

ZPR PWr – Zintegrowany Program Rozwoju Politechniki Wrocławskiej

Aplikacje webowe na platformę .NET

W13 – API kontroler, Ajax, Blazor

Syllabus

- Kotroler API
 - ReST , ReSTful
 - Metody HTTP i adresy w typie ReSTful
 - Kontroler API
 - curl
 - Narzędzie Postman
 - Narzędzie Swagger
- Ajax - kod do asynchronicznego ładowania danych z kontrolera API:
 - JavaScript
 - jQuery
- Blazor
 - Projekt typu Blazor server App
 - Operacje CRUD na kolekcji obiektów klasy `Student`
 - Analiza komunikacji w przeglądarce
 - Żądania HTTP
 - Komunikacja poprzez websocket
- Projekt typu WebAssembly App

Podejście ReST i RESTful

- REST, ReST - **R**epresentational **S**tate **T**ransfer
 - Zaprojektowane wokół zasobów (a nie operacji)
 - URI (**U**niform **R**esource **I**dentifier) definiujące konkretny zasób (zob. następny slajd)
 - Klient komunikuje się z serwerem wymieniając *reprezentację* danych (JSON).
 - Zunifikowany interfejs używający czasowników z metod HTTP: GET, POST, PUT, PATCH oraz DELETE.
 - Bezstanowość jak w HTTP
 - Jest sterowane przez hiperlinki w reprezentacji (poniżej)
- Uznaje się 4 stopnie we wprowadzaniu REST API:
 - 1) zdefiniuj jedno URI i użyj do wszystkiego żądań POST do tego URI.
 - 2) Stwórz oddzielne URI dla każdego zasobu.
 - 3) Użyj metod HTTP do zdefiniowania operacji na zasobach.
 - 4) użyj hipermediów HATEOAS (Hypertext as the Engine of Application State)
- RESTful oznacza użycie 4 stopnia.

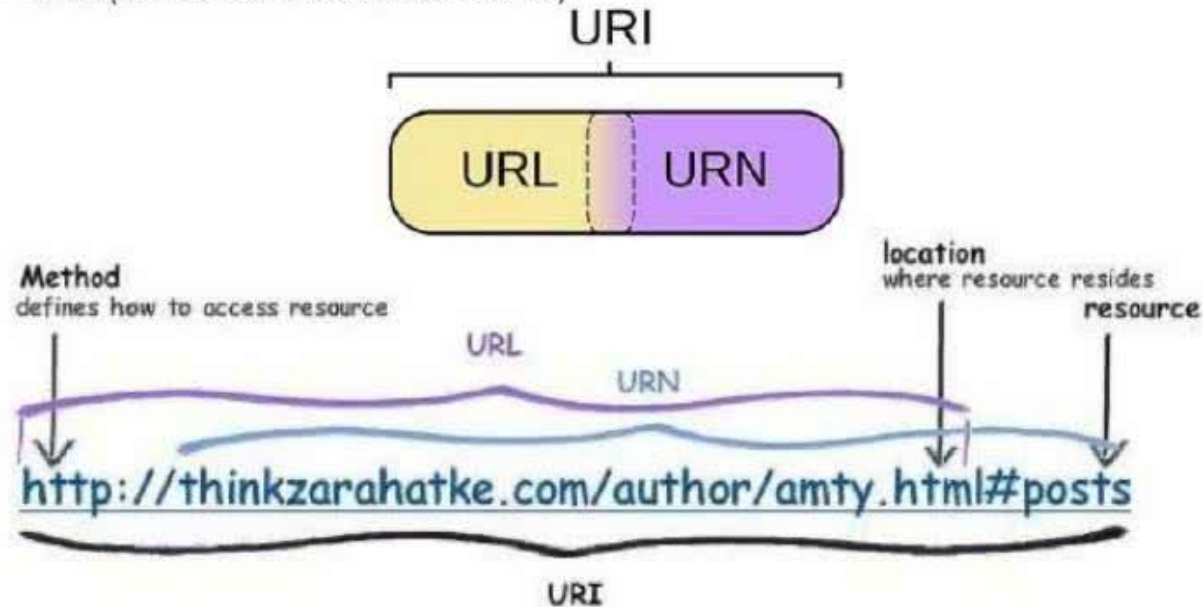
```
{
  "account": {
    "account_number": 12345,
    "balance": {
      "currency": "usd",
      "value": 100.00
    },
    "links": {
      "deposits": "/accounts/12345/deposits",
      "withdrawals": "/accounts/12345/withdrawals",
      "transfers": "/accounts/12345/transfers",
      "close-requests": "/accounts/12345/close-requests"
    }
  }
}
```

URI,URL,URN

- Różnice między URI,URL,URN
- Źródło:<https://ep.com.pl/novosci/koktajl-newsow/13819-adres-strony-internetowej-to-nie-url-glupcze>

URI(2)

- URL(Universal Resource Locator)
- URN(Universal Resource Name)



© Copyright 2013 iSecure Co., Ltd. The information contained herein is subject to change without notice.

