14. Web Api

HTTP je protokol namenjen ne samo za dostavljanje veb sadržaja u vidu stranica. HTTP je moćna platforma i za izgradnju API-ja koji nude usluge i podatke. HTTP je jednostavan, fleksibilan i sveprisutni. Gotovo svaka platforma ima biblioteke za HTTP. HTTP servisi mogu da ostvare širok spektar klijenata, uključujući pregledače, mobilne uređaje kao i tradicionalne desktop aplikacije.

ASP.NET Web API je radni okvir za izgradnju veb API-ja u .NET Framework. U nastavku pokazaćemo način kreiranja ASP.NET Web API koji formira listu knjiga.

Kreiranje Web API projekta

U ovom predavanju, koristićete ASP.NET Web API za kreiranje veb API koji daje listu knjiga. Veb stranica koristi jQuery pristupanje api funkcijama i vraćanje rezultata.



Slika 14.1. Prikaz stranice koju kreiramo primenom Web API

Pokrenimo Visual Studio i odaberimo New Project.

U delu **Templates** odaberimo **Installed Templates** zatim proširimo čvor **Visual C#**. Pod opcijom **Visual C#**, odaberimo **Web**. U listi šablona projekta odaberimo **ASP.NET Web Application**. Imenujmo projekat "mojaKnjizara" a zatim **OK**.

New Project						?	×
▶ Recent		Sort by: Default	- II 🗉		Search (Ctrl+E)		ρ.
 Installed 		ASP.NET Core We	eb Application	Visual C#	Type: Visual C#		
 Visual C# Windows U Windows C Web Previou NET Core NET Stand. Cloud Test WCF Al Tools Azure Data Lak Visual C# Stream Analyti Other Language Other Project T 	Iniversal Lassic Desktop s Versions ard ex ke ke ses yppes	SPNET Web Ap	plication (NET Framework)	Visual C#	Project templates for cre- applications. You can cre- forms, MVC or Web API add many other features	ating ASP.NE ate ASP.NE application in ASP.NET	VET T Web is and T
Online Not finding what Open Visual	t you are looking for? I Studio Installer						
<u>N</u> ame:	mojaKnjizara						
Location:	c:\users\admin\doc	uments\visual studio 2017\Pr	rojects	•	Browse		
Solution name:	mojaKnjizara				Create directory for solut	ion	
<u>Framework</u>	.NET Framework 4.	5.2 -			Create new Git repositor	/	
					OK	Car	icel

Slika 14.2. Formiranje novog projekta

U dijalogu **New ASP.NET Project** odaberite šablon **Empty**. Odaberite **Web API**. Klik na **OK**.

	web Application - I	nojaknjizara			
Empty	Web Forms	MVC	Web API	Single Page Application	A project template for creating RESTIul HTTP services that can reach a broad range of clients including browsers and mobile devices.
dd folders a	nd core references t	for:			Authentication: No Authentication
Web For					Change Authentication
Web <u>F</u> on	ocker Compose su	pport (Requires	Oocker for Win	ndows)	
Web <u>F</u> orr	ocker Compose su tests	pport (Requires [Docker for Win	idows)	

Slika 14.3. Izbor šablona za novi projekta

Dodavanje Modela

Model je objekat koji predstavlja podatke u aplikaciji. ASP.NET Web API može automatski da serijalizuje model u formate kao što je JSON, XML. A zatim se takvi podaci mogu ubaciti u telo poruke koja je HTTP odgovor. Većina klijenata može da analizira ili XML ili JSON. Štaviše, možete naznačiti format koji želite da se prihvati kroz zaglavlje HTTP zahteva za poruke.

Kreirajmo sada jednostavan model.

U prozoru **Solution Explorer**, desni-klik na folder Models, a zatim iz kontekstnog menija birati opciju **Add** a zatim odabrati **Class**.

