

Example 'SupportCenter'

ASP.NET Web API

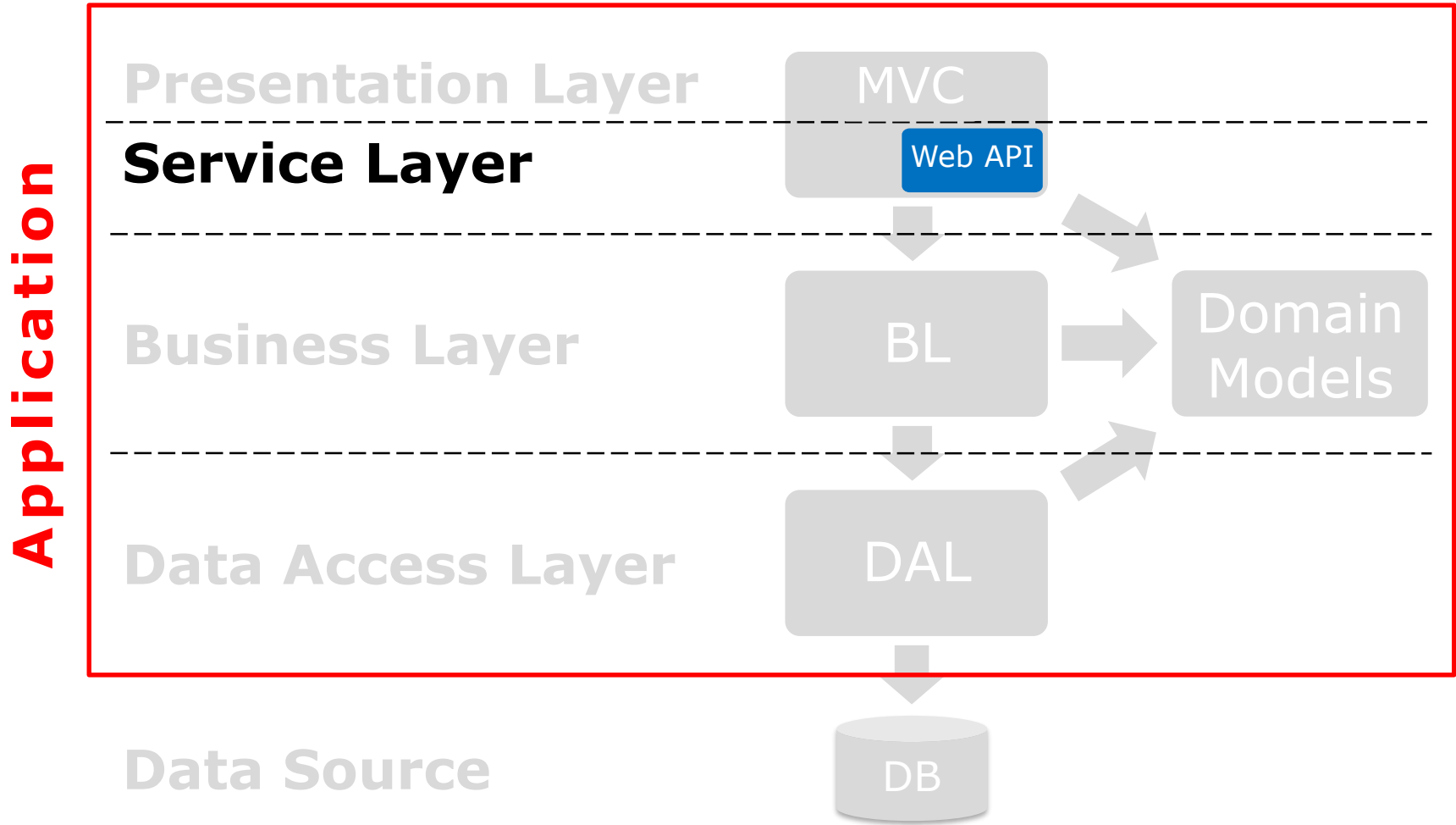
Info

- De applicatie moet voorzien worden van een (web)service-georiënteerde implementatie volgens een RESTful-design
- Hiervoor gaan we gebruik maken van het ASP.NET Web API-framework

MVC & Web API

AJAX-calls
(via jQuery)

MVC & Web API architectuur



Oefening

- Voeg het Web API-framework toe aan het 'UI-MVC'-project
- Voorzie default-routing configuratie via 'WebApiConfig.cs' in map 'App_Start'
 - namespace: SC.UI.Web.MVC
 - `http://www.domain.tld/api/{controller}/{id}`
 - url-prefix: api
 - id: optioneel
- Voorzie in de map 'Controllers' een submap 'Api' voor 'Web API Controllers'

Installatie 'Web API'

NuGet Package Manager: UI-MVC

Package source: Filter: ☐ Include prerelease

Microsoft.AspNet.WebApi.Core
This package contains the core runtime assemblies for ASP.NET Web API.

Microsoft.AspNet.WebApi
This package contains everything you need to host ASP.NET Web API on IIS.

Preview

Review Changes
Visual Studio is about to make changes to this solution. Click OK to proceed with the changes listed below.

UI-MVC

Installing:

- Microsoft.AspNet.WebApi.Client 5.2.3
- Microsoft.AspNet.WebApi.Core 5.2.3
- Microsoft.AspNet.WebApi.WebHost 5.2.3
- Microsoft.AspNet.WebApi 5.2.3

☐ Do not show this again

2

Microsoft.AspNet.WebApi

Action: Version:

1

Options

☒ Show preview window

Dependencies

File conflicts

[Learn about...](#)

License Acceptance

The following package(s) require that you accept their license terms before installing.

- Microsoft.AspNet.WebApi.Client** Author(s): Microsoft
[View License](#)
- Microsoft.AspNet.WebApi.Core** Author(s): Microsoft
[View License](#)
- Microsoft.AspNet.WebApi.WebHost** Author(s): Microsoft
[View License](#)
- Microsoft.AspNet.WebApi** Author(s): Microsoft
[View License](#)

By clicking "I Accept," you agree to the license terms for the package(s) listed above. If you do not agree to the license terms, click "I Decline."

3

\App_Start\WebApiConfig.cs

```
using System.Web.Http;

namespace SC.UI.Web.MVC
{
    public static class WebApiConfig
    {
        public static void Register(HttpConfiguration config)
        {
            config.Routes.MapHttpRoute(
                name: "DefaultApi",
                routeTemplate: "api/{controller}/{id}",
                defaults: new { id = RouteParameter.Optional }
            );
        }
    }
}
```

Global.asax

```
using System.Web.Http;

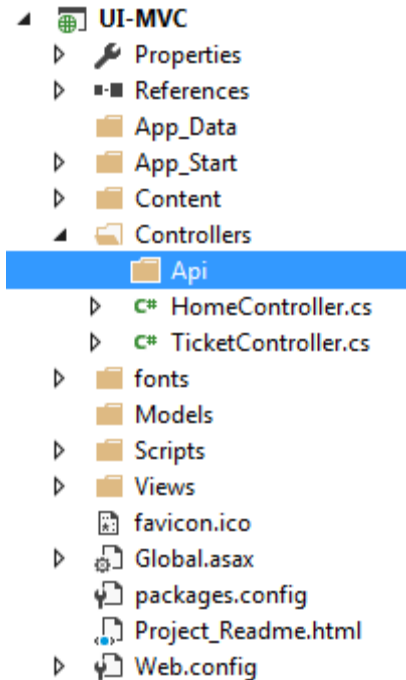
...

protected void Application_Start()
{
    AreaRegistration.RegisterAllAreas();
    GlobalConfiguration.Configure(WebApiConfig.Register);
    FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
    RouteConfig.RegisterRoutes(RouteTable.Routes);
    BundleConfig.RegisterBundles(BundleTable.Bundles);
}

...
```

WebApi routing configuratie moet, vanwege prefix 'api',
gebeuren voor MVC routing configuratie!

