Razvoj Web Aplikacija

Predavanje 13

Danas

- AJAX tehnika
- JSON format podataka
- jQuery i AJAX
- Alat LibMan

AJAX tehnika

AJAX

- **AJAX** – engl. *Asynchronous JavaScript and XML*
- Predstavlja tehniku koja JavaScript-u na klijentu omogućava komunikaciju sa serverom u cilju dohvaćanja ili slanja podataka
- Glavna komponenta koja omogućuje AJAX je **`XMLHttpRequest`** čija je izravna upotreba kompleksna, no korištenje kroz dostupne biblioteke to uvelike pojednostavljuje
- ***Asynchronous*** – prikaz stranice se izvršava u drugoj niti (dretvi), drugim riječima dok se zahtjev izvršava stranica ne "blokira" (ili ne bi trebala, ovisi o ispravnoj implementaciji)
- ***JavaScript*** – jezik koji upravlja ovom tehnikom je JavaScript
- ***XML*** – jezik koji opisuje podatke koji se prenose je ~~najčešće~~ ~~nekada~~ rijetko XML, u većini slučajeva je to JSON ili HTML

AJAX

- Mogućnosti korištenja AJAX tehnike na klijentskoj strani:
  - **XMLHttpRequest (XHR)** – najniža razina, prilično kompleksan kod
  - **jQuery** – nešto viša razina, jednostavniji kod, asinkronost se najčešće postiže "callback" mehanizmom
  - **Axios** – visoka razina, jednostavan kod, asinkronost se postiže async/await mehanizmom
  - **Fetch API** – visoka razina, jednostavan kod, asinkronost se postiže async/await mehanizmom, ugrađeno u pretraživač
- Na serverskoj strani je potrebno osigurati adrese s kojima će klijent asinkrono komunicirati
  - U Web API kontroleru nam treba akcija koja vraća JSON
  - U MVC kontroleru nam treba akcija koja vraća HTML (iz view-a) ili JSON

AJAX – XHR primjer

```
// XHR
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function () {
    if (this.readyState == 4 && this.status == 200) {
        var res = xhttp.responseText;
    }
};
xhttp.open("GET", "http://localhost:1234/test/index", true);
xhttp.send();
```

AJAX – jQuery vs Axios vs Fetch API

```
// jQuery
$.ajax({
    url: "http://localhost:1234/test/index",
    success: function (res) {
        var data = res;
    }
});

// Axios
const axios = require('axios');
const res = await axios.get("http://localhost:1234/test/index");
const data = res.data;

// Fetch API
const res = await fetch("http://localhost:1234/test/index");
const data = await res.json();
```

JSON format podataka

Formati podataka

- Podaci koji se u asinkronom pozivu s klijenta šalju serveru i podaci koje server vraća moraju biti formatirani na način razumljiv i klijentu i poslužitelju
- Klijent podatke šalje često kao dio rute (npr. id), putem query stringa ili u tijelu HTTP zahtjeva, a može ih poslati i u zaglavlju zahtjeva
- Najčešći formati u kojima server vraća podatke su:
  - HTML format
  - JSON format
- Manje često formati su
  - tekstualni format
  - XML format (AJAX – kratica zavarava)
- Klijent i poslužitelj moraju biti u dogovoru i znati što koji od njih može očekivati od druge strane

JSON

- **`JSON`** – *JavaScript Object Notation* je tekstualni format za izmjenu podataka
- Karakteristike:
  - objekt je zatvoren unutar **`{ }`** zagrada
  - sastoji se od parova **`naziv:vrijednost`**
  - parovi su odvojeni zarezom
  - vrijednost može biti string u dvostrukim navodnicima, broj (točka je decimalni separator), true/false, null, polje ili sam JSON objekt
    - polje je zatvoreno u [ ] zagradama i sastoji se od niza vrijednosti odvojenih zarezima

JSON

- JSON je dizajniran tako da se JavaScript objekt na lak način može pretvoriti u string koji sadržava JSON i obratno

```
//Javascript

<script>
        function Osoba()
        {
            this.Ime = "Luka";
            this.Prezime = "Lukić";
            this.Telefoni = ["123456789", "0987654321"];
            this.Prebivaliste = {
                privatno: new Adresa("Ilica", 1),
                poslovno: new Adresa("Frankopanska", 50)
            }
        }
</script>
```

```
//JSON

{
  "Ime": "Luka",
  "Prezime": "Lukić",
  "Telefoni": [
    "123456789",
    "0987654321"
  ],
  "Prebivaliste": {
    "privatno": {
      "Ulica": "Ilica",
      "Broj": 1
    },
    "poslovno": {
      "Ulica": "Frankopanska",
      "Broj": 50
    }
  }
}
```

jQuery i AJAX

jQuery funkcije ajax(), get(), post()

