REST i Web API

Representational State Transfer je arhitektonski stil za pisanje web servisa.
Definiše pravila pristupanja i upravljanja resursima servisa na sledeći način:

- Listanje korisnika: **GET /users**
- Dodavanje novog korisnika: **POST /users**
- Ažuriranje postojećeg korisnika: **PUT /users/123** (gde je 123 identifikator korisnika)
- Brisanje postojećeg korisnika: **DELETE /users/123** (gde je 123 identifikator korisnika)

RESTful web servis je web servis implementiran korišćenjem HTTP protokola i REST principa

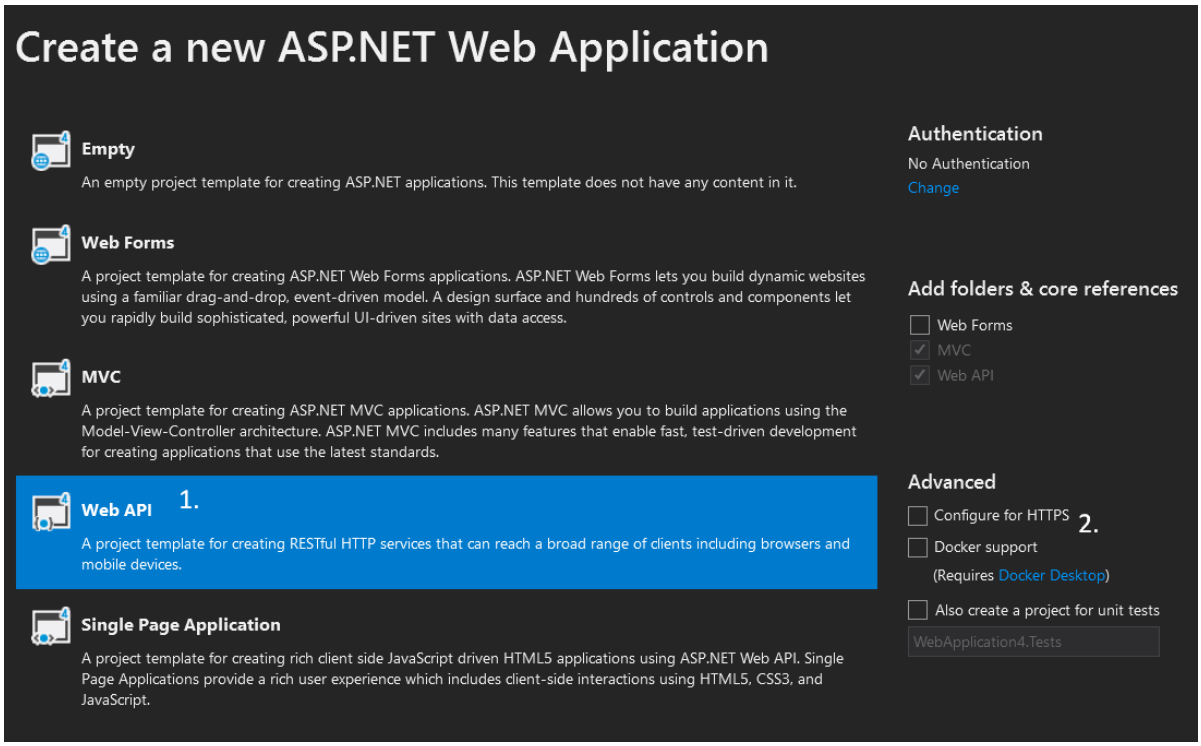
HTTP statusni kodovi:

- 1xx informativni odgovori
- 2xx uspešno (200 OK, 201 Created, 202 Accepted)
- 3xx redirekcija
- 4xx klijentske greške (400 Bad Request, 401 Unauthorized, 403 Forbidden)
- 5xx serverske greške (500 Internal Server Error)

ASP .NET Web API predstavlja idealnu platformu za kreiranje RESTful aplikacija na .NET razvojnom okviru