Submap 'Api'



Oefening

- Voorzie in de Details-pagina van een ticket dat de lijst met responses pas worden ingeladen als men op een knop 'Laad responses' klikt
- Stap 1: Back-end
 - TicketResponseController toevoegen
 - Template: Web API 2 – Empty
 - Privaat veld 'mgr' (ITicketManager)
 - Actionmethode 'Get' toevoegen
 - parameter: ticketNumber (int)
 - return
 - » type?
 - » response statuscodes?
 - indien er responses zijn -> 200 OK
 - indien er geen responses zijn -> 204 No Content
 - testen met Fiddler
 - GET
 - url: .../api/TicketResponse?ticketnumber=1

TicketResponseController

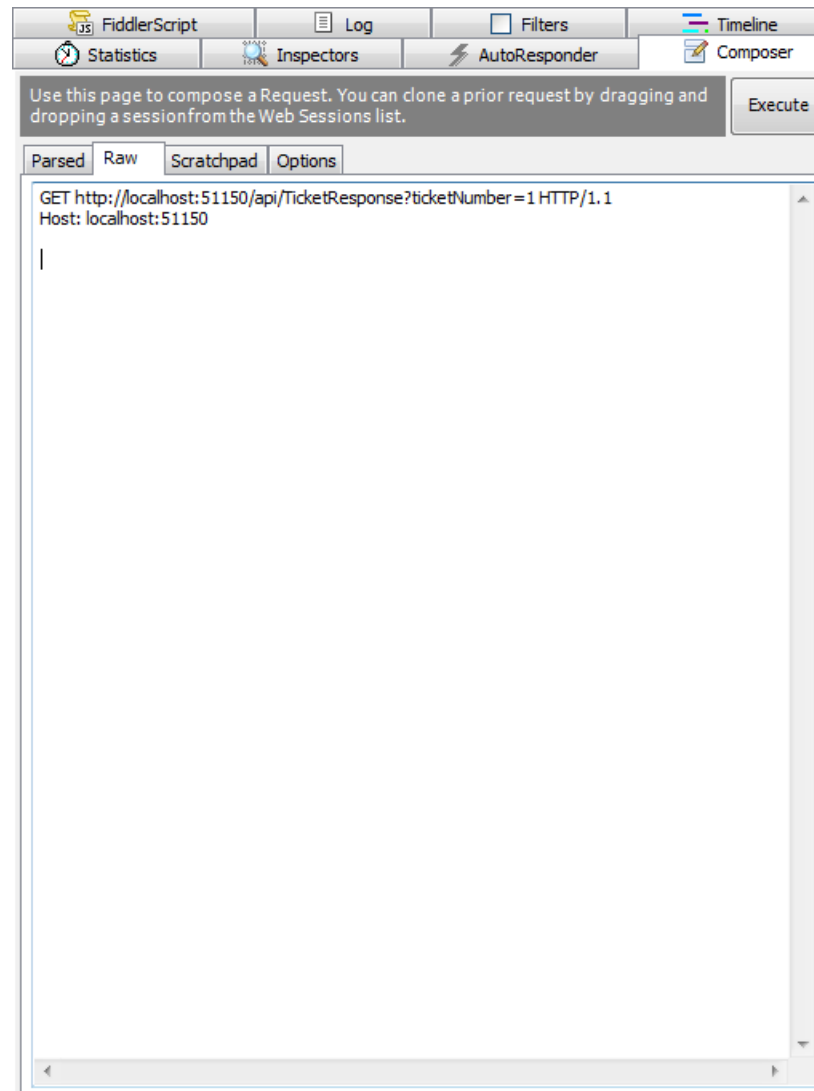
```
...
using SC.BL;
using SC.BL.Domain;
namespace SC.UI.Web.MVC.Controllers.Api
{
    public class TicketResponseController : ApiController
    {
        private ITicketManager mgr = new TicketManager();

        // GET: api/TicketResponse?ticketNumber=5
        public IHttpActionResult Get(int ticketNumber)
        {
            var responses = mgr.GetTicketResponses(ticketNumber);

            if (responses == null || responses.Count() == 0)
                return StatusCode(HttpStatusCode.NoContent);

            return Ok(responses);
        }
    }
}
```

Fiddler – Composer



Fiddler – Inspector

The Fiddler Inspector window displays an HTTP request and response. The request is a GET to `http://localhost:51150/api/TicketResponse?ticketNumber=1`. The response is an HTTP/1.1 200 OK with a Content-Type of `application/json; charset=utf-8`. The response body is a JSON array with one object: `[{"Id":1,"Text":"Account was geblokkeerd","Date":"2012-09-09T13:24:48"}]`. A red box labeled "http-request" highlights the request area, and another red box labeled "http-response" highlights the response area. A red arrow points from the response body to the JSON view on the right.

http-request

http-response

The Fiddler Inspector window shows the JSON response body in the JSON view. The response is a JSON array with three objects. The first object is `{ "Date": "2012-09-09T13:24:48", "Id": 1, "IsClientResponse": false, "Text": "Account was geblokkeerd", "Ticket": null }`. The second object is `{ "Date": "2012-09-09T13:29:11", "Id": 2, "IsClientResponse": false, "Text": "Account terug in orde en nieuw paswoord ingesteld", "Ticket": null }`. The third object is `{ "Date": "2012-09-10T07:22:36", "Id": 3, "IsClientResponse": true, "Text": "Aanmelden gelukt en paswoord gewijzigd", "Ticket": null }`. The JSON view is selected, and the text "JSON parsing completed." is visible at the bottom.

JSON parsing completed.

Oefening (vervolg)

- Stap 2: Front-end
 - vervang de uitwerking met partial-view door een AJAX-call die de ticketresponses zal ophalen en tonen in de pagina
 - Voorzie volgende html-elementen onder 'Responses'
 - button-element 'Laad antwoorden'
 - class = "btn btn-default" (bootstrap)
 - onclick-event = roept functie 'loadResponsesOfTicket' op met als parameter het id of het op dat moment getoonde ticket
 - div-element (id = addNewResponse)
 - initieel 'onzichtbaar'
 - met hierin een table-element (id = responses, class = table)
 - » kolomtitels: 'Response', 'Date' en 'IsClientResponse'

Details.cshtml

```
...

<h4>Responses</h4>
@* @Html.Partial("_TicketResponsesPartial", Model.Responses) *@
@* /* OF: via ViewBag */ *@
@* @Html.Partial("_TicketResponsesPartial"
    , (IEnumerable<SC.BL.Domain.TicketResponse>)ViewBag.Responses) *@
@* /* OF: via AJAX-call */ *@
<p><button class="btn btn-default" onclick="loadResponsesOfTicket(@Model.TicketNumber)">Laad
    antwoorden</button></p>
<div id="addNewResponse" style="display: none">
    <table id="responses" class="table">
        <tr>
            <th>Response</th>
            <th>Date</th>
            <th>IsClientResponse</th>
        </tr>
    </table>
</div>
```

Oefening (vervolg)

- Stap 3: Front-end-logica (JavaScript)
 - Voorzie volgende javascript-functies
 - loadResponsesOfTicket(ticketNumber)
 - maak een ajax-call naar de back-end om de ticketresponse-data op te halen adhv het binnenkomende 'ticketNumber'
 - » vraag om de response in json-format te sturen
 - » bij een goede response wordt 'showTicketResponses' aangeroepen en teruggestuurde data meegegeven
 - » bij een fout toon je adhv een alert de boodschap "Oeps, something went wrong!"
 - showTicketResponses(responses)
 - loop over elk element van de verzameling van ticketresponse-data en roep 'addResponseToList' aan en geef telkens het element hieraan mee
 - addResponseToList(response)
 - voeg de response toe aan de tabel met id 'responses'

Details.cshtml

```
@section scripts {
    <script type="text/javascript">
        // Toon responses
        function loadResponsesOfTicket(ticketNumber) {
            $.ajax('/api/TicketResponse?ticketnumber='+ticketNumber, {
                type: 'GET',
                dataType: 'json' // data-type expected back (response-data parsed to object)
            })
            .done(function (data) { showTicketResponses(data); })
            .fail(function () { alert('Oeps, something went wrong!'); });
        }
        function showTicketResponses(responses) {
            $.each(responses, function (index, value) { addResponseToList(value); });
            $("#addNewResponse").show();
        }
        function addResponseToList(response) {
            var date = new Date(response.Date);
            var checked = response.IsClientResponse ? 'checked="checked"' : '';
            $('table#responses').append('<tr>'
                + '<td>'+response.Text+'</td>'
                + '<td>'+date.toLocaleDateString()+ ' '+date.toLocaleTimeString()+ '</td>'
                + '<td><input type="checkbox" class="check-box" disabled="disabled" '
                    + checked+ ' /></td>'
                + '</tr>');
        }
    </script>
}
```