Metody HTTP i adresy w typie RESTful

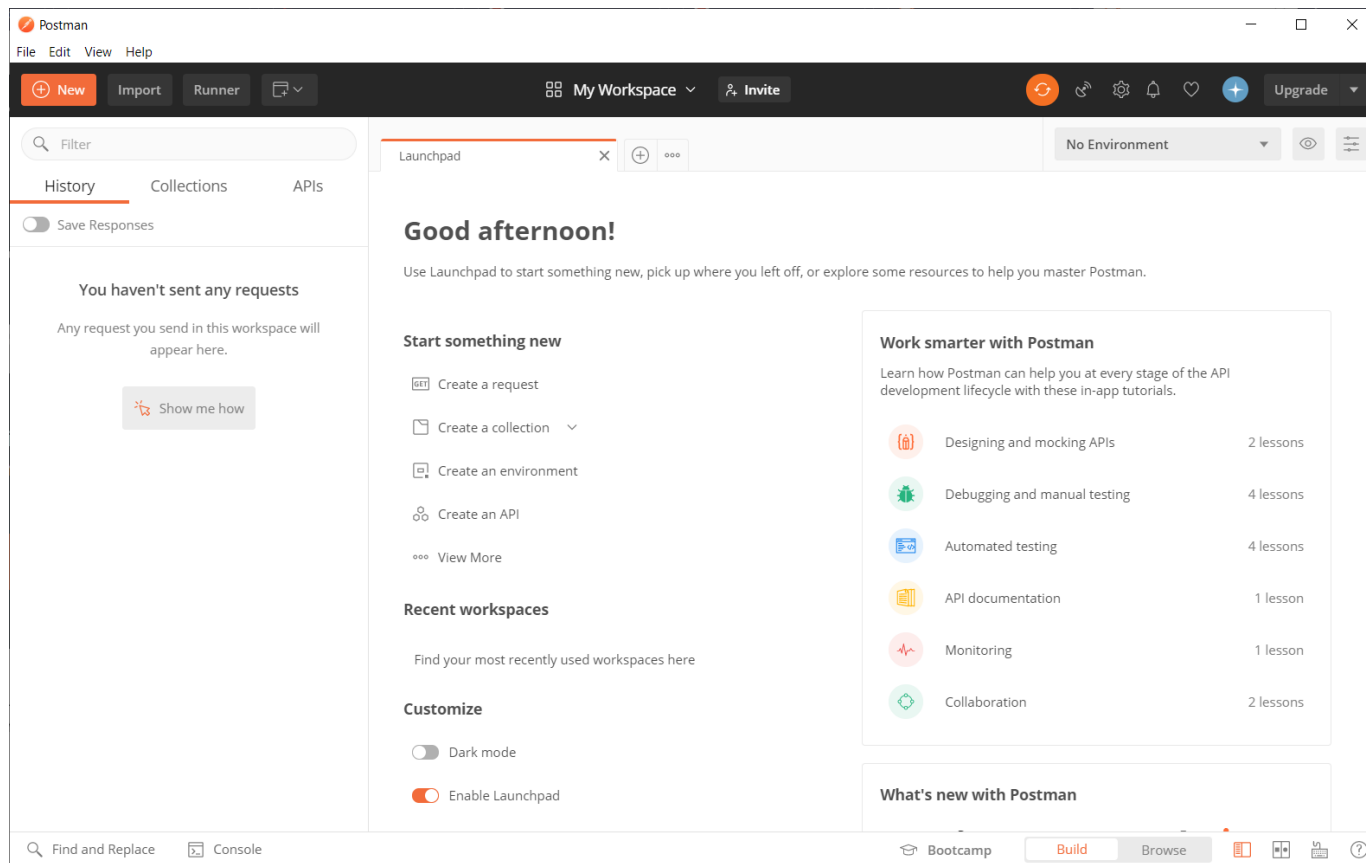
Metoda HTTP	Adres URL	Opis
GET	/api/student	Pobranie wszystkich obiektów Student
GET	/api/student/1	Pobranie obiektu Student z id równym 1
POST	/api/student	Utworzenie nowego obiektu Student
PUT	/api/student	Zastąpienie istniejącego obiektu
PATCH	/api/student/1	Modyfikacja obiektu Student o id równym 1
DELETE	/api/student/1	Usunięcie obiektu Student o id równym 1

Ładunek w żądaniach API

- Ładunek (ang. payload) to zawartość ciała (ang. body) w żądaniu i odpowiedzi
- Może być w różnych formatach
 - W wykładzie użyty zostanie JSON
 - Kiedyś był to XML, ale jest zbyt rozwlekły
- Obecność ładunku wynika dość logicznie z operacji:
 - GET – brak w żądaniu, **w odpowiedzi** jeden lub wszystkie zasoby
 - POST – **w żądaniu**: minimum wymagane pola zasobu, **w odpowiedzi**: cały utworzony zasób z magazynu danych
 - PUT: **w żądaniu**: zasób do utworzenia/zastąpienia , **odpowiedź** jak w POST
 - PATCH: **w żądaniu**: pola wymagające zmiany, odpowiedź: potwierdzenie lub nie zmian
 - DELETE: brak ładunku w żądaniu i odpowiedzi

Postman – narzędzie do testowania

- <https://www.postman.com/downloads/>
 - Windows
 - MacOS
 - Linux
- Należy stworzyć konto lub użyć konta Google



Postman – użycie 1/2

- Wysyłanie żądania HTTP w wybranym formacie
- Odbiór wyniku żądania
- Wybrać „Create a request”