Add New Item - mojaKnjizara				?	×
 Installed 	Sort by: Default		Search (Ctrl+E)		ρ.
 Visual C# Code 	Class	Visual C#	Type: Visual C#		
Data General	Class for U-SQL	Visual C#	An empty class declaration		
Web Windows Forms	f Azure Function	Visual C#			
WPF	••• Interface	Visual C#			
SQL Server	ADO.NET Entity Data Model	Visual C#			
Storm Items	Application Manifest File	Visual C#			
v Onnie	Assembly Information File	Visual C#			
	Class Diagram	Visual C#			
	Code Analysis Rule Set	Visual C#			
	Code File	Visual C#			
	CSharp Editor Script	Visual C#			
	CSharp MonoBehaviour	Visual C#			
	DataSet	Visual C#			
	Debugger Visualizer	Visual C# 🚽			
Name: Knjiga					
			Add	Can	cel

Slika 14.4. Dodavanje nove klase za model

Imenujmo klasu "Knjiga". Zatim dodajmo sledeća svojstva ovoj klasi.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
namespace mojaKnjizara.Models
{
    public class Knjiga
    {
        public int Id { get; set; }
        public string Naziv { get; set; }
        public string Kategorija { get; set; }
        public decimal Cena { get; set; }
    }
}
```

Dodavanje kontrolera

U Web API, kontroler je objekat koji rukuje HTTP zahtevom. Dodaćemo kontroler koji može da vrati listu proizvoda ili jedan proizvod a koji je zahtevan na osnovu prosleđenog Id podatka.

Napomena

Ako ste već koristili ASP.NET MVC, onda ste već bliski i koristili ste kontrolere. Web API kontroleri su slični MVC kontrolerima, s tim što nasleđuju klasu ApiController umesto klase Controller.

Korak 1. U prozoru **Solution Explorer**, desni-klik na Controllers folder, zatim odaberite **Add** a zatim odaberite **Controller**.



Slika 15.5. Dodavanje kontrolera

Napomena: Kreiranje WebApi kontrolera moguće je i izborom opcije Web Api Controller Class

Korak 2. Odaberite Web API Controller - Empty. a zatim Add.

Installed			
Common Controller	*;;	MVC 5 Controller - Empty	Web API 2 Controller – Empty by Microsoft
	*;;	MVC 5 Controller with read/write actions	v2.0.0.0 An empty Web API controller.
	Ð	MVC 5 Controller with views, using Entity Framework	Id: ApiControllerEmptyScaffolder
	A	Web API 2 Controller – Empty	
	*;;	Web API 2 Controller with actions, using Entity Framework	
	* t ;	Web API 2 Controller with read/write actions	
	*;;	Web API 2 OData v3 Controller with actions, using Entity Framework	
	*\$	Web API 2 OData v3 Controller with read/write actions	
		crick here to go online and ind more scattolding extensions.	

Slika 14.6. Izbor kontrolera

Korak 3. U dijalogu Add Controller imenujte kontroler KnjigaController. Odaberite Add.

Add Controller			
Controller name:	KnjigaController		
		Add	Cancel

Slika 14.7. Definisanje naziva kontrolera

Mehanizam scaffolding kreira jedan fajl naziva KnjigaController.cs u folderu Controllers.



Slika 14.8. Pogled na prozor Solution Explorer

Napomena: Nije neophodno da se kontroler postavi u folder koji se naziva Controllers. Izbor naziva foldera je pitanje konvencije i organizacije fajlova u projektu.

Ako ovaj fajl nije već automatski otvoren, dvostrukim-klikom na fajl on će se otvoriti za editovanje. Uredite kod na način kako je to prikazano u nastavku:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;
using mojaKnjizara.Models;
namespace mojaKnjizara.Controllers
{
 public class KnjigaController : ApiController
  ł
    // Kreiranje podataka
    // Kasnije ovi podaci se preuzimaju iz neke baze
    Knjiga[] knjige = new Knjiga[]
    {
     new Knjiga { Id = 1, Naziv = "Avanture Toma Sojer",
Kategorija = "Dečiji", Cena = 100 },
```

```
new Knjiga { Id = 2, Naziv = "Rat i mir", Kategorija
= "Klasika", Cena = 375 },
      new Knjiga { Id = 3, Naziv = "Tihi Don", Kategorija =
"Klasika", Cena = 169.9M }
    };
    [System.Web.Http.HttpGet]
    public IEnumerable<Knjiga> SveKnjige()
    {
      return knjige;
    }
    [System.Web.Http.HttpGet]
    public IHttpActionResult Knjiga(int id)
    {
      var knjiga = knjige.FirstOrDefault((p) => p.Id == id);
      if (knjiga == null)
      {
          return NotFound();
      }
      return Ok(knjiga);
    }
    [HttpGet]
    public IHttpActionResult Knjiga(int id)
    {
      var knjiga = knjige.FirstOrDefault((p) => p.Id == id);
      if (knjiga == null)
      {
          return NotFound();
      }
      return Ok(knjiga);
    }
    [HttpPost]
    public string Post()
    {
      var naslov =
HttpContext.Current.Request.Params["naslov"];
      var kategorija =
HttpContext.Current.Request.Params["kategorija"];
      return "OK";
    }
  }
}
```

Kontroler definiše dve metode koje vraćaju knjige:

- Metoda SveKnjige vraća celu listu knjiga kao neki IEnumerable<Knjiga> tip.
- Metod Knjiga pronalazi jednu knjigu iz liste knjiga koje postoje.