Oefening

- Voorzie in de Details-pagina, onder de lijst van responses, een tekstvak en een knop 'Verzenden', om een antwoord toe te voegen aan het ticket
- Stap 1: Back-end
 - TicketResponseController
 - Actionmethode 'Post'
 - parameter(s)?
 - » data moet uit request-body gelezen worden -> complex-type parameter 'response' van het type 'NewTicketResponseDTO' (in map 'Models')
 - Properties: TicketNumber (int), ResponseText (string), IsClientResponse (bool)
 - return response-codes?
 - » indien gelukt -> 201 Created (met locatie!)
 - » indien niet gelukt -> 404 Bad Request -> en geef volgend bericht mee "Er is iets misgelopen bij het registreren van het antwoord!"
 - testen met Fiddler
 - POST
 - url: .../api/TicketResponse
 - data

NewTicketResponseDTO

...

```
namespace SC.UI.Web.MVC.Models
{
    public class NewTicketResponseDTO
    {
        public int TicketNumber { get; set; }
        public string ResponseText { get; set; }
        public bool IsClientResponse { get; set; }
    }
}
```

TicketResponseController

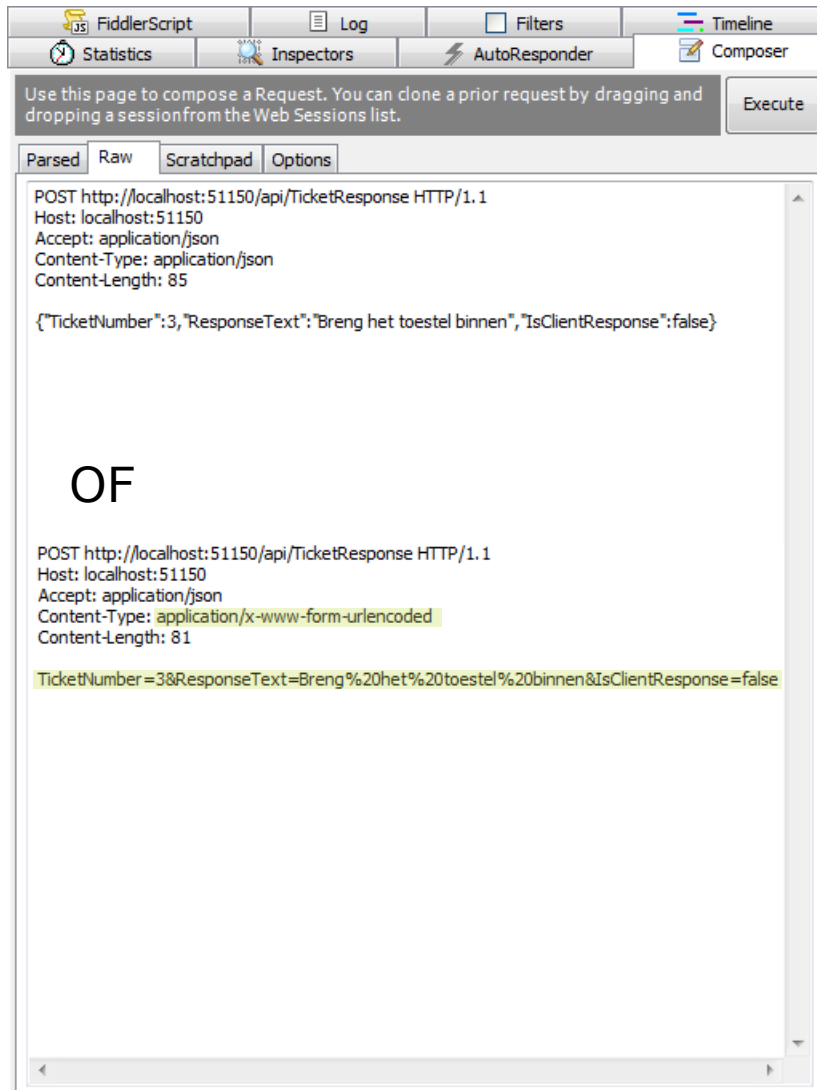
```
...
using SC.BL;
using SC.BL.Domain;
namespace SC.UI.Web.MVC.Controllers.Api
{
    public class TicketResponseController : ApiController
    {
        ...

        // POST: api/TicketResponse
        public IHttpActionResult Post(NewTicketResponseDTO response)
        {
            TicketResponse createdResponse = mgr.AddTicketResponse(response.TicketNumber
                                                                    , response.ResponseText, response.IsClientResponse);

            if (createdResponse == null)
                return BadRequest("Er is iets misgelopen bij het registreren van het antwoord!");

            return CreatedAtRoute("DefaultApi",
                                new { Controller = "TicketResponse", id = createdResponse.Id },
                                createdResponse);
        }
    }
}
```

Fiddler



FiddlerScript Log Filters Timeline
Statistics Inspectors AutoResponder Composer

Use this page to compose a Request. You can clone a prior request by dragging and dropping a session from the Web Sessions list. Execute

Parsed Raw Scratchpad Options

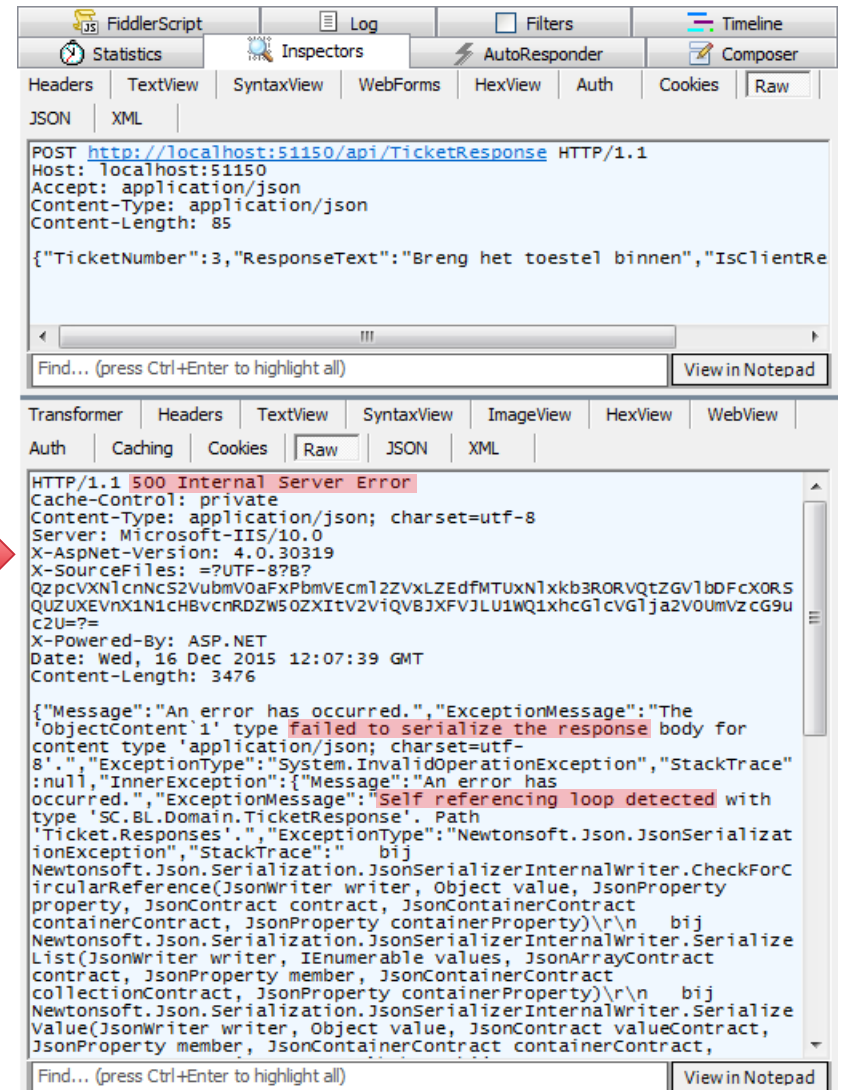
```
POST http://localhost:51150/api/TicketResponse HTTP/1.1
Host: localhost:51150
Accept: application/json
Content-Type: application/json
Content-Length: 85

{"TicketNumber":3,"ResponseText":"Breng het toestel binnen","IsClientResponse":false}
```