Automatyczna zamiana w obydwie strony

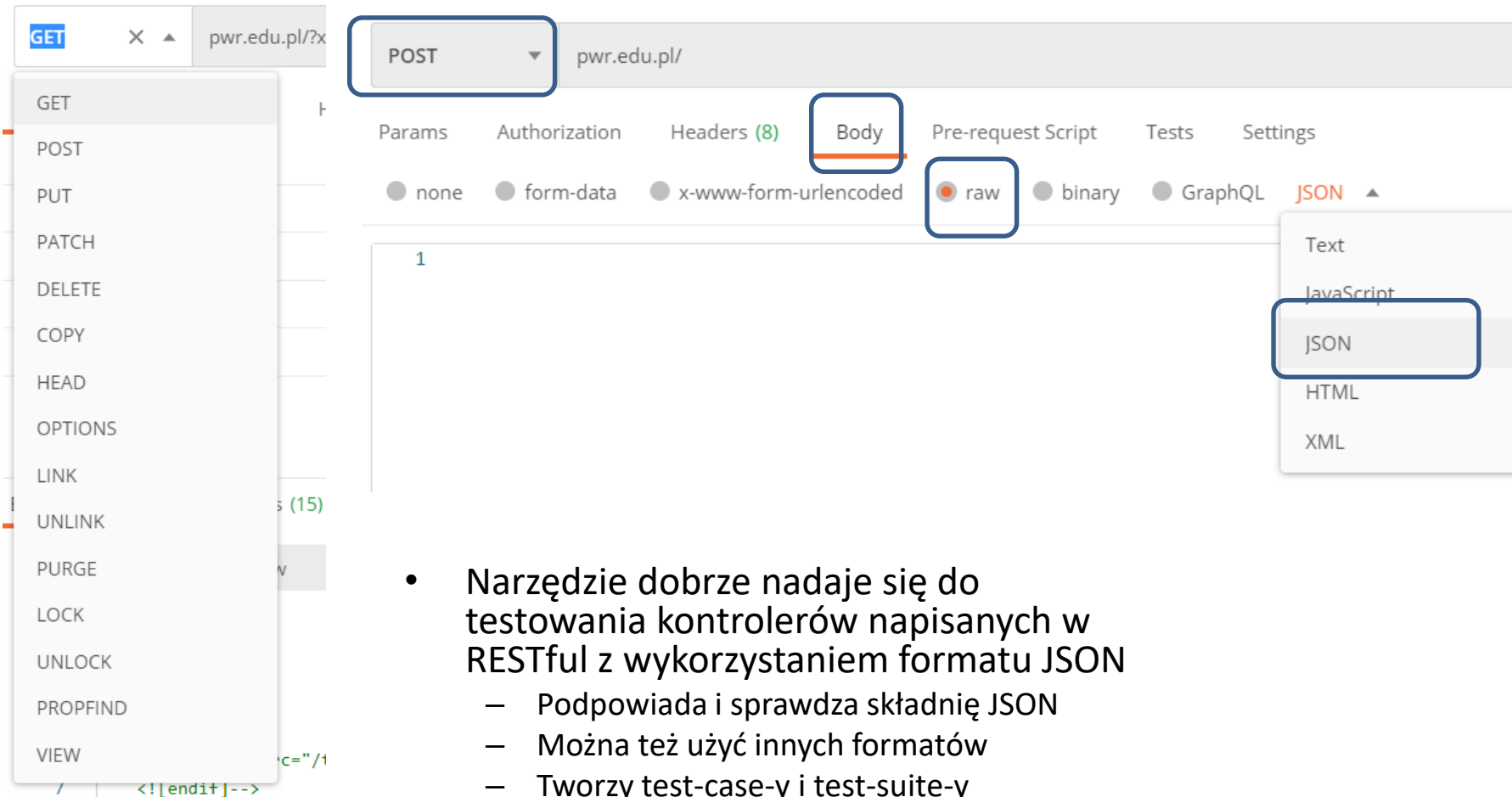
The screenshot shows the Postman interface with a GET request to `pwr.edu.pl/?x=4&Name=Dariusz`. The `Params` tab is selected, displaying the following query parameters:

KEY	VALUE	DESCRIPTION
x	4	
Name	Dariusz	

The response is shown in the bottom section, indicating a `Status: 200 OK` with a `Time: 3.40 s` and `Size: 192.18 KB`. The response body is displayed in `HTML` format, showing the following code:

```
1 <!DOCTYPE html>
2 <html lang="pl">
3
4 <head>
5   <!--[if IE]>
6     <script src="/themes/_system/js/jquery-1.12.3.min.js"></script>
7   <![endif]-->
8   <title>Politechnika Wrocławska</title>
9   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```


Postman – użycie 2/2



The screenshot shows the Postman interface for configuring a POST request. The top bar displays the method 'POST' and the URL 'pwr.edu.pl/'. Below this, the 'Body' tab is selected, and the 'raw' radio button is chosen. A dropdown menu on the right indicates the 'JSON' format is selected. The left sidebar shows a list of HTTP methods, with 'GET' at the top.

- Narzędzie dobrze nadaje się do testowania kontrolerów napisanych w RESTful z wykorzystaniem formatu JSON
 - Podpowiada i sprawdza składnię JSON
 - Można też użyć innych formatów
 - Tworzy test-case-y i test-suite-y

Kontroler RESTful – przygotowanie modelu danych

- Klasy dla danych:
 - Klasa `Student`
 - Interfejs `IRepository`
 - Dodanie interfejsu do serwisu
- W projekcie MVC, z wykorzystaniem tylko strony `Index`

