HUMAN CAPITAL
NATIONAL COHESION STRATEGY

EUROPEAN UNION
EUROPEAN
SOCIAL FUND

AJAX-ready server scripts

Michał Okulewicz
Warsaw University of Technology
Faculty of Mathematics and Information Science
M.Okulewicz@mini.pw.edu.pl
http://www.mini.pw.edu.pl/~okulewiczm

AJAX-ready server scripts

- In order to generate proper responses to AJAX calls server script should generate only a part of the website
- It may also generate a more general XML or JSON content instead of an HTML matching only one website
- While XML has been initially used as a response for an AJAX call it is better to generate a part of HTML or a JSON object

AJAX-ready server scripts: HTML

- A following WebForm (in general template scripts like PHP or JSP) generates HTML table, which can be loaded in a prepared website

```
<%@ Page … ContentType=”text/html” %>
<table>
<% { for (int i = 0; i < 10; ++i) %>
   <tr><td><%= i + 1 %></td></tr>
<% } %>
</table>
```

AJAX-ready server scripts: JSON

- A following WebForm (in general template scripts like PHP or JSP) generates JSON content

```
<%@ Page … ContentType=”application/json” %>
{“ids”:
<% { for (int i = 0; i < 10; ++i) %>
   “id” : <%= i + 1 %>,
<% } %>}
```

- More complicated content would be probably even more unclear
- One should apply some form of JSON generating web service