OF

```
POST http://localhost:51150/api/TicketResponse HTTP/1.1
Host: localhost:51150
Accept: application/json
Content-Type: application/x-www-form-urlencoded
Content-Length: 81

TicketNumber=3&ResponseText=Breng%20het%20toestel%20binnen&IsClientResponse=false
```



FiddlerScript Log Filters Timeline
Statistics Inspectors AutoResponder Composer

Headers TextView SyntaxView WebForms HexView Auth Cookies Raw

JSON XML

```
POST http://localhost:51150/api/TicketResponse HTTP/1.1
Host: localhost:51150
Accept: application/json
Content-Type: application/json
Content-Length: 85

{"TicketNumber":3,"ResponseText":"Breng het toestel binnen","IsClientResponse":false}
```

Find... (press Ctrl+Enter to highlight all) View in Notepad

Transformer Headers TextView SyntaxView ImageView HexView WebView

Auth Caching Cookies Raw JSON XML

```
HTTP/1.1 500 Internal Server Error
Cache-Control: private
Content-Type: application/json; charset=utf-8
Server: Microsoft-IIS/10.0
X-AspNet-Version: 4.0.30319
X-SourceFiles: =?UTF-8?B?QzpcVXN1cnNCS2VubmVoaFxpbmVEcm12ZVxLZEdfMTUxN1xkb3RORVQVtZGV1bDcXORsQUZUXEVnX1N1CHBvcnRDZW50ZXItv2ViQVBJXjVJLU1WQ1xhcG1cvG1ja2V0UmvzCG9uC2U=?=
X-Powered-By: ASP.NET
Date: Wed, 16 Dec 2015 12:07:39 GMT
Content-Length: 3476

{"Message":"An error has occurred.,"ExceptionMessage":"The 'ObjectContent'1' type failed to serialize the response body for content type 'application/json; charset=utf-8'.","ExceptionType":"System.InvalidOperationException","StackTrace":null,"InnerException":{"Message":"An error has occurred.,"ExceptionMessage":"Self referencing loop detected with type 'SC.BL.Domain.TicketResponse'. Path 'Ticket.Responses'.","ExceptionType":"Newtonsoft.Json.JsonSerializationException","StackTrace":"    bij\n    Newtonsoft.Json.Serialization.JsonSerializerInternalWriter.CheckForCircularReference(JsonWriter writer, Object value, JsonProperty property, JsonContract contract, JsonContainerContract containerContract, JsonProperty containerProperty)\r\n    bij\n    Newtonsoft.Json.Serialization.JsonSerializerInternalWriter.SerializeList(JsonWriter writer, IEnumerable values, JsonArrayContract contract, JsonProperty member, JsonContainerContract collectionContract, JsonProperty containerProperty)\r\n    bij\n    Newtonsoft.Json.Serialization.JsonSerializerInternalWriter.SerializeValue(JsonWriter writer, Object value, JsonContract valueContract, JsonProperty member, JsonContainerContract containerContract,
```

Find... (press Ctrl+Enter to highlight all) View in Notepad

Oefening (vervolg)

- PROBLEEM

Het ticketresponse-object verwijst via property 'Ticket' naar het bijhorende ticket-object, welk op zich terug naar het ticketresponse-object verwijst via de property 'Responses'

-> circulaire referentie!!

- Aandachtspunt: indien geassocieerde data niet ingeladen is, maar Lazy Loading is geconfigureerd (waardoor je proxy-objecten hebt), zal alsnog de geassocieerde data ingeladen worden tijdens het serialiseren en ontstaat er toch circulaire referentie!)

- OPLOSSING

1. Domein model voorzien van attributen om aan te geven welke properties wel en niet geserialized moeten worden
 - JsonIgnor-attribute voor JSON.NET-framework (alleen json-serializing)
 - DataContract-/DataMember-attributes (algemeen)
 - vervuiling van de domein modellen! ☹
2. DTO(s) gebruiken ☺
'TicketResponseDTO' (in map 'Models'): zelfde properties als het domein model 'TicketResponse' zonder Ticket-property, maar in de plaats voorzien van een property 'TicketNumberOfTicket' om de 'unique identifier' van het ticket bij te houden in de DTO

TicketResponseDTO

...

```
namespace SC.UI.Web.MVC.Models
{
    public class TicketResponseDTO
    {
        public int Id { get; set; }
        public string Text { get; set; }
        public DateTime Date { get; set; }
        public bool IsClientResponse { get; set; }
        public int TicketNumberOfTicket { get; set; }
    }
}
```

TicketResponseController

```
...
using SC.BL;
using SC.BL.Domain;
namespace SC.UI.Web.MVC.Controllers.Api
{
    public class TicketResponseController : ApiController
    {
        ...

        // POST: api/TicketResponse
        public IHttpActionResult Post(NewTicketResponseDTO response)
        {
            TicketResponse createdResponse = mgr.AddTicketResponse(response.TicketNumber
                                                                    , response.ResponseText, response.IsClientResponse);

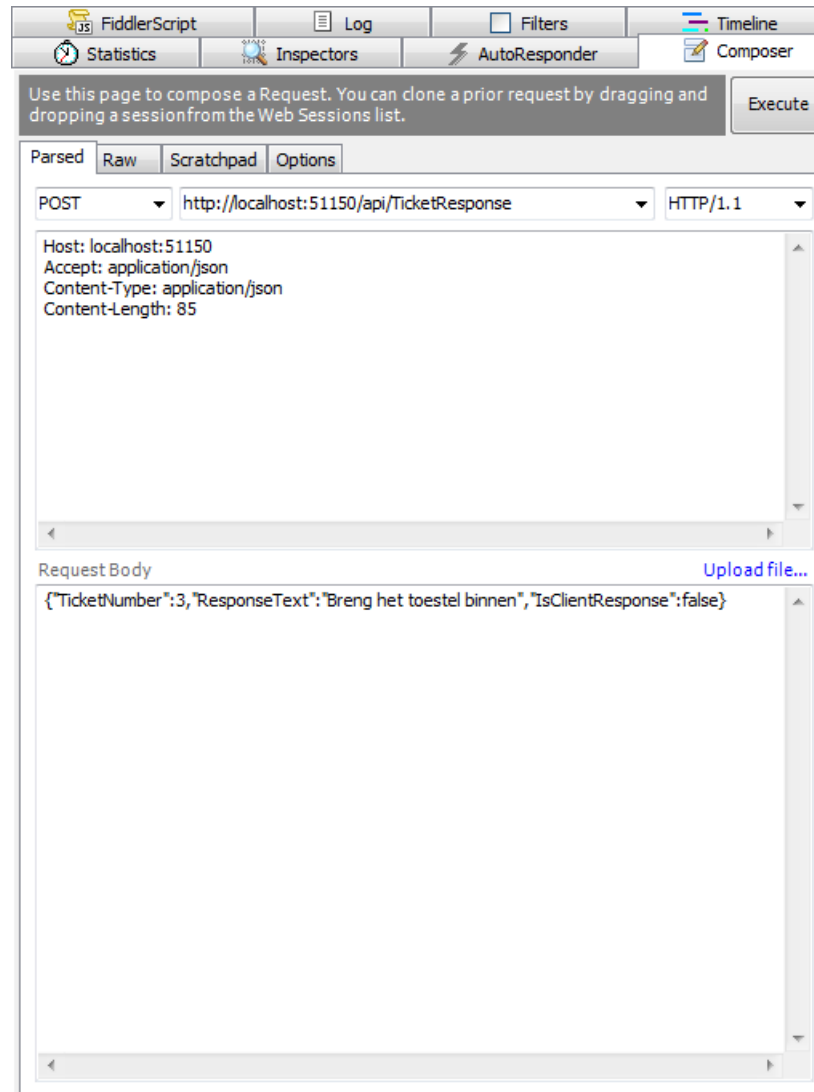
            if (createdResponse == null)
                return BadRequest("Er is iets misgelopen bij het registreren van het antwoord!");

            TicketResponseDTO responseData = new TicketResponseDTO()
            {
                Id = createdResponse.Id,
                Text = createdResponse.Text,
                Date = createdResponse.Date,
                IsClientResponse = createdResponse.IsClientResponse
            };

            return CreatedAtRoute("DefaultApi",
                                new { Controller = "TicketResponse", id = responseData.Id },
                                responseData);
        }
    }
}
```