Svaki metod na kontroleru odgovara jednom ili više URL-ova:

U primeru se koristi pojednostavljen primer u kome se knjige formiraju u kontroleru tako što ih kreiramo neposredno u kodu. Umesto toga u realnom slučaju, to bi bilo preko pristupa nekoj bazi podataka ili drugom spoljnjem izvoru podataka odnosno dobavljanjem podataka iz nje.

Tabela 14.1. Primer povezivanja metode i URL-a

Metod kontrolera	URI
SveKnjige	/api/Knjiga
Knjiga	/api/Knjiga/ <i>id</i>

Za metodu Knjiga argument id je u URI. Na primer, da bi se dobila knjiga čiji je ID = 5, odgovarajući URI je: api/knjiga/5.

Adresa na jedinstven način opisuje pristup do servisa i za jednu adresu može postojati jedan GET metod. Ukoliko metoda ima naziv **Get** onda nije neophodno da se postavlja dekorator **HttpGet** iznad naziva metode. Ako se koristi HttpGet onda se metodi koja prihvata Get zahtev treba dodati i dekorator za naziv akacije, na primer: ActionName("sveknjige")]

U ovom slučaju, adresa kojim se definiše Get zahtev bila bi: http://localhost:xxx/api/nazivKontrolera/sveknjige

Veb API ima u kreiranom projektu podrazumevano definisan način rutiranja URI ka metodama, za više informacija pogledati: https://docs.microsoft.com/en-us/aspnet/vebapi/overview/veb-api-routing-and-actions/routing-inaspnet-veb-api.

JavaScript / jQuery pozivi

U ovom poglavlju videćemo kako da dodamo neku HTML stranicu koja koristi AJAX da bi pozvala veb API. Koristićemo jQuery kako bi napravili AJAX poziv a zatim ažurirali stranicu sa rezultatom tj knjigama.

Korak 1. U prozoru Solution Explorer, desni-klik na projekat a zatim odaberite Add, zatim odaberite New Item.



Slika 14.9. Dodavanje nove stavke

Korak 2. U dijalogu Add New Item odaberite čvor Web pod opcijom Visual C#, a zatim odaberite HTML Page. Nazovite stranicu Index.html.

▲ Installed So ▲ Visual C# (Code (Data General (▶ Web (Windows Forms (WPF (▶ ASP.NET Core (SQL Server ())	nt by: Default HTML Page JavaScript File Style Sheet Web Form	Visual C# Visual C# Visual C#	Search (Ctrl+E) Type: Visual C# An HTML page that can i code	nclude clier	ہ م it-side
Visual C# Cde Data General Windows Forms WPF P ASP.NET Core SQL Server SQL Server	HTML Page JavaScript File Style Sheet Web Form	Visual C# Visual C# Visual C#	 Type: Visual C# An HTML page that can i code 	nclude clier	ıt-side
Data General Vindows Forms WPF ASP.NET Core SQL Server	JavaScript File Style Sheet Web Form	Visual C# Visual C#	code		
 ▶ Web ₩indows Forms ₩PF ▶ ASP.NET Core SQL Server 	Style Sheet	Visual C#			
WPF ASP.NET Core SQL Server	Web Form				
SQL Server		Visual C#			
Storm Items	Web Form with Master Page	Visual C#			
▷ Online	MVC 5 View Page (Razor)	Visual C#			
L	MVC 5 View Page with Layout (Razor)	Visual C#			
	Web API Controller Class (v2.1)	Visual C#			
	SignalR Hub Class (v2)	Visual C#			
*	Signaik Persistent Connection Class (v2)	Visual C#			
	ASP.NET Module	Visual C#			
4	Browser File	Visual C#			
	CoffeeScript File	Visual C#			

Slika 14.10. Izbor HTML stranice

Postoji više načina kako da se uključi biblioteka jQuery u veb stranicu. U ovom primeru korišćen je <u>Microsoft Ajax CDN</u>. Moguće je preuzeti celu biblioteku sa lokacije <u>http://jquery.com/</u>, a zatim je uključiti u projekat, kao na slici.