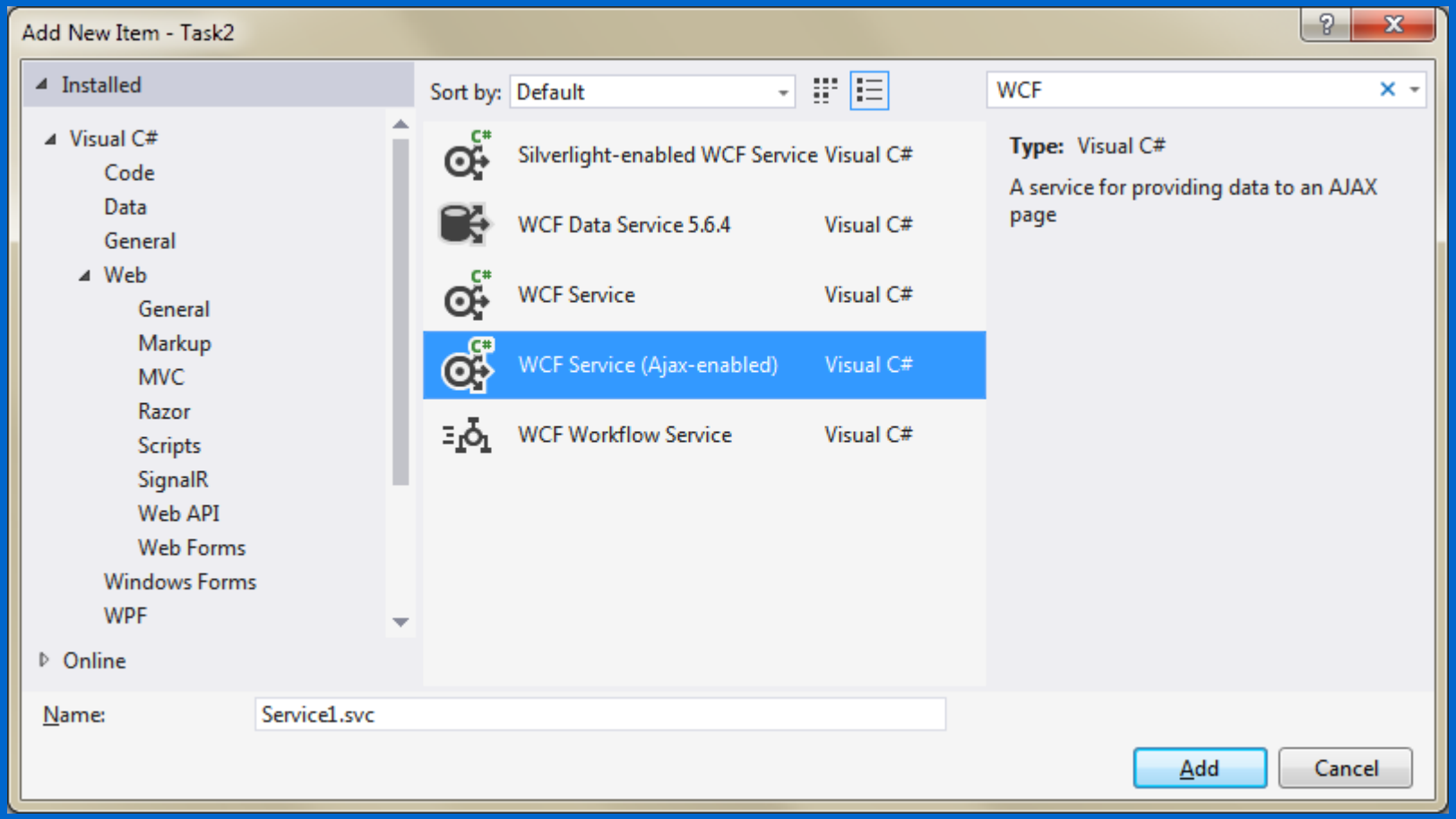
Microsoft .NET JSON services

- .NET framework offers two methods of creating a web service automatically serializing and deserializing JSON/.NET objects
  - AJAX enabled Windows Communication Foundation (older)
  - Web API (v.2) controllers (newer)
    - available in .NET Core
    - RESTful

AJAX-enabled WCF service

- After adding an AJAX-enabled service
  - A `<system.serviceModel>` element will appear in Web.config file
  - WCF service would have a webHttpBinding endpoint
  - WCF service would have a ServiceHost markup
  - WCF service would be compatible with ASP.NET
- WCF services would be by default available as the URL of the form `/[ClassName].svc/MethodName`

AJAX-enabled WCF service



AJAX-enabled WCF service

```
<?xml version="1.0"?>
<configuration>
    <system.web>
      <compilation debug="true" targetFramework="4.5" />
      <httpRuntime targetFramework="4.5" />
    </system.web>

    <system.serviceModel>
        <behaviors>
            <endpointBehaviors>
                <behavior name="Task3.DataGeneratorAspNetAjaxBehavior">
                    <enableWebScript />
                </behavior>
            </endpointBehaviors>
        </behaviors>
        <serviceHostingEnvironment aspNetCompatibilityEnabled="true"
            multipleSiteBindingsEnabled="true" />
        <services>
            <service name="Task3.DataGenerator">
                <endpoint address="" behaviorConfiguration="Task3.DataGeneratorAspNetAjaxBehavior"
                    binding="webHttpBinding" contract="Task3.DataGenerator" />
            </service>
        </services>
    </system.serviceModel>
</configuration>
```

Configuration of methods

- All exposed methods must have an `OperationContract` attribute
- To configure methods as a REST style services one should configure them by specifying `RequestFormat`, `ResponseFormat` in `WebInvoke` attribute
- To configure methods as a GET HTTP accessible contents one should configure them in a `WebGet` attribute

Configuration of methods

```
    [ServiceContract(Namespace = "")]
    [AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
    public class DataGenerator
    {
        // To use HTTP GET, add [WebGet] attribute. (Default ResponseFormat is
WebMessageFormat.Json)
        // To create an operation that returns XML,
        //     add [WebGet(ResponseFormat=WebMessageFormat.Xml)],
        //     and include the following line in the operation body:
        //         WebOperationContext.Current.OutgoingResponse.ContentType = "text/xml";
        [OperationContract]
        [WebInvoke(
                RequestFormat = WebMessageFormat.Json,
                ResponseFormat = WebMessageFormat.Json, Method = "POST",
                BodyStyle = WebMessageBodyStyle.WrappedRequest)]
        public Data[] GetData()
        {
        ...
        }
    }
```

Accessing Session and Cookies

- In the constructor and other methods of a WCF service object a `System.Web.HttpContext` object is available
- `HttpContext` object exposes `Request`, `Response` and `Session` objects allowing for managing of the web application in the way identical to WebForms

REST services

- REpresentational State Transfer
- **Architectural pattern** for creating an API that uses HTTP as its underlying communication method (Roy Fielding, 2000) http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm
- The philosophy of using a REST service is to access/modify a given RESOURCE (NOT to perform an ACTION as in calling a method)
- What should be done with a resource is specified by a request verb (an HTTP method), with the following SQL equivalents
  - GET => SELECT
  - POST => CREATE
  - PUT => UPDATE
  - DELETE => DELETE

REST services cont.