Hiervoor kunnen tools
zoals 'AutoMapper'
voor gebruikt worden

Fiddler – Composer



Fiddler – Inspector

FiddlerScript Log Filters Timeline
Statistics Inspectors AutoResponder Composer
Headers TextView SyntaxView WebForms HexView Auth Cookies Raw
JSON XML

POST <http://localhost:51150/api/TicketResponse> HTTP/1.1
Host: localhost:51150
Accept: application/json
Content-Type: application/json
Content-Length: 85

{ "TicketNumber": 3, "ResponseText": "Breng het toestel binnen", "IsClientRe

Find... (press Ctrl+Enter to highlight all) View in Notepad

Transformer Headers TextView SyntaxView ImageView HexView WebView
Auth Caching Cookies Raw JSON XML

HTTP/1.1 201 Created
Cache-Control: no-cache
Pragma: no-cache
Content-Type: application/json; charset=utf-8
Expires: -1
Location: <http://localhost:51150/api/TicketResponse/5>
Server: Microsoft-IIS/10.0
X-AspNet-Version: 4.0.30319
X-SourceFiles: =?UTF-8?B?QzpcVXN1cnNcS2VubmV0aFxBbmVEcm12ZVxLZEdfMTUXN1
X-Powered-By: ASP.NET
Date: Wed, 16 Dec 2015 13:33:01 GMT
Content-Length: 124

{ "Id": 5, "Text": "Breng het toestel binnen", "Date": "2015-12-16T14:33:01.4

Find... (press Ctrl+Enter to highlight all) View in Notepad

Transformer Headers TextView SyntaxView ImageView HexView WebView
Auth Caching Cookies Raw JSON XML

JSON

- ...Date=2015-12-16T14:33:01.4874544+01:00
- ...Id=5
- ...IsClientResponse=False
- ...Text=Breng het toestel binnen
- ...Ticket=(null)

Expand All Collapse JSON parsing completed.

Oefening (vervolg)

- Stap 2: Front-end
 - voeg onder de tabel 'responses' (maar binnen de div 'addNewResponse') volgende elementen toe:
 - input-element (id & name = responseText)
 - type = text
 - button-element 'Verzenden'
 - class = "btn btn-default"
 - onclick-event: roept functie 'postResponse' aan

TicketResponseController

```
...
<h4>Responses</h4>
@*@Html.Partial("_TicketResponsesPartial", Model.Responses)*@
@* /* OF: via ViewBag */ *@
@*@Html.Partial("_TicketResponsesPartial",
    (IEnumerable<SC.BL.Domain.TicketResponse>)ViewBag.Responses)*@
@* /* OF: via AJAX-call */ *@
<p><button class="btn btn-default" onclick="loadResponsesOfTicket(@Model.TicketNumber)">Laad
    antwoorden</button></p>
<div id="addNewResponse" style="display: none">
    <table id="responses" class="table">
        <tr>
            <th>Response</th>
            <th>Date</th>
            <th>IsClientResponse</th>
        </tr>
    </table>
    <input id="responseText" name="responseText" type="text" class="form-control"
        style="width: 300px; float: left; margin-right: 5px" />
    <button type="button" class="btn btn-default" onclick="postResponse()">Verzenden</button>
</div>
```

Oefening (vervolg)

- Stap 3: Front-end-logica (JavaScript)
 - Voorzie volgende javascript-functie
 - `postResponse()`
 - maak een ajax-call naar de back-end (POST)
 - » geef de nodige data mee die de server verwacht: ticketnummer (van het getoonde ticket), antwoord (uit het tekst-veld 'responseText' halen) en geef aan dat het geen antwoord van de klant is
 - geef aan dat de meegestuurde data in json-formaat is
 - » vraag om de response in json-formaat te sturen
 - » bij een goed response roep je 'addResponseToList' aan en geef je de teruggestuurde ticketresponse-data mee, en maak je nadien het veld 'responseText' leeg
 - » bij een fout toon je adhv een alert de boodschap "Oeps, something went wrong!"

Details.cshtml

```
@section scripts {  
    <script type="text/javascript">  
        ...  
  
        // Voeg nieuw response toe  
        function postResponse() {  
            var ticketNumber = '@Model.TicketNumber';  
            var response = $('#responseText').val();  
            if (response != '') {  
                $.ajax('/api/TicketResponse', {  
                    type: 'POST',  
                    data: JSON.stringify({ ticketNumber: ticketNumber,  
                                          responseText: response,  
                                          isClientResponse: false }),  
                    contentType : 'application/json',  
                    dataType: 'json' // data-type expected back  
                })  
                .done(function (data) { addResponseToList(data);  
                                     $('#responseText').val(''); })  
                .fail(function () { alert('Oeps, something went wrong!'); });  
            }  
        }  
    </script>  
}
```

Oefening

- Voorzie in de Details-pagina, achter de huidige status, een knop 'Close' om de status van het ticket te wijzigen naar 'Closed'
- Stap 1: Back-end
 - TicketController toevoegen
 - Template 'Web API 2 Controller – Empty'
 - Private veld: 'mgr' (ITicketManager)
 - Actionmethode 'PutTicketStateToClosed'
 - http-methode: PUT
 - routing-url: .../api/Ticket/{id}/State/Closed
 - » attribute-routing inschakelen!
 - parameter: id (int)
 - return response-code
 - » indien gelukt -> 204 No Content

\App_Start\WebApiConfig.cs

```
using System.Web.Http;

namespace SC.UI.Web.MVC
{
    public static class WebApiConfig
    {
        public static void Register(HttpConfiguration config)
        {
            config.MapHttpAttributeRoutes();

            config.Routes.MapHttpRoute(
                name: "DefaultApi",
                routeTemplate: "api/{controller}/{id}",
                defaults: new { id = RouteParameter.Optional }
            );
        }
    }
}
```

