

Tehničko veleučilište u Zagrebu

Politehnički specijalistički diplomski stručni studij specijalizacija Informatika

Napredne tehnike programiranja web servisa
(.NET)



WebApi – ASP.NET MVC

- Uvod u ASP.NET MVC arhitekturu
 - Pojmovi: controller, akcija controller-a, routing
- ASP.NET MVC web api
 - Kreiranje web api sučelja
 - Korištenje postojećeg web api sučelja
- Api + javascript
 - Javascript predlošci
- Razlike – Api vs “obični” controller

ASP.NET MVC - uvod

- MVC – Model – view – controller
- Pozivom određene URL adrese na serveru se izvršava kod u odgovarajućoj akciji controller-a.
- Akcija = funkcija
 - Povratna vrijednost: ActionResult
 - Moguće proslijediti parametre
 - Preko URL-a
 - Kroz HTML formu

ASP.NET MVC - uvod

- Pri izvođenju akcije stvara se nova instanca kontrolera
 - Varijable ne ostaju očuvane između dva poziva
- Akcije i kontrolere se može anotirati s dodatnim metapodacima
- Povezivanje URL-a i akcije se naziva “Routing”

ASP.NET MVC - uvod

- Route (route) se konfiguriraju u posebnoj datoteci
- Ruta se sastoji od
 - URL resursa
 - URL parametara
 - Default vrijednosti
 - Imenski prostor za traženje odgovarajućeg kontrolera
 - Dodatna ograničenja

ASP.NET MVC - uvod

```
context.MapRoute(  
    "Upis_default",  
    "Predupis",  
    new { controller = "OnlineCheckinFeb2014", action = "Index" },  
    new { constraint = new SubdomainRouteConstraint("spec") },  
    new string[] { "Tvz.Web.Areas.Upis.Controller" }  
);
```

URL resurs
(/Predupis)

Naziv rute
(Upis_default)

Ograničenja
(constraint)

Linkanje kontrolera i
akcije, dodatno i
parametri

Imenski prostor za
pronalazak
odgovarajućeg
kontrolera

ASP.NET MVC – Api route

- Api rute su slične

```
public static void Register(HttpConfiguration config)
{
    config.Routes.MapHttpRoute(
        name: "Student courses",
        routeTemplate: "api/1.0/StudentData/Courses",
        defaults: new { controller = "StudentData", action = "GetCourses" }
    );
}
```

```
public class MvcApplication : System.Web.HttpApplication
{
    protected void Application_Start()
    {
        InitAutoMapper();
        AreaRegistration.RegisterAllAreas();

        WebApiConfig.Register(GlobalConfiguration.Configuration);
        FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
        RouteConfig.RegisterRoutes(RouteTable.Routes);
        BundleConfig.RegisterBundles(BundleTable.Bundles);
        AuthConfig.RegisterAuth();
    }
}
```

ASP.NET MVC - API

- Svaki kontroler nasljeđuje iz klase ApiController
- Novi kontroler se dodaje naredbom “Add new item”, “Web api controller class”
- Automatski se kreira 5 standardnih metoda
 - **Get ()** – dohvaćanje liste elemenata
 - **Get (id)** – dohvaćanje jednog specifičnog elementa
 - **Put (id, form)** – sinonim za izmjenu nekog elementa, može se koristiti i za dodavanje novog
 - **Post (form)** – sinonim za dodavanje novog elementa
 - **Delete (id)** – brisanje određenog elementa

ASP.NET MVC - API

- U pravilu, akcije api kontrolera vraćaju
 - **void** – u slučaju akcija koje vrše izmjenu predmeta (POST, PUT, DELETE)
 - **objekt** – u slučaju raznih varijanti GET poziva
- Akcija koja nema povratnu vrijednost signalizira pozivatelju preko HTTP status kodova da li je izvršavanje uspjelo ili ne
 - 200, 201 – OK
 - 304 – not modified (cache)
 - 403 – forbidden
 - 404 – not found
 - 500 – server error

ASP.NET MVC - API

- Ako je povratna vrijednost objekt, serijalizira se u odgovarajući tekstualni format
 - XML – ako napravimo poziv iz browsera
 - JSON – možemo zahtjevati preko “content-type” http header parametra doda “application/json”