- Features of a RESTful service
  - Stateless
    - All the necessary data are passed in the request
  - Cacheable
    - Resource defines its availability for being cached
  - Client-server
    - UI separation from data storage
  - Layered system
    - Client does not need to know about proxies and load-balancers
  - Code on Demand (optional)
    - Server might provide client with the application logic (eg. Java Applet, JavaScript library)

REST services requests examples

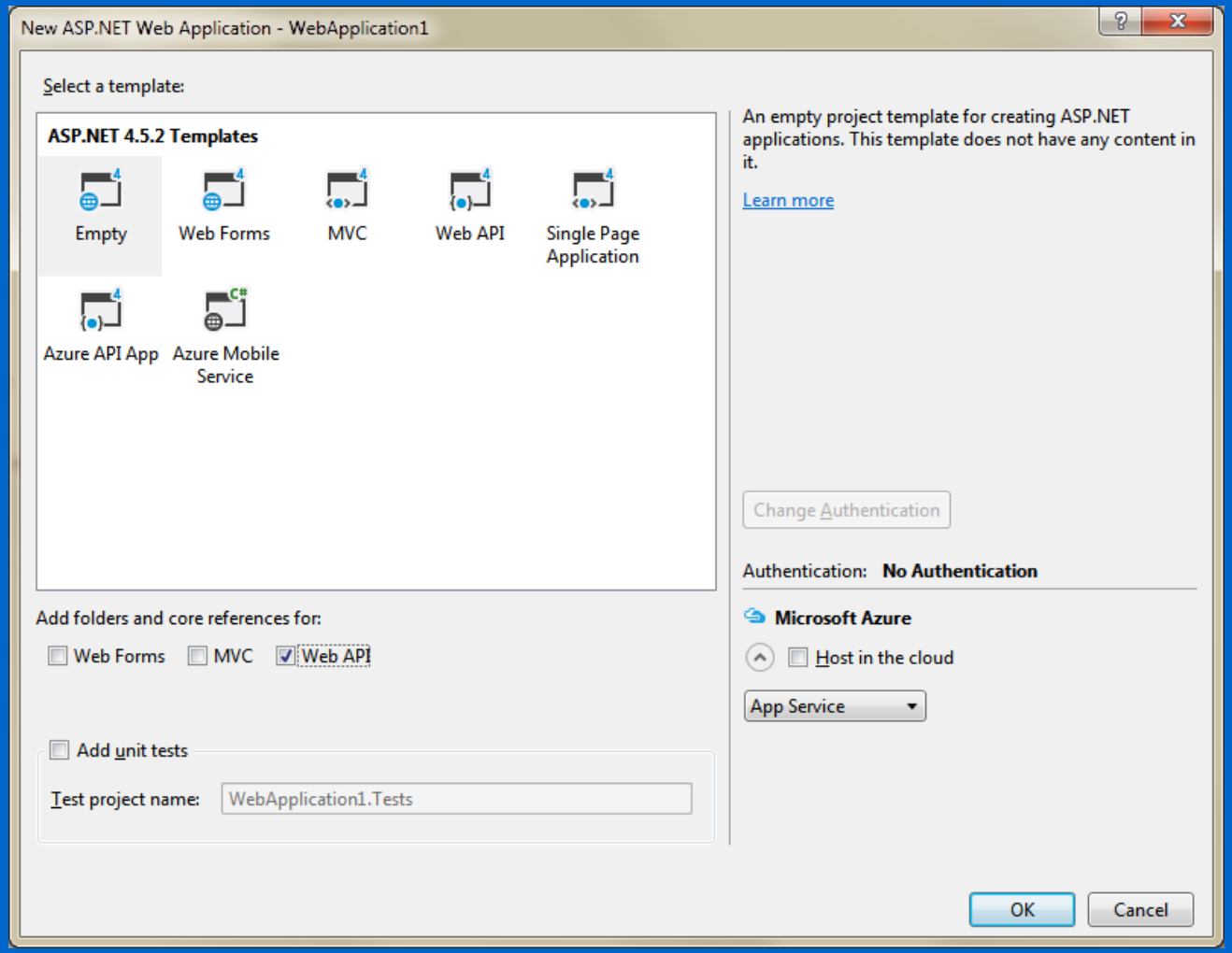
Resource	Verb	Expected Outcome	Response Code
/Products	GET	A list of all products in the system	200/OK
/Products?Colour=red	GET	A list of all products in the system where the colour is red	200/OK
/Products	POST	Creation of a new product	201/Created
/Products/81	GET	Product with ID of 81	200/OK
/Products/881 (a product ID which does not exist)	GET	Some error message	404/Not Found
/Products/81	PUT	An update to the product with an ID of 81	204/No Content
/Products/81	DELETE	Deletion of the product with an ID of 81	204/No Content