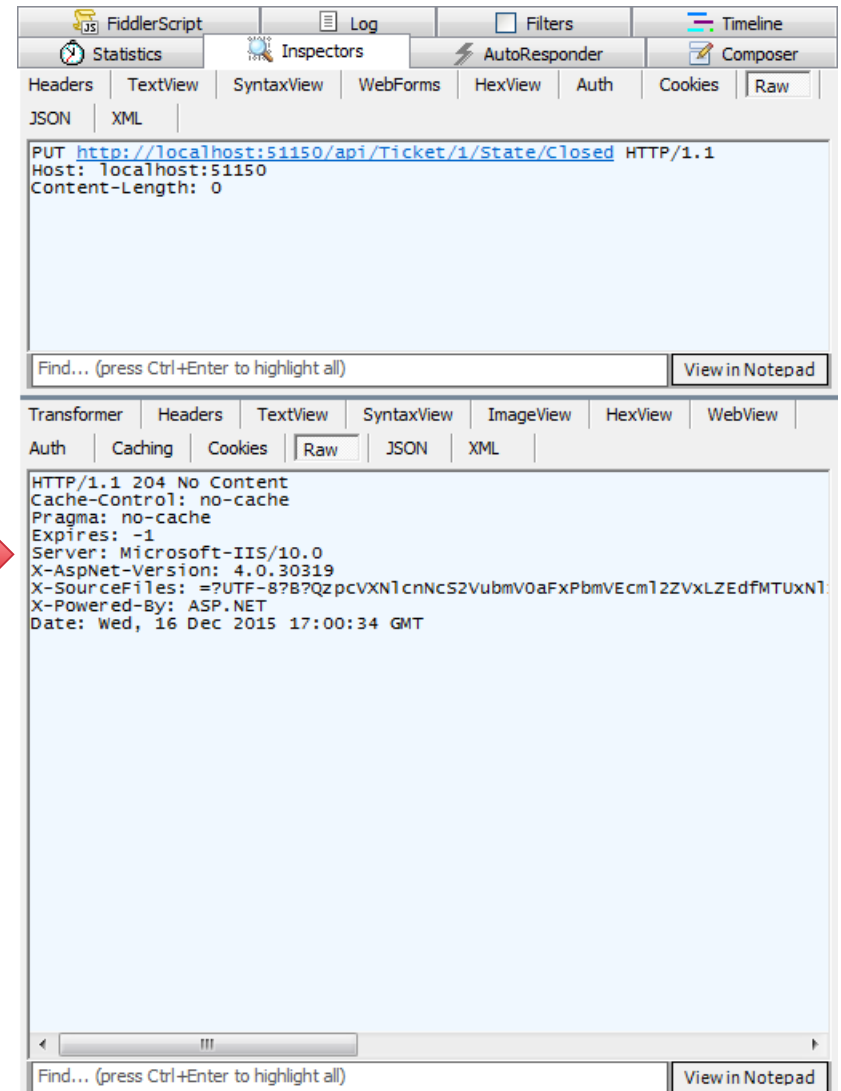
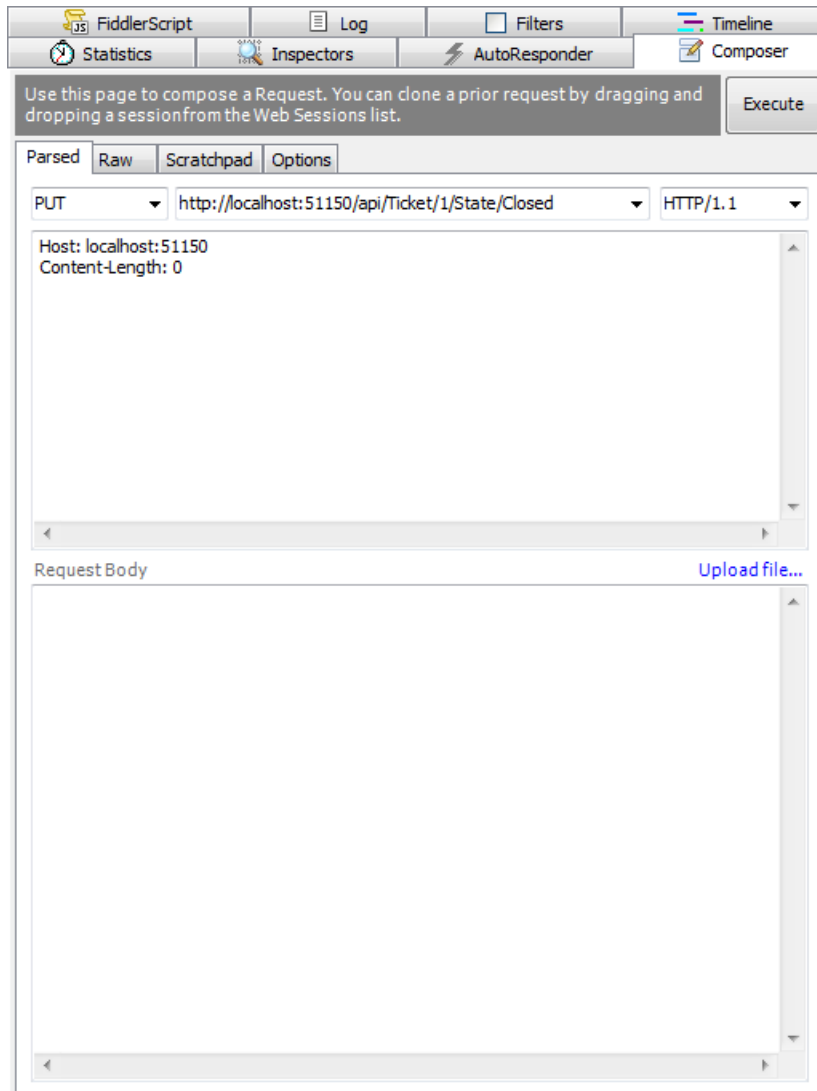

TicketController.cs

```
...
using SC.BL;
using SC.BL.Domain;
namespace SC.UI.Web.MVC.Controllers.Api
{
    public class TicketController : ApiController
    {
        private ITicketManager mgr = new TicketManager();

        ...

        // PUT: api/Ticket/5
        [HttpPut]
        [Route("api/Ticket/{id}/State/Closed")]
        public IHttpActionResult PutTicketStateToClosed(int id)
        {
            mgr.ChangeTicketStateToClosed(id);
            return StatusCode(HttpStatusCode.NoContent);
        }
    }
}
```

Fiddler



Oefening (vervolg)

- Stap 2: Front-end
 - span-element rond status-waarde (id = status)
 - button-element
 - class = "btn btn-default btn-xs"
 - onclick-event: roept 'closeTicket' aan met als parameter het ticketnummer van het huidige ticket
 - enkel voorzien indien de status niet 'Closed' is
- Stap 3: Front-end-logica (JavaScript)
 - Voorzie volgende javascript-functies
 - loadResponsesOfTicket(ticketNumber)
 - maak een ajax-call naar de back-end om de status van het ticket met binnenkomende 'ticketNumber' naar de status 'Closed' te wijzigen
 - » bij een goede response wijzig je de tekst van het element met id 'state' naar 'Closed'
 - » bij een fout toon je adhv een alert de boodschap "Oeps, something went wrong!"

\Views\Ticket\Details.cs

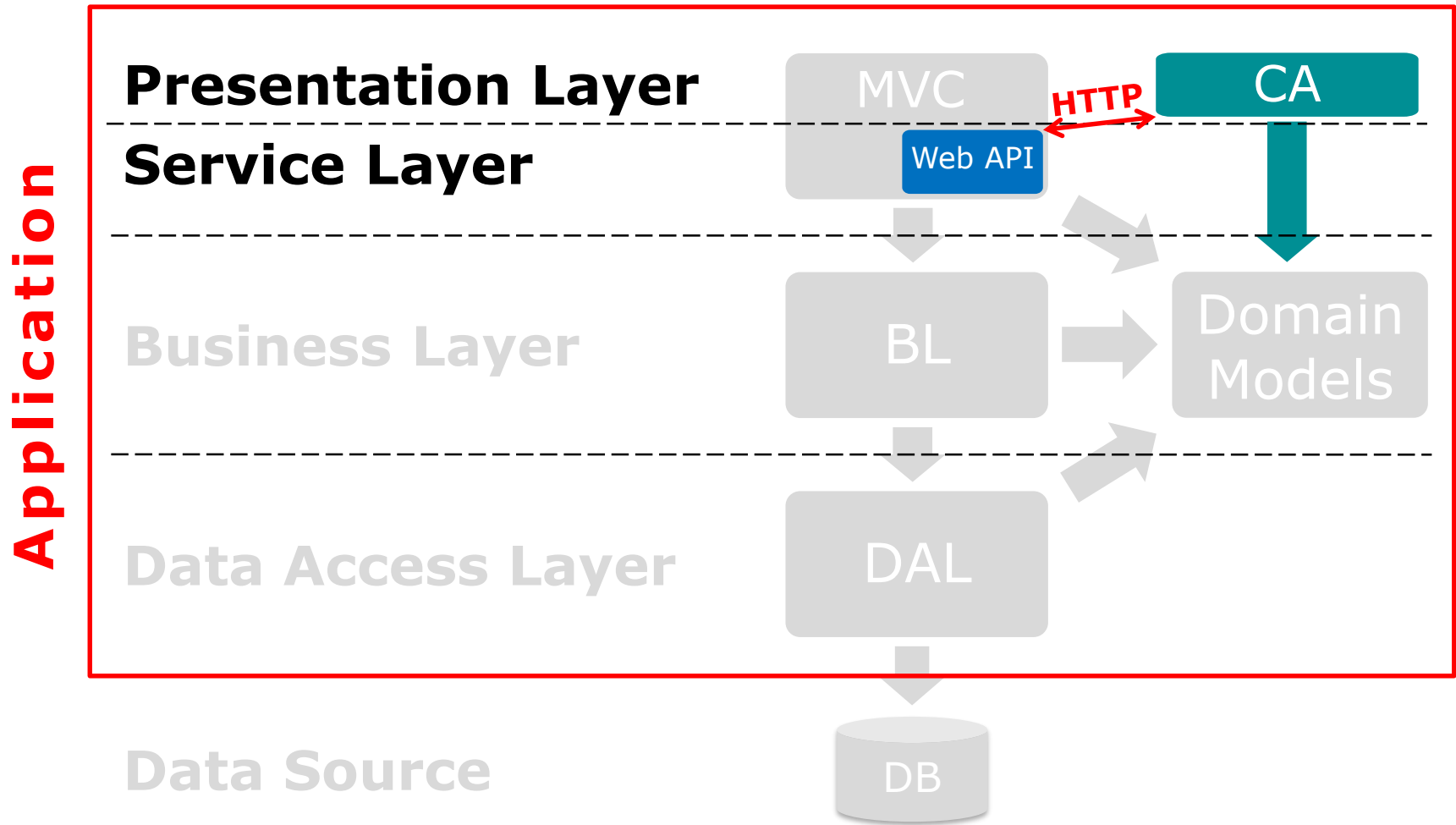
```
...
<dd>
    <span id="state">@Html.DisplayFor(model => model.State)</span>
    @if (Model.State != SC.BL.Domain.TicketState.Closed)
    {
        <button type="button" class="btn btn-default btn-xs"
            onclick="closeTicket(@Model.TicketNumber)">Close</button>
    }
</dd>
...
```

```
@section scripts {
    <script type="text/javascript">
        function closeTicket(ticketNumber) {
            $.ajax('/api/Ticket/' + ticketNumber + '/State/Closed', {
                type: 'PUT'
            })
            .done(function (data) { $('#state').html('Closed'); })
            .fail(function () { alert('Oeps, something went wrong!'); });
        }
    </script>
}
```