```
public class Student
{
    public int Id { get; set; }
    public int Index { get; set; }
    public string Name { get; set; }
}
```

Student.cs

```
public interface IRepository
{
    IEnumerable<Student> Students { get; }
    Student this[int id] { get; }
    Student AddStudent(Student student);
    Student UpdateStudent(Student student);
    void DeleteStudent(int id);
    Student GetNextStudent(int id);
    Student GetPreviousStudent(int id);
}
```

IRepository.cs

Później (dla AJAX-a)

```
public class Startup
{
    ...
    public void ConfigureServices(IServiceCollection services)
    {
        services.AddSingleton<IRepository, MemoryRepository>();
        services.AddControllersWithViews();
    }
    ...
}
```

Startup.cs

Klasa MemoryRepository

- Implementacja interfejsu, MemoryRepository

```
public class MemoryRepository : IRepository    {
    private readonly Dictionary<int, Student> items; // po co musi być readonly?!
    public MemoryRepository()
    {
        items = new Dictionary<int, Student>();
        new List<Student>
        {
            new Student{ Index=11111, Name="Smith"},
            new Student{ Index=22222, Name="Kowal"},
            new Student{ Index=33333, Name="Schneider"}
        }.ForEach(s => AddStudent(s));
    }
    public Student this[int id] => items.ContainsKey(id)?items[id]:null;

    public IEnumerable<Student> Students => items.Values;

    public Student AddStudent(Student student) {
        if (student.Id == 0)
        {
            int key = items.Count;
            while (items.ContainsKey(key)) { key++; };
            student.Id = key;
        }
        items[student.Id] = student;
        return student;
    }

    public void DeleteStudent(int id)=>items.Remove(id);

    public Student UpdateStudent(Student student) => AddStudent(student);
    ...// next lines for AJAX
}
```

MemoryRepository.cs

Sprawdzenie poprzez MVC – Kontroler Home

```
public class HomeController : Controller
{
    private IRepository Repository { get; set; }

    public HomeController(IRepository repo) => Repository = repo;

    public ActionResult Index() => View(Repository.Students);

    [HttpPost]
    public IActionResult AddStudent(Student student)
    {
        Repository.AddStudent(student);
        return RedirectToAction("Index");
    }
}
```

HomeController.cs

- Utworzenie widoku o poniższym wyglądzie:

WebApiController Home Privacy

Index:

Name:

ID	Index	Name
0	11111	Smith
1	22222	Kowal
2	33333	Schneider

Sprawdzenie poprzez MVC – widok Index.cshtml

```
@model IEnumerable<Student>
@{ Layout = "_Layout"; }

<form id="addform" asp-action="AddStudent" method="post">
    <div class="form-group">
        <label for="Index">Index:</label>
        <input class="form-control" name="Index" />
    </div>
    <div class="form-group">
        <label for="Name">Name:</label>
        <input class="form-control" name="Name" />
    </div>
    <div class="text-center panel-body">
        <button type="submit" class="btn btn-sm btn-primary">Dodaj</button>
    </div>
</form>

<table class="table table-sm table-striped table-bordered m-2">
    <thead><tr><th>ID</th><th>Index</th><th>Name</th></tr></thead>
    <tbody>
        @foreach (var r in Model)
        {
            <tr>
                <td>@r.Id</td>
                <td>@r.Index</td>
                <td>@r.Name</td>
            </tr>
        }
    </tbody>
</table>
```

Index.cshtml

- Scenariusz użycia: uruchomienie i dodanie studenta

Kontroler API dla klasy Student 1/2

- Standardowo routing dla kontrolerów Api zaczynający się od segmentu „/api”
- Od Core 2.1 istnieje atrybut ApiControllerAttribute
- Kontroler API dziedziczy po ControllerBase (a nie po Controller)

StudentApiController.cs

```
[EnableCors]
[Route("api/student")]
[ApiController]
public class StudentApiController : ControllerBase
{
    private IRepository repository;

    public StudentController(IRepository repo) => repository = repo;

    [HttpGet]
    public IEnumerable<Student> Get() => repository.Students;

    [HttpGet("{id}")]
    public Student Get(int id) => repository[id];

    [HttpPost]
    public Student Post([FromBody] Student res) =>
        repository.AddStudent(new Student
        {
            Index = res.Index,
            Name = res.Name
        });
}
```

Kontroler API dla klasy Student 2/2

- Dla operacji PATCH trzeba doinstalować `Microsoft.AspNetCore.JsonPatch`
 - Po wpisaniu „`[FromBody] JsonPatchDocument<Student> patch`” VS 2019 sam proponuje instalację.

StudentApiController.cs

```
[HttpPut]
public Student Put([FromBody] Student res) =>
    repository.UpdateStudent(res);