Kreiranje ASP .NET Web API projekta

Postupak kreiranja ASP .NET Web API projekta je sličan kao što smo kreirali prazan projekat. Jedina razlika je u poslednjem prozoru na **slici 1.**



ASP .NET Web API - folderi:

- Web API konfiguracija je smeštena u **App_Start/WebApiConfig.cs.**

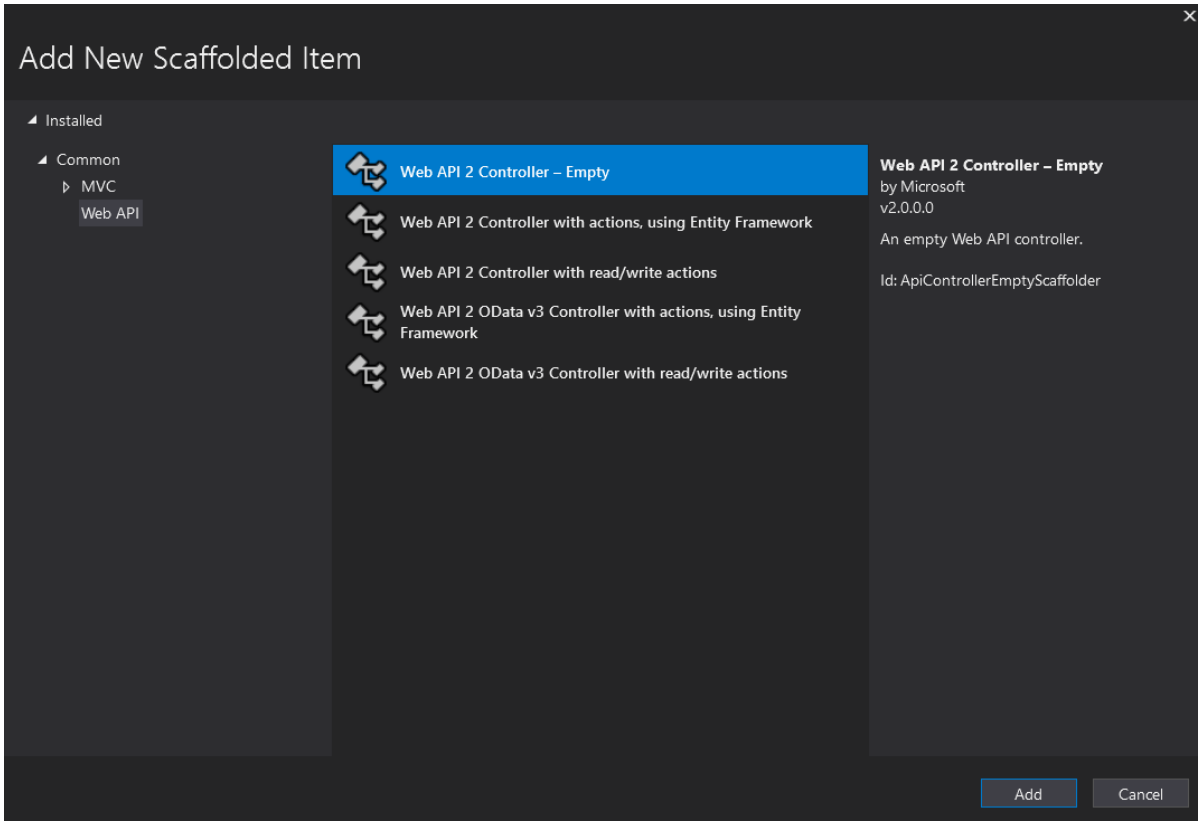
- Globalna konfiguraciona klasa je **Global.asax.** U **Application_Start()** je izvršeno konfigurisanje Web API putanja (routes).
- Web API se konfiguriše pomoću code based konfiguracije uz pomoć GlobalConfiguration klase.
- **Configure()** metoda zahteva callback metodu gde smo konfigurisali naš Web API.

ASP .NET Web API - rutiranje

- Slično ASP.NET MVC rutiranju. Povezuje dolazeći HTTP zahtev sa odgovarajućom akcionom metodom kontrolera.