- Služe za dohvaćanje podataka sa servera u bilo kojem formatu
- Ako u zahtjevu šaljemo podatke koji mijenjaju stanje na serveru (izvršavamo insert/modify/delete aktivnost), upotrebljavamo **`post()`**
- Ako zahtjev služi dohvaćanju podataka, preporuka je koristiti **`get()`**
- Funkciju **`ajax()`** možemo koristiti u oba slučaja
- Parametar **data** je opcionalan – predstavlja podatke koji se šalju serveru u sklopu zahtjeva. Obično su tipa string ili objektni literal
- Zadnji parametar (ili dva) je najčešće anonimna funkcija koja se poziva kad podaci stignu sa poslužitelja, a služi za obradu pristiglih podataka

```
// HTTP GET
$.get("http://localhost:1234/test", data, callback);

// HTTP POST
$.post("http://localhost:1234/test", data, callback);

// HTTP GET/POST
$.ajax({
    url: "http://localhost:1234/test",
    method: "GET",
    data: data,
    success: function (res) {
        // Code in case of success
    },
    error: function (res) {
        // Code in case of failure
    }
});
```

jQuery funkcije ajax(), get(), post()

- Ako se podatak šalje u obliku stringa on mora biti formatiran u query string formatu
  - podatak je u obliku **`naziv=vrijednost`**
  - podaci se odvajaju znakom **`&`**
  - podaci moraju biti URL kodirani kako ne bi bilo nedopuštenih znakova. U tu svrhu dobro je koristiti javascript funkciju **`encodeURIComponent`**() koja vraća URL kodiran string
- jQuery nudi metodu **`serialize()`** koja se poziva na formi i kao rezultat daje *query string* nastao na osnovi imena elemenata forme i korisnikovih unosa
- Ako se podatak šalje kao objektni literal ne treba URL kodirati podatke jer će jQuery funkcije prema potrebi napraviti kodiranje

jQuery funkcija ajax()

- Objedinjuje sve funkcije za rad s AJAX-om
- Prima samo jedan parametar, objektni literal koji sadržava opcije, a neke od najkorisnijih su:
  - **`url`** – adresa na koju šaljemo zahtjev
  - **`data`** – sadržava podatke koje šaljemo u zahtjevu (string ili objektni literal)
  - **`dataType`** – tip podataka koji se očekuje od poslužitelja (`xml, json, text`)
  - **`method`** – način slanja zahtjeva
    - `HTTP GET, HTTP POST, HTTP PUT, HTTP DELETE`
  - **`header`** – parametri u zaglavlju zahtjeva

jQuery funkcija ajax()

```
$.ajax({
    url: "http://localhost:1234/test",
    method: "GET",
    dataType: "json",
    data: {
        firstName: "Joe",
        email: "joe@example-mail.com"
    },
    success: function (res) {
        // Code in case of success
    },
    error: function (jqXhr, textStatus,
errorThrown) {
        // Code in case of failure
    }
});
```

```
$.ajax({
    url: "http://localhost:1234/test",
    method: "GET",
    dataType: "json",
    data: {
        firstName: "Joe",
        email: "joe@example-mail.com"
    }
})
.done(function (res) {
    // Code in case of success
})
.fail(function (jqXhr, textStatus,
errorThrown) {
    // Code in case of failure
})
```

Alat LibMan

LibMan

- U ASP.NET Core MVC projektima često se događa da nisu instalirane klijentske biblioteke, već su samo konfigurirane u datoteci **libman.js**
- **LibMan** alat se koristi da bi se upravljalo tim bibliotekama
  - LibMan mora biti instaliran
  - **libman restore** – instalira JS biblioteke u projekt
  - **libman clean** – deinstalira JS biblioteke iz projekta
- Alat koristi online servise za download biblioteka (provider):
  - Cdnjs,
  - Unpkg,
  - JsDelivr
  - Filesystem (lokalna mapa na računalu)
- Da bismo uređivali datoteku libman.js, najjednostavnije je koristiti **Add > Client Side Library…**
- Datoteku je jednostavno i ručno uređivati zbog jednostavne strukture
- Kada radite svoj projekt, možete koristiti upravo datoteku libman.js da biste kod arhiviranja projekta uštedili na prostoru
  - Osim što možete obrisati mape **.vs**, **obj** i **bin**, slobodno možete obrisati i mapu **wwwroot/lib**, ona uvijek može biti vraćena pomoću **libman restore**

LibMan

- DEMO

Hvala na pažnji!