		**):	1	ADDITICATION INSTUNIS	• -		r
	Area					T	Quick Launch (Ctrl
۵	New Item	Ctrl+Shift+A					
1	Existing Item	Shift+Alt+A	Ļ				
**************************************	New Scaffolded Item New Folder Add ASP.NET Folder From Cookiecutter Docker Support Application Insights Telemetry REST API Client New Azure WebJob Project Existing Project as Azure WebJob Reference Service Reference	•	*	Build Rebuild Clean View Analyze Convert	· · +	Solution Ex Search Sol	Construction Construction Construction Connected Service Properties References App_Data App_Start Connollers cer Knigacontrollers
tp.	Connected Service Analyzer		•	Publish Publish Stored Procedures Overview			Models C= Knjiga.cs Global.asax
	HIML Page JavaScript File Style Sheet Web Form MVC 5 View Page (Bazor)		9 22	Scope to This New Solution Explorer View Show on Code Map Add		,	Index.html packages.config Web.config Web.Debug.co
	Web API Controller Class (v2.1)		Ě	Manage NuGet Packages			
	ADO.NET Entity Data Model SQL Server Database Helper (Razor v3)		\$	Set as StartUp Project Debug Initialize Interactive with Project		•	
+14.0	Class	Shift+Alt+C	Å	Cut	0	trl+X	
	C1035	Shine Altec	111	rasic	0	LUTY	1

Slika 14.11. Uključivanje postojeće biblioteke projektu

Nakon dodavanja, u Project folderu pojavljuje se ubačeni **js** document, kao na slici:



Slika 14.12. Solution Explorer prikaz dodate biblioteke

A zatim isti možete koristiti i u aplikaciji:

```
<!--<script
src="https://ajax.aspnetcdn.com/ajax/jQuery/jquery-
3.3.1.min.js"></script>-->
```

```
<script src="jquery-3.3.1.min.js"></script>
```

Testiranje Get metoda

Get zahtev se može testirati neposredno preko web čitača unosom adrese metode u adres-bar čaitača. U stranicu index.html može se dodati kod za testiranje Get poziva.

Koristeći form element:

</form>

```
Koristeći Ajax – primer 1:
```

```
$("#btnGet").on('click', function () {
    alert("GET Request Sent");
```

```
var request = $.ajax({
    url: "http://localhost:4713/api/knjiga",
    method: "GET",
    dataType: "json"
}).done(function (msg) {
    console.log(msg);
}).fail(function (jqXHR, textStatus) {
    console.log(jqXHR, textStatus);
});
```

```
});
```

U prvom slučaju odgovor sistema je vidljiv na stranici, a u drugom se može ispitati u konzolnom prozoru (F12).

Primena u HTML stranicama

Lista

Dakle, da bi se dobila lista knjiga potrebno je da se pošalje HTTP GET zahtev na URI: "/api/knjiga". jQuery funkcija getJSON šalje AJAX zahtev. Za odgovor očekuje se niz JSON objekata. Funkcija done definiše povratnu funkciju tzv. callback koji se poziva kada je zahtev obavljen uspešno. U povratnoj metodi menjamo DOM elemente uključujući povratne informacije:

Jedan podatak

Da bi se dobila knjiga po ID, potrebno je da korisnik učita ID I klikom na dugme pošalje zahtev tipa HTTP GET na adresu "/api/knjige/*id*", gde je *id* ID od knjige koja se traži:

}

Testiranje Post metoda

Post metodom klijent šalje podatke preko tela poruke. Serverska strana samo jednom čita podatke. Postoji nekoliko mogućih realizacija na serveskoj odnosno klijentskoj strani pogledajmo neke od njih.