Kreiranje ASP .NET Web API kontrolera

Kontroler obradjuje dolazeći HTTP zahtev i vraća odgovor korisniku
Ime kontrolera mora da se završava sa **Controller** i mora biti naslednik **System.Web.Http.ApiController** klase
Sve public metode kontrolera se zovu **akcione metode** (action methods).
Ime akcione metode može biti isto kao i ime HTTP metode, ili je dovoljno da akciona metoda počinje imenom HTTP metode sa bilo kojim nastavkom.



Da biste mogli da implementirate osnovne CRUD operacije u ovom kontroleru neophodno je da dodate novi Model, **klasu User.cs**, u direktorijum Models

In [ ]:
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace WebAPI.Models
{
    public class User
    {
```

```
        public int Id { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }

    }
}
```

Takodje, potrebno je da kreirate klasu **Users** koja sadrži listu svih korisnika

In [ ]:
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace WebAPI.Models
{
    public class Users
    {
        public static List<User> UsersList { get; set; } = new List<User>()
        {
            new User() { Id = GenerateId(), FirstName = "Pera", LastName = "Peric" }
            new User() { Id = GenerateId(), FirstName = "Sima", LastName = "Simic" }
            new User() { Id = GenerateId(), FirstName = "Misa", LastName = "Misic" }
        };

        public static User FindById(int id)
        {
            return UsersList.Find(item => item.Id == id);
        }

        public static User AddUser(User user)
        {
            user.Id = GenerateId();
            UsersList.Add(user);
            return user;
        }

        public static void RemoveUser(User user)
        {
            UsersList.Remove(user);
        }

        public static User UpdateUser(User user)
        {
            User existingUser = FindById(user.Id);
            existingUser.FirstName = user.FirstName;
            existingUser.LastName = user.LastName;

            return existingUser;
        }

        private static int GenerateId()
        {
            return Math.Abs(Guid.NewGuid().GetHashCode());
        }
    }
}
```

Dodati sledeći kod u klasu **UsersController**:

In [ ]:
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
```

```
using System.Net.Http;
using System.Web.Http;
using WebAPI.Models;

namespace WebAPI.Controllers
{
    public class UsersController : ApiController
    {

        public List<User> Get()
        {
            return Users.UsersList;
        }

        public User Get(int id)
        {
            return Users.FindById(id);
        }

        public IHttpActionResult Post(User user)
        {
            if (user == null)
            {
                return BadRequest();
            }
            if (user.FirstName == null || user.FirstName.Length == 0)
            {
                return BadRequest();
            }
            if (user.LastName == null || user.LastName.Length == 0)
            {
                return BadRequest();
            }
            return Ok(Users.AddUser(user));
        }


        public IHttpActionResult Put(User user)
        {
            if (user == null)
            {
                return BadRequest();
            }
            if (user.FirstName == null || user.FirstName.Length == 0)
            {
                return BadRequest();
            }
            if (user.LastName == null || user.LastName.Length == 0)
            {
                return BadRequest();
            }
            if(Users.FindById(user.Id) == null)
            {
                return BadRequest();
            }
            return Ok(Users.UpdateUser(user));
        }

        public IHttpActionResult Delete(int id)
        {
            User user = Users.FindById(id);
            if (user == null)
            {
                return NotFound();
            }
```