CA & Web API

HTTP-calls
(via HttpClient)

CA & Web API architectuur



Oefening

- Voeg aan het project 'UI-CA' een (proxy)klasse 'Service' toe die de communicatie met de back-end (Web API-service) verzorgt
 - voorzie een private constante 'baseUri' (string) met de url van de service
bv.: `http://localhost:xxxxx/api/`
 - dit vind je terug bij de eigenschappen van het project 'UI-MVC' onder 'Web > Servers > Project Url' aangevuld met de default routing prefix 'api' voor Web API-controllers
- OPGELET
Indien je de http-traffic tussen de console applicatie en de webapi-service wil kunnen bekijken met Fiddler moet je in de url 'localhost' vervangen door 'localhost.fiddler'

Properties 'UI-MVC'

UI-MVC

Application
Build
Web
Package/Publish Web
Package/Publish SQL
Silverlight Applications
Build Events
Resources
Settings
Reference Paths
Signing
Code Analysis

Configuration: N/A Platform: N/A

Start Action

☒ Current Page
☐ Specific Page
☐ Start external program
Command line arguments
Working directory
☐ Start URL
☐ Don't open a page. Wait for a request from an external application.

Servers

☒ Apply server settings to all users (store in project file)
IIS Express
Project Url: http://localhost:51150/ Create Virtual Directory
☐ Override application root URL
http://localhost:51150/

Debuggers

☒ ASP.NET ☐ Native Code
☒ Enable Edit and Continue

```
using System.Web.Http;

namespace SC.UI.Web.MVC
{
    public static class WebApiConfig
    {
        public static void Register(HttpConfiguration config)
        {
            config.Routes.MapHttpRoute(
                name: "DefaultApi",
                routeTemplate: "api/{controller}/{id}",
                defaults: new { id = RouteParameter.Optional }
            );
        }
    }
}
```


Service.cs

...

```
namespace SC.UI.CA
```

```
{  
    internal class Service  
    {  
        //private const string baseUri = "http://localhost:51150/api/";  
        // Use this when using fiddler to capture traffic!  
        private const string baseUri = "http://localhost.fiddler:51150/api/";  
  
    }  
}
```

Oefening

- Zorg er voor dat in 'Program' de antwoorden van een ticket via de service opgehaald worden ipv de manager aan te spreken
 - Klasse 'Service'
 - voorzie een methode 'GetTicketResponses' met dezelfde signature als die van de manager
 - maak een http-call naar de back-end (GET) om de ticketresponse-data op te halen adhv het binnenkomende 'ticketNumber'
 - » vraag om de response in json-format te sturen
 - » bij een goede response 'deserialize' de binnenkomende json-string naar een verzameling van 'TicketResponse'-objecten
 - » bij een fout gooi je een exception met als bericht "{response-statuscode} {response-statusbericht}"

Oefening (vervolg)

- Klasse 'Program'
 - voorzie een privaat veld 'srv' (type 'Service')
 - pas de implementatie van het menu-item 'Toon de antwoorden van een ticket' aan zodat er gebruik gemaakt wordt van de service ipv de manager
 - er zou maar één regel gewijzigd moeten worden!
- OPGELET
 - Stel via de eigenschappen van de solution beide projecten 'UI-CA' en 'UI-MVC' in als startup-projecten
 - Test de uitwerking! Let op: indien je localhost.fiddler gebruikt moet Fiddler actief zijn (capturing!)

Service.cs

```
...
using System.Net.Http;
using Newtonsoft.Json;
using SC.BL.Domain;
namespace SC.UI.CA
{
    internal class Service
    {
        ...

        public IEnumerable<TicketResponse> GetTicketResponses(int ticketNumber)
        {
            IEnumerable<TicketResponse> responses = null;
            using (HttpClient http = new HttpClient())
            {
                string uri = baseUri + "TicketResponse?ticketNumber=" + ticketNumber;
                HttpRequestMessage httpRequest = new HttpRequestMessage(HttpMethod.Get, uri);
                //Verwachte content-type van de response meegeven
                httpRequest.Headers.Add("Accept", "application/json");
                //Request versturen en wachten op de response
                HttpResponseMessage httpResponse = http.SendAsync(httpRequest).Result;
                if (httpResponse.IsSuccessStatusCode)
                {
                    //Body van de response uitlezen als een string
                    string responseContentAsString = httpResponse.Content.ReadAsStringAsync().Result;
                    //Body-string (in json-format) deserialiseren (omzetten) naar een verzameling van TicketResponse-objecten
                    responses = JsonConvert.DeserializeObject<List<TicketResponse>>(responseContentAsString);
                }
                else
                {
                    throw new Exception(httpResponse.StatusCode + " " + httpResponse.ReasonPhrase);
                }
            }
            return responses;
        }
    }
}
```

Vergeet niet om aan het project 'UI-CA':

- een reference naar 'System.Net.http' toe te voegen
- het NuGet-package 'Newtonsoft.Json' toe te voegen