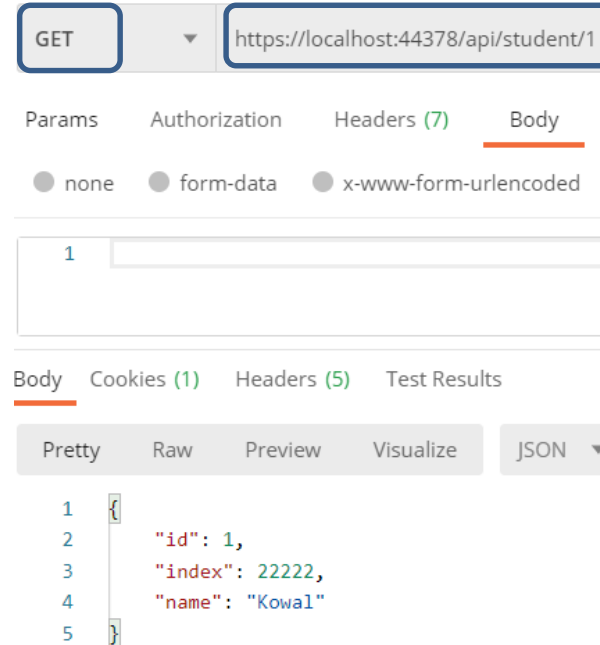
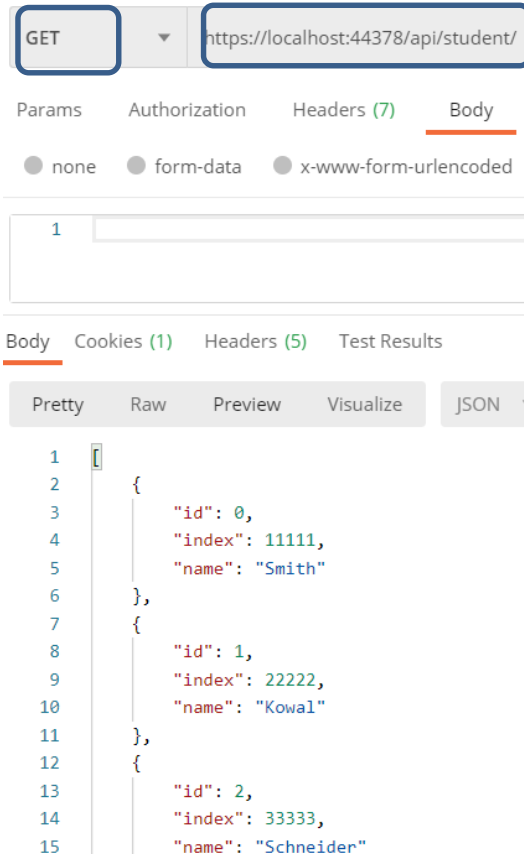
[HttpPatch("{id}")]
public StatusCodeResult Patch(int id,
    [FromBody] JsonPatchDocument<Student> patch)
{
    Student res = Get(id);
    if (res != null)
    {
        patch.ApplyTo(res);
        return Ok();
    }
    return NotFound();
}

[HttpDelete("{id}")]
public void Delete(int id) => repository.DeleteStudent(id);

