмиране

План-тезис: Създаване на REST API. Извикване на REST заявки с JavaScript и AJAX. Внедряване на проект (deployment).

---

## REST API

- **REST** (***Representational State Transfer***) е стил софтуерна архитектура за реализация на уеб услуги. Архитектурният модел REST включва взаимодействията между сървър и клиент, осъществени по време на трансфера на данни.

---

## RESTful API

- RESTful уеб API е уеб приложение, което използва принципите на HTTP и REST. Представлява колекция от ресурси с четири дефинирани аспекта:
  - Основният „URL“ за уеб приложенията като http://example.com/blabla
  - Internet media типът на данните поддържани от уеб приложенията. Това най-често е JSON, но може да бъде всеки друг валиден Интернет медиен тип.
  - Операции поддържани от уеб приложението използвайки HTTP метиди (GET, PUT, POST, или DELETE).
  - Приложенията трябва да се задвижват от хипертекст.

---

## JSON

- JSON е много често използван формат на данни, използван в уеб комуникацията
  - Основно в комуникация браузър-сървър или сървър-сървър
  - Официалният тип интернет медия (MIME) за JSON е application/json
  - JSON файловете имат разширение .json

- JSON обикновено се използва като заместител на XML в AJAX
  - По-кратък и лесен за разбиране
  - По-бърз за четене и писане и е по-интуитивен
  - Не поддържа схеми и пространства от имена

```
{
    "firstName": "Pesho",
    "courses": ["C#", "JS", "ASP.NET"]
    "age": 23,
    "hasDriverLicense": true
}
```

---

## XML

- XML дефинира набор от правила за кодиране на документи
  - Идва от Extensible Markup Language
  - Подобен на JSON
    - По отношение на читаемостта от човека и обработката от машини
    - По отношение на йерархия (стойности в стойности)
- XML е текстов формат
  - Силна поддръжка за различни човешки езици чрез Unicode
  - Дизайнът се фокусира силно върху действителните документи

```
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

- Има 2 типа MIME за XML - application/xml и text/xml
- .xml разширение
- Има много приложения:

- Широко използван в SOA
- Конфигуриране на .NET приложения
- Използва се във формати на Microsoft Office
- XHTML е трябвало да бъде строг HTML формат

---

## Web API

- Web API е интерфейс за програмиране на приложения
  - Използван от Web Browser (SPA), Mobile Applications, Games, Desktop Applications, Web Server
- Състои се от публично изложени крайни точки (endpoint-и)
  - Крайните точки съответстват на дефинирана система за заявка-отговор
  - Комуникацията обикновено се изразява във формат JSON или XML
  - Комуникацията обикновено се осъществява чрез уеб протокол
    - Най-често HTTP - чрез уеб сървър, базиран на HTTP

### ASP.NET Core Web API (Return Types)

- ASP.NET Core предлага няколко опции за типове връщане на API Endpoint
  - Специфичен тип
    - Най-простият тип действие
  - IActionResult тип
    - Подходящо, когато са възможни няколко типа ActionResult в съответното действие
- Препоръчва се използването на ActionResult <T>

```
[HttpGet]
public IEnumerable<Product> Get()
{
    return
this.productService.GetAllProducts();
}

[HttpGet("{id}")]
[ProducesResponseType(200, Type =
typeof(Product))]
[ProducesResponseType(404)]
public IActionResult GetById(int id)
{
    var product =
this.productService.GetById(id);

    if (product == null) return
```

**ASP.NET Core Web API (GET Методи)**

```
[HttpGet]
public ActionResult<IEnumerable<Product>> GetProducts()
  => this.productService.GetAllProducts();

[HttpGet("{id}")]
public ActionResult<Product> GetProduct(long id)
{
    var product = this.productService.GetById(id);
    if (product == null) return NotFound();
    return product;
```

**ASP.NET Core Web API (POST Методи)**

```
[HttpPost]
public ActionResult<Product> PostProduct(ProductBindingModel
pm)
{
    this.productService.RegisterProduct(pm);

    return CreatedAtAction("GetProduct", new { id = pm.Id },
pm);
}
```

**ASP.NET Core Web API (PUT Методи)**

```
[HttpPut("{id}")]
public IActionResult PutProduct(long id, ProductBindingModel
pm)
{
    if (id != pm.Id) return BadRequest();
    this.productService.EditProduct(id, pm);
    return NoContent();
}
```

**ASP.NET Core Web API (DELETE Методи)**

```
[HttpDelete("{id}")]
public ActionResult<Product> DeleteProduct(long id)
{
    var product = this.productService.DeleteProduct(id);
    if (product == null) return NotFound();
    return product;
}
```

---

## JavaScript

- JavaScript (JS) е скриптов език
  - Изпълняват се команди (скрипт)
    - Не се компилира
  - Може да работи в интерактивен режим
- Наред с HTML и CSS, JavaScript е една от 3-те основни технологии в уеб света
  - JavaScript позволява динамичност и интерактивност в уеб страниците
    - Има достъп до DOM и API(известия, геолокация, …)

---

## AJAX

- Зареждане на заден план на динамично съдържание / данни
- Зарежда данни от уеб сървъра и ги визуализира
- Два вида AJAX
  - Частично изобразяване на страница
    - Зарежда HTML фрагмент + показва го в 
  - JSON услуга
    - Зарежда JSON обект и го показва с JS / jQuery

## API for AJAX

```
<button onclick="loadRepos()">Load Repos</button>
<div id="res"></div>
```

```
function loadRepos() {
  const req = new XMLHttpRequest();
  req.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200)
      document.getElementById("res").textContent =
        this.responseText;
  };
  req.open("GET",
    "https://api.github.com/users/testnakov/repos", true);
  req.send();
}
```

## Fetch API

- Методът `fetch()` позволява извикване на уеб заявки
- Той е подобен на XMLHttpRequest (XHR). Основната разлика е, че Fetch API:
  - Използва обещания (Promises)
  - Е по-прост и чист API

```
fetch('./api/some.json')
  .then(function(response) {…})
  .catch(function(err) {…})
```

- Отговорът на заявката от fetch() е Stream обект
- Четенето на потока се случва асинхронно
- Когато се извика методът json(), се връща обещание
  - Статуса на отговора се проверява (трябва да е 200), преди да се преобрази отговора в JSON

- GET Заявка

```
fetch('https://swapi.co/api/people/4')
  .then((response) => response.json())
  .then((data) => console.log(JSON.stringify(data)))
  .catch((error) => console.error(error))
```

- POST Заявка

```
fetch('/url', {
  method: 'post',
  headers: { 'Content-type': 'application/json' },
  body: JSON.stringify(data),
})
```

- Методи на Тялото :D
  - clone() - създаване на копие на отговора
  - json() - разрешава обещания с JSON
  - redirect() - създаване на ново обещание, но с различен URL адрес
  - text() - решава обещанието със низ