```
            Users.RemoveUser(user);
            return StatusCode(HttpStatusCode.NoContent);
        }
    }
}
```

Testirati implementirane metode upotrebom **Postman-a**.

JQuery

JQuery je JavaScript biblioteka koja olakšava pisanje klijentskih aplikacija
Omogućava jednostavan pristup elementima HTML stranice

JQuery kod je moguće definisati:

- unutar zasebnog .js fajla koji se dodaje unutar html fajla preko script taga
- unutar script taga unutar html fajla

Sav JQeury kod vezan za pristup i manipulaciju elementima HTML stranice pišemo u okviru ove funkcije:

$(document).ready(function() {
...
})

Ovim se izbegava da se JQuery kod izvršava pre nego što se završi učitavanje stranice

JQuery sintaksa - $('selektor').akcija()

- $ - aktivira biblioteku JQuery
- (selektor) – selektuje HTML element
- akcija() – izvrši akciju nad selektovanim elementom

U JQuery elemente selektujemo isto kao u CSS-u:

- $('#btnCreate') – selektuje elemenat sa id-jem btnCreate
- $('. glavniDiv')– selektuje elemente sa klasom glavniDiv
- $('p') – selektuje sve p elemente

Događaji

- Dodaju se selektovanom elementu deklaracijom funkcije
- $('btnCreate'). click(function(){...});

Imena dogadjaja su iz JavaScript domena bez **on** prefiksa (click, change, submit)

Manipulacija DOM stablom

- **val() / val(novaVrednost)** – preuzimanje/ postavljanje vrednosti elementa forme
- **text() / text(novaVrednost)** – preuzimanje/postavljanje tekstualnog sadržaja elementa
- **attr(nazivAtributa) / attr(nazivAtributa, novaVrednost)** – preuzimanje / postavljanje vrednosti atribita sa zadatim nazivom
- **html() / html(novaVrednost)** – preuzimanje/postavljanje html sadržaja elementa

AJAX

Asynchronous JavaScript and XML.
Omogućava razmenu podataka sa serverom bez da se osvežava čitava HTML stranica.
Server i klijent mogu da razmenjuju podatke u formatu običnog teksta, HTML-a, XML-a i JSON-a.

GET AJAX poziv ima dva parametra:

- Putanju na serveru na koju upućujemo zahtev
- Callback funkciju, koja će se izvršiti kada stigne odgovor sa servera

Callback funkcija ima dva parametra:

- data – podaci koji stižu sa servera
- status – status kod odgovora sa servera

Kreirati **index.html** fajl i prekopirati sledeći kod:

In [ ]:
```
<!DOCTYPE html>
<html>
<head>
    <title></title>
    <meta charset="utf-8" />
    <script src="Scripts/jquery-3.4.1.js"></script>
    <script type="text/javascript">
        $(document).ready(function() {

            loadUsers();

            function loadUsers() {
                $.get("/api/users", function (data, status) {
                    let tableOfUsers = '<table border="1">';
                    tableOfUsers += '<tr><th>Id</th><th>First Name</th><th>Last Name
                    for (element in data) {
                        let user = '<td>' + data[element].Id + '</td>';
                        user += '<td>' + data[element].FirstName + '</td>';
                        user += '<td>' + data[element].LastName + '</td>';
                        tableOfUsers += '<tr>' + user + '</tr>';
                    }
                    tableOfUsers += '</table>';
                    $('#content').html(tableOfUsers);
                });
            }
        });

    </script>
</head>
<body>
    <h1>Users</h1>
    <div id="content"></div>
    <hr />
    <h3>Add new user:</h3>
    <label for="name">First Name:</label> <input type="text" id="firstName" name="na
    <label for="name">Last Name:</label> <input type="text" id="lastName" name="name
    <input type="submit" value="Create" id="btnCreate" />
</body>
</html>
```

POST AJAX poziv ima tri parametra:

- Putanju na serveru na koju upućujemo zahtev
- Podatke koje šaljemo na server
- Callback funkciju, koja će se izvršiti kada stigne odgovor sa servera

Callback funkcija ima dva parametra:

- data – podaci koji stižu sa servera
- status – status kod odgovora sa servera

Prekopirati sledeći kod ispod poziva loadUsers(); metode u **index.html** fajlu:

In [ ]:
```
$('#btnCreate').click(function () {
    let firstName = $('#firstName').val();
    if (firstName.length === 0) {
        alert('First Name is missing!');
        return;
    }

    let lastName = $('#lastName').val();
    if (lastName.length === 0) {
        alert('Last Name is missing!');
        return;
    }

    $.post('/api/users', { 'firstName': firstName, 'lastName': lastName },
        function (result) {
            alert('User is created!');
            $('#firstName').val('');
            $('#lastName').val('');
            loadUsers();
        }
    );
});
```