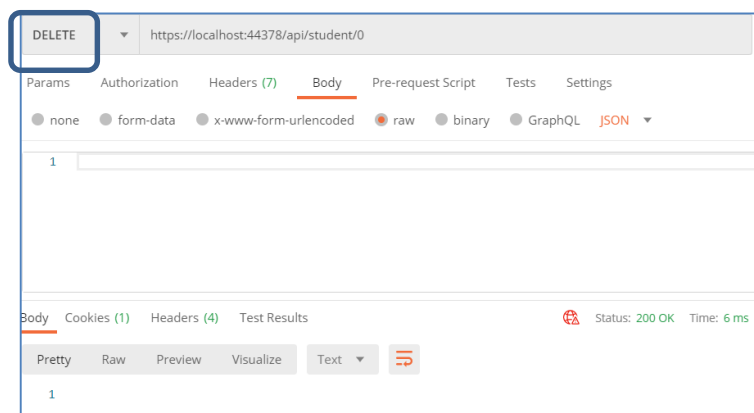
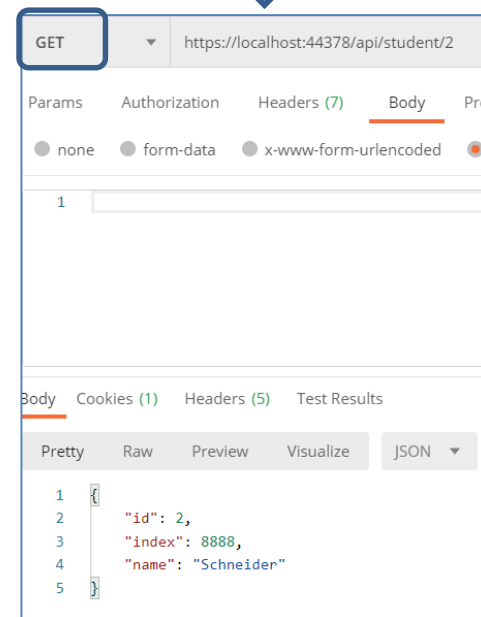
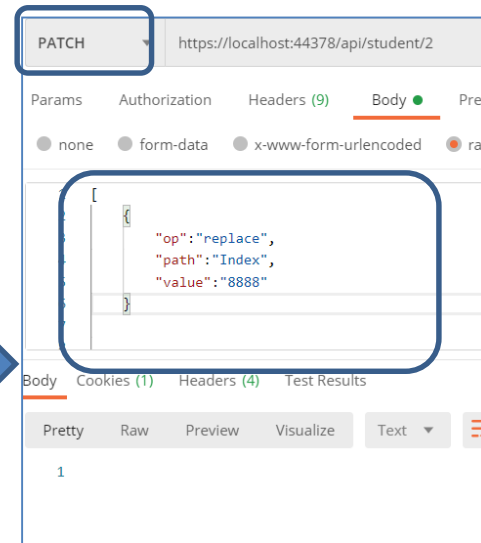
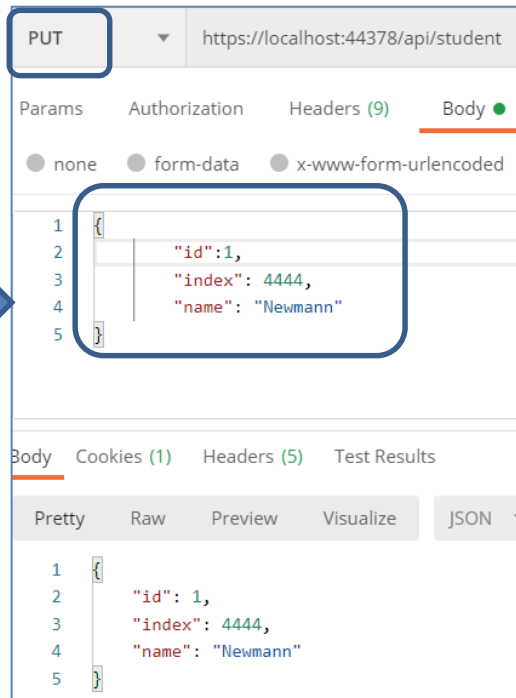
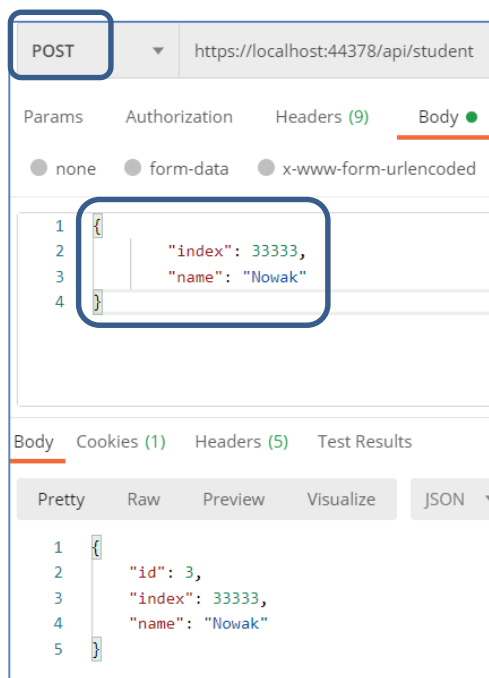
// next lines for AJAX
...
```

Testowanie za pomocą Postmana 1/2

- Scenariusz użycia: wszystkie operacje z kontrolera z użyciem Postman-a



Testowanie za pomocą Postmana 2/2



Operacja PATCH i Core 3.x

- W Core wersji 3.0 operacja PATCH nie działa poprawnie. Aby to poprawić należy doinstalować pakiet `Microsoft.AspNetCore.Mvc.NewtonsoftJson` oraz odpowiednio skonfigurować serwisy (np. jak poniżej).
- Powoduje to zapis i odczyt wszystkich zawartości z JSON przez ten pakiet, również podczas obsługi związanej z pakietem `System.Text.Json`.
 - Jak ograniczyć to tylko do dla operacji Patch w RESTfull:
<https://stackoverflow.com/questions/57914725/how-to-implement-jsonpatch-in-net-core-3-0-preview-9-correctly>
- **Często** operacja Patch **nie jest implementowana** w kontrolerach API.
 - Dla niezbyt rozbudowanych obiektów operacja PUT jest wystarczająca i bardziej przejrzysta

Startup.cs

```
services.AddControllersWithViews()  
    .AddNewtonsoftJson();
```

Komenda curl

Testowanie z linii komend: `curl`

- Bardzo dużo opcji
- Jeśli jest ładunek w zapytaniu najlepiej przygotować plik tekstowy i dołączyć odpowiednią opcją
- Można ustawić/zobaczyć nagłówek żądania/odpowiedzi



```
Wiersz polecenia

C:\Users\darius>curl https://localhost:44378/api/student
[{"id":0,"index":11111,"name":"Smith"}, {"id":1,"index":22222,"name":"Kowal"}, {"id":2,"index":33333,"name":"Schneider"}]

C:\Users\darius>curl https://localhost:44378/api/student/2
{"id":2,"index":33333,"name":"Schneider"}