ASP.NET MVC - API

http://localhost:50100/student

Content-Type	application/json	✕	Manage
Header	Value		

Send Preview Add to collection

Body Cookies (4) Headers (9) **STATUS** 200 OK **TIME** 139 ms

Pretty Raw Preview   JSON XML

```
1 [
2   "value1",
3   "value2"
4 ]
```

ASP.NET MVC - API

- REST api servis može (u pravilu) vraćati JSON ili XML
- Postoje dvije opcije kako možemo raditi s REST api servisima:
 - serijalizirati rezultat u odgovarajuće klase
 - ručno pretraživati XML ili JSON rezultat
- Naravno, puno je bolja opcija serijalizacija

ASP.NET MVC - API

- Kod RESTapi servisa dobra je stvar fleksibilnost, dok je posljedica fleksibilnosti potencijalno zastarjela dokumentacija ili nedostatak iste
- Iz XML-a se ne može sa 100% sigurnošću odrediti struktura klasa za serijalizaciju
- Ipak, moguće je napraviti niz zahtjeva na web servis te s dovoljnom dozom sigurnosti pretpostaviti strukturu odgovarajućih klasa za serijalizaciju

ASP.NET MVC - API

- Postoji nekoliko načina kako možemo iz XML dokumenta generirati C# klase:
 - u **VS2012**, postoji opcija “EDIT -> **paste special**”, koja automatski XML koji je na clipboardu pretvara u cs klasu prilikom akcije “paste”
 - **XSD tool** – može iz XML-a generirati odgovarajuću cs klasu

ASP.NET MVC - API

- Konzumaciju REST servisa koji vraća XML/JSON možemo napraviti kroz sljedećih nekoliko koraka:
 - Napraviti WebRequest na željeni resurs
 - Dohvatiti odgovor u obliku XML-a (ili JSON-a)
 - Pomoću odgovarajuće Serializer klase serijalizirati odgovor u objekt
- Umjesto ručno opisanog načina, možemo koristiti i plugin – primjerice, RESTSharp – koji olakšava rukovanje, te je dovoljno specificirati:
 - URL i parametre do željenog resursa
 - Klasu u koju je potrebno serijalizirati odgovor

ASP.NET MVC – js predlošci

- RESTapi se može vrlo efikasno koristiti za prikaz podataka u internet pregledniku – putem javascript predložaka
- Pogodnost je što odgovor servisa možemo zahtijevati u JSON formatu
- Tako dohvaćene podatke (ajax mehanizmom) možemo obraditi javascript radnim okvirom za predloške, te prikazati predložak

ASP.NET MVC – js predlošci

- Prednosti korištenja javascript predložaka u kombinaciji s REST servisima:
 - Dio logičke obrade (interpretacija podataka) se prenosi sa servera na klijenta – rasterećenje servera
 - Server služi samo za dohvat podataka – može se iskoristiti u raznim klijentima
- Nedostaci
 - Nedostatak compile-time podrške
 - Dvostruki zahtjev na server
- Neki primjeri: Mustache, Closure, Handlebars, ...

ASP.NET MVC – js predlošci

```
<script id="student-template" type="text/x-handlebars-template">
  <h3>{{Ime}} {{Prezime}}, {{JMBAG}}</h3>
  <h4>Položeni predmeti</h4>
  <ul>
    {{#PolozeniPredmeti}}
      <li>{{Naziv}} Ocjena: {{Ocjena}}</li>
    {{/PolozeniPredmeti}}
  </ul>
</script>
```

```
@section scripts{
<script type="text/javascript">
  $(document).ready(function () {
    $.ajax({
      url: "/student",
      data: { jmbag: "0246421358" },
      contentType: 'application/json',
      success: function (studentData) {
        var source = $("#student-template").html();
        var template = Handlebars.compile(source);
        $("#studentData").html(template(studentData));
      }
    });
  });
</script>
}
```

ASP.NET MVC vs API

- Pregled razlika između MVC kontrolera i API kontrolera:
 - **Razlika u konceptu** – MVC vraća HTML koji se prikazuje korisniku, API vraća podatke
 - API koristi GET, POST, PUT, DELETE; dok MVC koristi u pravilu GET i POST
 - API donosi automatsku serijalizaciju podataka u zahtjevani format (content-type)
 - API je općenito fleksibilniji:
 - Podaci se serijaliziraju automatski, sve promjene automatski propagirane
 - Može se koristiti iz bilo koje druge tehnologije

PITANJA?