Source: https://blogs.msdn.microsoft.com/martinkearn/2015/01/05/introduction-to-rest-and-net-web-api/

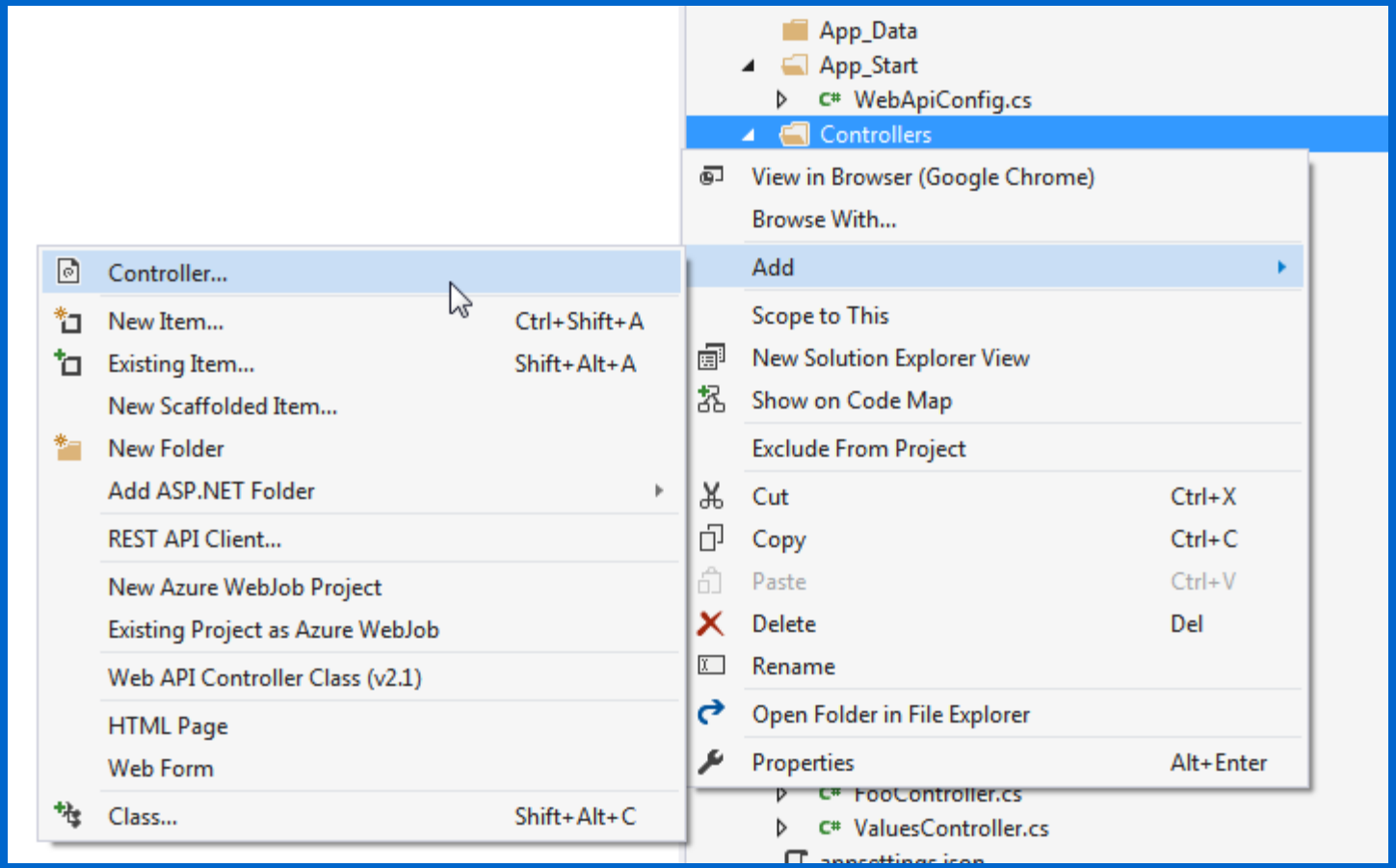
MS .NET Web API

- Create an ASP.NET Web Project with WebAPI folders and references
- Create a class extending an `ApiController`
- Name it `SomethingController`
- By default a `localhost:port/api/something` `URIs` will be bound to the controller's methods
- https://www.asp.net/web-api/overview/getting-started-with-aspnet-web-api/tutorial-your-first-web-api
- https://www.asp.net/web-api/overview/web-api-routing-and-actions/routing-in-aspnet-web-api

MS .NET Web API



Adding controller



Adding controller