Serverska metoda je bez argumenata. Ukoliko je naziv metode drugačiji od Post onda se dodaje dekorator [HttpPost] za ovu metodu.

```
public string Post()
{
    var naslov =
HttpContext.Current.Request.Params["naslov"];
    var kategorija =
HttpContext.Current.Request.Params["kategorija"];
    return "OK";
}
```

Ukoliko se šalje jedan podatak drugi će na prijemu biti **null**. Pogledajmo tri primera zahteva.

```
<form action=". . ./api/knjiga" method="POST">
    <div><input type="text" name="naziv"></div>
    <div><button type="submit">POST</button></div>
</form>
<form action=". . ./api/knjiga" method="POST">
    <div><input type="text" name="naziv"></div>
    <div><input type="text" name="kategorija"</div>
    <div><button type="submit">POST</button></div>
</form>
var knjigaZaSlanje = {
    "Id":"33",
    "Naziv":"Pesme",
    "Kategorija":"Poezija",
    "Cena":444.44
$("#post").on('click', function () {
   alert("POST Request Sent");
   var request = $.ajax({
        url: "http://localhost:4713/api/knjiga",
        method: "POST",
        data: knjigaZaSlanje,
        dataType: "xml"
   })
   .done(function (msg) {
       console.log(msg);
   })
   .fail(function (jqXHR, textStatus) {
       console.log(jqXHR, textStatus);
   });
```

});

Prvi se odnosi na slanje iz forme gde se šalje samo jedan podatak, drugi sa poslata dva podataka a treći preko Ajax-a. Sva tri slanja mogu se prihvatiti metodom koja je navedena.

Slanje kompleksnih tipova

Kompleksni objekti se mogu poslati preko forme ili koristeći Ajax. U oba slučaja možemo na serverskoj strani obezbediti prhvat taktvih objekata u

celosti. Naravno, da bi se primljeni podaci na serveru mogli prebaciti u objekat željenog tipa neophodno je da postoji deserijalizacija za taj objekata. Dakle, na serverskoj strani dovoljno je samo da se navede objekat koji se očekuje u prihvatnoj metodi.

```
public HttpResponseMessage Post(Knjiga k)
    if (ModelState.IsValid && k != null)
    {
        //// Create a 201 response.
        var response = new
HttpResponseMessage(HttpStatusCode.Created)
        {
            Content = new StringContent("OOKK")
        };
        return response;
    }
    else
    {
        return
Request.CreateResponse(HttpStatusCode.BadRequest);
    }
}
```

Ovakva metoda prihvata objekte koji imaju svojstva kojima je opisan objekat Knjiga. Na ovaj način se kreira očekivani objekat u celosti. Ukoliko nedostaju neke vrednosti objekat ima svojstva koja dobijaju podrazumevane vrenosti (npr. null ili 0)

Testiranja na klijenstoj strani koja su pokazana za slučaj metode rada sa metodom Post() mogu se koristiti i u ovom slučaju.

Slanje prostih tipova

Ako se šalje samo jedan podatak koristeći Post metodu i taj podatak mora biti pripremjen na strani klijenta u vidu objekta pri čemu se kao ključ ne navodi ništa: {"": "Tom Sojer"}. Na primer:

```
post2
.on('click', function () {
    alert("POST2 Request Sent");
```

```
var request = $.ajax({
    url: "http://localhost:4713/api/values",
    method: "POST",
    data: samoTekst, //var samoTekst={"":"Tom Sojer"};
    dataType: "json"
})
.done(function (msg) {
    console.log(msg);
})
.fail(function (jqXHR, textStatus) {
    console.log(jqXHR, textStatus);
});
```

});

Za ovako poslate podatke koji su prosti postoji mala modifikacija u kodu:

```
public void Post([FromBody]string value)
```

Testiranje aplikacije

Klikom na skraćenicu F5 pokreće se aplikacija sa mogućnošću debagovanja, **Debug mode**. Stranica izgleda na sledeći način:

🖻 🖅 Moja knjižara X 🕂 + 🗸		-		×
\leftrightarrow \rightarrow O \textcircled{a} O localhost:1695/	¢=	h	Ċ	
Sve knjiga				
Avanture Toma Sojer: \$100				
• Kat 1 mir 5575 • Tihi Don: \$169.9				
Pretraga po ID				
Pretraga po ID knjige				

Slika 14.13. Prikaz početne stranice

Da bi se dobila knjiga po ID, unese se vrednost u polje klikne na dugme za pretragu:

₽ ₽	Moja knjižara	×	+ ~				-		×
$\leftarrow \rightarrow$	۵ û	① localh	ost:1695/	a	☆	Zţ≡	L	Ċ	
Sve kn	ijiga								~
 Ava Rat Tihi 	nture Toma S i mir: \$375 Don: \$169.9	ojer: \$100	1						
Pretra	ga po ID	r I							
2	Pretraga po I	ID knjige							
Rat i mir:	\$375								

Slika 14.14. Prikaz rezultata pretrage

Ako se unese neispravan ID, server vraća HTTP grešku:

6 4	Moja knjižara	×	+ ~							-		×
$\leftarrow \rightarrow$	C 🏠	i localh	ost:1695/			ă	b [1]	☆	γ⊒	h	Ċ	
Sve kn • Ava • Rat • Tihi Pretra 7 Error: No	njiga anture Toma S i mir: \$375 i Don: \$169.9 oga po ID Pretraga po t Found	Sojer: \$100 D knjige)									

Slika 14.15. Prikaz pogrešnog zahteva za pretragu

HTTP Request / Response

Kada radite sa HTTP servisima može biti veoma korisno da se vide HTTP zahtevi odnosno odgovori. To se može izvesti pomoću alatke za programere koja se pokreće tasterom **F12**. Odaberite karticu **Network**.

Elements	Consol	е	Debu	igyer	Net	work	Performance	e I
		a a	*	8		≦ ▼• (Content t <mark>y</mark> p	e
Start prof	iling sessior	n (Ctrl+	E)	/		Protocol	Method	Result

Slika 14.16. Kartice u veb čitaču korišćene za razvoj



zatim pritisnite dugme za "hvatanje" poruka. Sada se vratite se na veb stranicu i pritisnite taster F5 da ponovo učitati veb stranicu. Veb čitač će izvršiti hvatanje HTTP saobraćaja između veb čitača i veb servera. Slika pokazuje mrežni saobraćaj za stranicu:

localhost - DevTools - Microsoft Edge							
Elements Console Debugger Net	work Perfor	mance Memory	Emulation				
🕨 🗏 🖬 📽 🏷 🍆 🍋	🗧 🔻 Conte	nt type					
Name	Protocol Met	hod Result	Content type	Received	Time	Initiator	Oms
http://localhost:1695/	HTTP GET	304 Not Modified		(from cache)	18,72 ms	document	
jquery-3.3.1.min.js http://localhost:1695/	HTTP GET	304 Not Modified		(from cache)	4,73 ms		
http://localhost:1695/	HTTP GET	200 OK		(from cache)	0 s		
Knjiga http://localhost:1695/api/	HTTP GET	200 OK	application/json	205 B	17,44 ms	XMLHttpRequest	

Slika 14.17. Network kartica

Nađite poziv za URI **api/Knjiga**/. Odaberite tu stavku, a zatim pogledajte detalje u desnom delu prozora. Među detaljima postoje tabovi koji prikazuju **request/response** zaglavlja i telo poruke.

Headers Body Parameters Cookies Timings
Request URL: http://localhost:1695/api/Knjiga
Request Method: GET
Status Code: 📕 200 / OK
✓ Request Headers
Accept: application/json, text/javascript, */*; q=0.01
Accept-Encoding: gzip, deflate
Accept-Language: sr-Latn-RS
Cache-Control: max-age=0
Connection: Keep-Alive
Host: localhost:1695
Referer: http://localhost:1695/
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) C
X-Requested-With: XMLHttpRequest
✓ Response Headers
Cache-Control: no-cache
Content-Length: 205
Content-Type: application/json; charset=utf-8
Date: Sun, 20 May 2018 20:59:12 GMT
Expires: -1
Pragma: no-cache
Server: Microsoft-IIS/10.0
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
X-SourceFiles: =?UTF-8?B?YzpcdXNIcnNcYWRtaW5cZG9jdW1IbnRzXHZpc3VhbCBzdHVkaW8gMjAx





Slika 14.18. Praćenje sadržaja poruka

Pitanja i zadaci za proveru znanja

- 1. Objasnite pojam Web Api aplikacija. Da li poznajete neki api?
- 2. Kreirajte novi projekat tipa Web Api. Da li možete birati MVC opciju pri kreiranju projekta?
- Ubacite vaš model Radnik u ovaj projekat. Obratite pažnju na nazive kolona. Zašto?
- 4. Kreirati kontroler za podatke o radnicima.
- 5. Iz koje klase je izvedena klasa kontrolera?
- 6. Kreirajte metode koje vraćaju sve radnike odnosno jednog radnika.
- 7. Definišite URL za pozive kreiranih metoda.
- Dodajte projektu HTML stranicu a zatim primenom JavaScript-a i jQuery jezika napišite forme za dobavljanje podataka preko vaših metoda.
- Napraviti novi projekat koji je tipa ASP MVC aplikacije pa tom projektu dodajte kontroler i HTML stranicu kao u prethodnom slučaju.