C:\Users\darius>curl -h
Usage: curl [options...] <url>
  --abstract-unix-socket <path> Connect via abstract Unix domain socket
  --anyauth             Pick any authentication method
  -a, --append          Append to target file when uploading
  --basic              Use HTTP Basic Authentication
  --cacert <CA certificate> CA certificate to verify peer against
  --capath <dir>       CA directory to verify peer against
  -E, --cert <certificate[:password]> Client certificate file and password
  --cert-status        Verify the status of the server certificate
  --cert-type <type>   Certificate file type (DER/PEM/ENG)
  --ciphers <list of ciphers> SSL ciphers to use
  --compressed         Request compressed response
  -K, --config <file>   Read config from a file
  --connect-timeout <seconds> Maximum time allowed for connection
  --connect-to <HOST1:PORT1:HOST2:PORT2> Connect to host
  -C, --continue-at <offset> Resumed transfer offset
  -b, --cookie <data>  Send cookies from string/file
  -c, --cookie-jar <filename> Write cookies to <filename> after operation
  --create-dirs        Create necessary local directory hierarchy
  --crlf              Convert LF to CRLF in upload
  --crlfile <file>     Get a CRL list in PEM format from the given file
  -d, --data <data>    HTTP POST data
  --data-ascii <data> HTTP POST ASCII data
```

Narzędzie Swagger

- Narzędzie Swagger
 - Pakiet `Swashbuckle.AspNetCore`
- Po zainstalowaniu wystarczy uruchomić dodawanie serwisów i wstawić do strumienia przetwarzania żądania (kody poniżej)

```
// can be even without options
services.AddSwaggerGen(c =>
{
    c.SwaggerDoc("v1", new OpenApiInfo { Title = "WebAppApiTest", Version = "v1" });
});
```

Startup.cs

```
if (env.IsDevelopment())
{
    app.UseDeveloperExceptionPage();
    app.UseSwagger();
    app.UseSwaggerUI(c => c.SwaggerEndpoint("/swagger/v1/swagger.json", "WebAppApiTest v1"));
}
```

Startup.cs

- Narzędzie rozbudowane o wiele opcji pomocniczych
 - Można dostosować sobie interfejs graficzny tego narzędzia
 - API kontrolery mogą udostępniać dodatkowy opis operacji
 - itd.

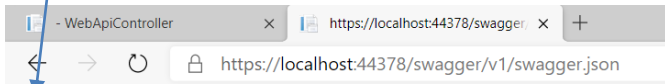
Zastosowanie Swagger-a

Dokument JSON opisujący endpoint-y wg specyfikacji OpenApi

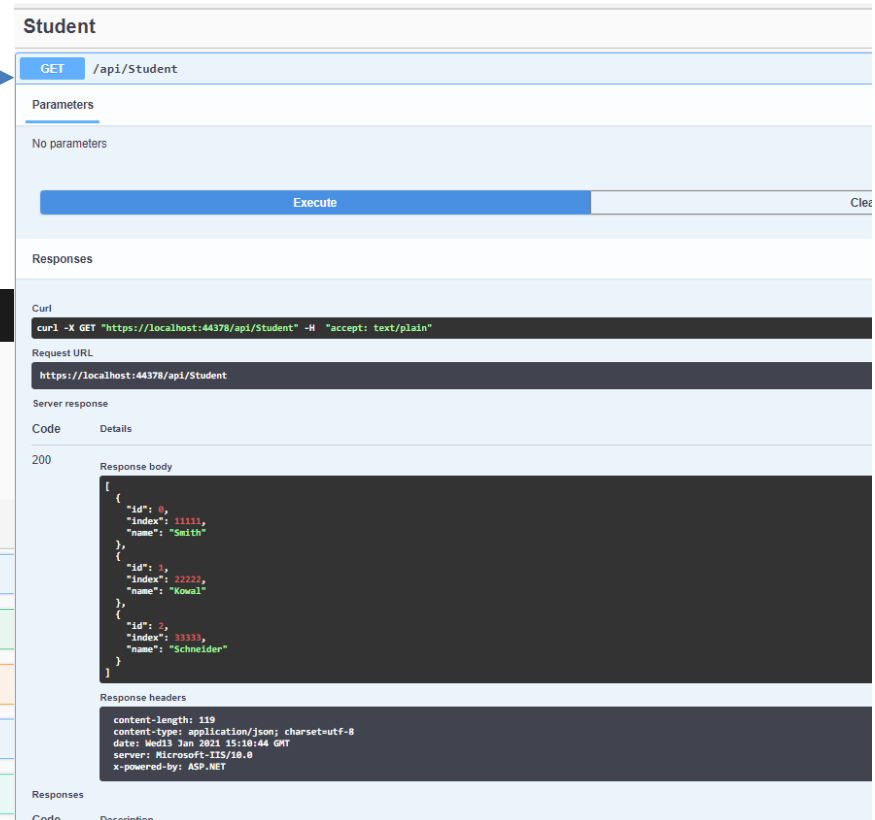
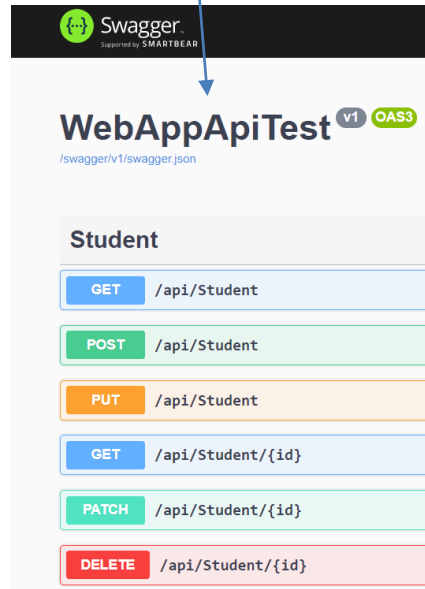
- <http://localhost:<port>/swagger/v1/swagger.json>

Testowanie działania endpointów:

- <http://localhost:<port>/swagger/>
- Np.. Wybrać GET, „try it out”, „Execute”