Program.cs

```
...
namespace SC.UI.CA
{
    class Program
    {
        private static bool quit = false;
        private static readonly ITicketManager mgr = new TicketManager();
        private static readonly Service srv = new Service();

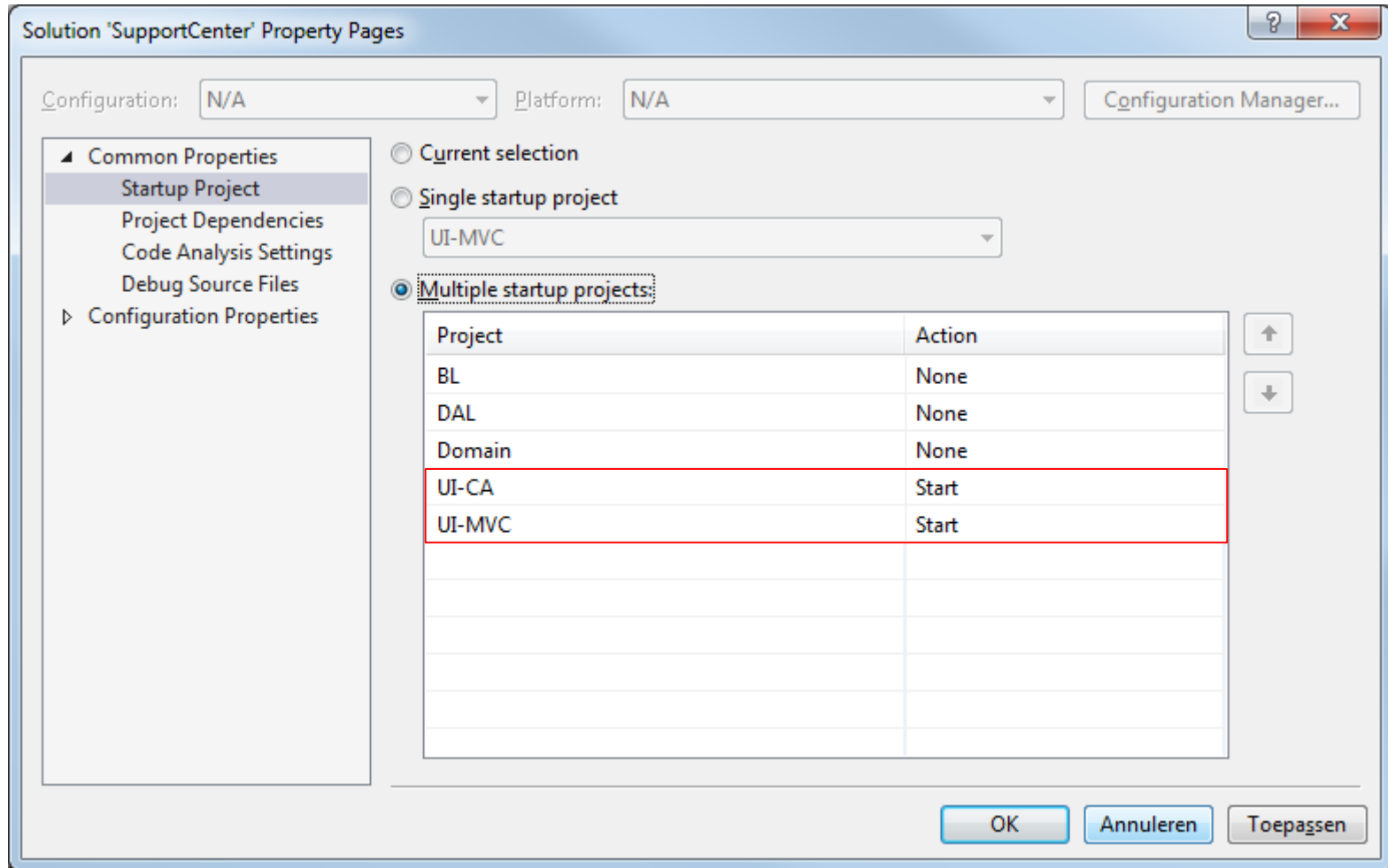
        ...

        private static void ActionShowTicketResponses()
        {
            Console.Write("Ticketnummer: ");
            int input = Int32.Parse(Console.ReadLine());

            //IEnumerable<TicketResponse> responses = mgr.GetTicketResponses(input);
            // via Web API-service
            IEnumerable<TicketResponse> responses = srv.GetTicketResponses(input);
            if (responses != null) PrintTicketResponses(responses);
        }

        ...
    }
}
```

Properties solution



Oefening

- Zorg er voor dat in 'Program' het toevoegen van een antwoord aan een ticket via de service gebeurt ipv de manager
 - Klasse 'Service'
 - voorzie een methode 'AddTicketResponse' met dezelfde signature als die van de manager
 - maak een http-call naar de back-end (POST)
 - » geef de nodige data mee die via de parameters binnenkomen
 - gebruik hiervoor een anonymous object dat je 'serialized' naar json-formaat
 - » vraag om de response in json-formaat te sturen
 - » bij een goede response 'deserialize' de binnenkomende json-string naar een 'TicketResponse'-object
 - » bij een fout gooi je een exception met als bericht "{response-statuscode} {response-statusbericht}"
 - Klasse 'Program'
 - wijzig de implementatie van menu-item 'Geef een antwoord op een ticket' zodat de service gebruikt wordt

Service.cs

```
...
internal class Service
{
    ...
    public TicketResponse AddTicketResponse(int ticketNumber, string response, bool isClientResponse)
    {
        TicketResponse tr = null;

        using (HttpClient http = new HttpClient())
        {
            string uri = baseUrl + "TicketResponse";
            HttpRequestMessage httpRequest = new HttpRequestMessage(HttpMethod.Post, uri);
            //Request data toevoegen aan body, via anonymous object dat je serialiseert naar json-formaat
            object data = new { TicketNumber = ticketNumber, ResponseText = response, IsClientResponse = isClientResponse };
            string dataAsJsonString = JsonConvert.SerializeObject(data);
            httpRequest.Content = new StringContent(dataAsJsonString, Encoding.UTF8, "application/json");
            //Verwachte content-type van de response meegeven
            httpRequest.Headers.Add("Accept", "application/json");
            //Request versturen en wachten op de response
            HttpResponseMessage httpResponse = http.SendAsync(httpRequest).Result;
            if (httpResponse.IsSuccessStatusCode)
            {
                //Body van de response uitlezen als een string
                string responseContentAsString = httpResponse.Content.ReadAsStringAsync().Result;
                //Body-string (in json-format) deserialiseren (omzetten) naar een TicketResponse-object
                tr = JsonConvert.DeserializeObject<TicketResponse>(responseContentAsString);
            }
            else
            {
                throw new Exception(httpResponse.StatusCode + " " + httpResponse.ReasonPhrase);
            }
        }

        return tr;
    }
}
```


Program.cs

```
...
namespace SC.UI.CA
{
    class Program
    {
        ...

        private static void ActionAddResponseToTicket()
        {
            Console.Write("Ticketnummer: ");
            int ticketNumber = Int32.Parse(Console.ReadLine());
            Console.Write("Antwoord: ");
            string response = Console.ReadLine();

            //mgr.AddTicketResponse(ticketNumber, response, false);
            // via WebAPI-service
            srv.AddTicketResponse(ticketNumber, response, false);
        }

        ...
    }
}
```

Oefening

- Zorg er voor dat in 'Program' het wijzigen van de status van ticket naar 'Closed' via de service gebeurt ipv de manager
 - Klasse 'Service'
 - voorzie een methode 'ChangeTicketStateToClosed' met dezelfde signature als die van de manager
 - maak een ajax-call naar de back-end om de status van het ticket met binnenkomende 'ticketNumber' naar de status 'Closed' te wijzigen
 - » bij een fout gooi je een exception met als bericht "{response-statuscode} {response-statusbericht}"
 - Klasse 'Program'
 - wijzig de implementatie van menu-item 'Markeer ticket als 'Closed' zodat de service gebruikt wordt

Service.cs

```
...
internal class Service
{
    ...
    public void ChangeTicketStateToClosed(int ticketNumber)
    {
        using (HttpClient http = new HttpClient())
        {
            string uri = baseUri + "Ticket/" + ticketNumber + "/State/Closed";
            HttpRequestMessage httpRequest = new HttpRequestMessage(HttpMethod.Put, uri);
            //Request versturen en wachten op de response
            HttpResponseMessage httpResponse = http.SendAsync(httpRequest).Result;
            if (!httpResponse.IsSuccessStatusCode)
            {
                throw new Exception(httpResponse.StatusCode + " " + httpResponse.ReasonPhrase);
            }
        }
    }
}
```

Program.cs

```
...
namespace SC.UI.CA
{
    class Program
    {
        ...

        private static void ActionCloseTicket()
        {
            Console.Write("Ticketnummer: ");
            int input = Int32.Parse(Console.ReadLine());

            //mgr.ChangeTicketStateToClosed(input);
            // via WebAPI-service
            srv.ChangeTicketStateToClosed(input);
        }

        ...
    }
}
```