```
{
  "openapi": "3.0.1",
  "info": {
    "title": "WebAppApiTest",
    "version": "v1"
  },
  "paths": {
    "/api/Student": {
      "get": {
        "tags": [
          "Student"
        ],
        "responses": {
          "200": {
            "description": "Success",
            "content": {
              "text/plain": {
                "schema": {
                  "type": "array",
                  "items": {
                    "$ref": "#/components/schemas/Student"
                  }
                }
              }
            }
          }
        }
      }
    }
  }
}
```



AJAX

Problem – przechodzenie między studentami

- Chcemy na stronie `Details` z danymi studenta mieć możliwość przesunięcia się od razu do kolejnego/poprzedniego studenta, aby zobaczyć jego dane w tym samym widoku `Details`.
 - Kolejność mają wyznaczać identyfikatory `Id`. Jeśli nie posortujemy danych, trudno będzie określić, który jest następny.
- Założenie: do **dodawania** danych `Student`-a mamy akcje w kontrolerze `HomeController`, wykonane dla demonstracji kontrolera API (wcześniejsza część tego wykładu). Kolejne kody rozwiązują tylko problem przechodzenia między studentami (odczytu danych).
- W interfejsie `IRepository` i jego implementacji `MemoryRepository` tworzymy metody zwracające następnego/poprzedniego studenta po podanym indeksie.

Znajdowanie poprzedniego/następnego studenta.

- Jeśli poprzedniego/kolejnego elementu nie ma, zwrócona będzie wartość `null`.

```
public interface IRepository
{
    ...
    // next lines for AJAX presentation
    Student GetNextStudent(int id);
    Student GetPreviousStudent(int id);
}
```

IRepository.cs

```
public class MemoryRepository : IRepository
{
    ...
    // next lines for AJAX presentation
    public Student GetNextStudent(int id)
    {
        return items
            .Select(s => s.Value)
            .Where(s => s.Id > id)
            .OrderBy(s => s.Id)
            .FirstOrDefault();
    }
    public Student GetPreviousStudent(int id)
    {
        return items
            .Select(s => s.Value)
            .Where(s => s.Id < id)
            .OrderByDescending(s => s.Id)
            .FirstOrDefault();
    }
}
```

MemoryRepository.cs

Klasyczne rozwiązanie

- Rozwiązanie pierwsze – klasyczne:
 - Tworzymy `StudentController` kontroler dla `Studenta`.
 - Zostawiamy (tworzymy) akcje i widoki `Index` i `Details`
 - Wystarczające dla demonstracji działania
 - Do późniejszej modyfikacji
 - W akcji `Index` studentów sortujemy wg `id`.
 - W widoku `Index` przy każdym studentcie link do jego szczegółów (akcji `Details`)

```
public ActionResult Index()
{
    return View(repository.Students.OrderBy(s=>s.Id));
}
```

StudentController.cs

```
@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.Id)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Index)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Name)
        </td>
        <td>
            @Html.ActionLink("Details", "Details", new { id=item.Id })
        </td>
    </tr>
}
```

Index.cshtml

Nowe akcje

- Tworzymy nowe akcji do przechodzenia na poprzedniego/kolejnego studenta
 - Przekierowanie na akcję `Details` z nowym `Id` lub pozostanie na bieżącym jeśli nie ma poprzedniego/kolejnego studenta.

```
...
<dt class="col-sm-2">
    @Html.DisplayNameFor(model => model.Name)
</dt>
<dd id="name" class="col-sm-10">
    @Html.DisplayFor(model => model.Name)
</dd>
</dl>
</div>
<div>
    <a asp-action="Index">Back to List</a>
</div>
<div>
    StudentController:
    @Html.ActionLink("Prev", "Previous", new { id = Model.Id }) |
    @Html.ActionLink("Next", "Next", new { id = Model.Id })
</div>
<div>
    JavaScript (AJAX) :
    <button onclick="jsPrev()"> Previous </button> |
    <button onclick="jsNext()"> Next </button>
</div>
<div>
    jQuery (AJAX) :
    <button onclick="jqPrev()"> Previous </button> |
    <button onclick="jqNext()"> Next </button>
</div>
```

Details.cshtml

Widok Details

WebApiController Home Privacy Students

Index

[Create New](#)

Id	Index	Name	
0	11111	Smith	Details
1	22222	Kowal	Details
2	33333	Schneider	Details

© 2020 - WebApiController - [Privacy](#)

WebApiController Home Privacy Students

Details Student

Id	2
Index	33333
Name	Schneider

[Back to List](#)

StudentController: [Prev](#) | [Next](#)

JavaScript(AJAX): [Previous](#) | [Next](#)

jQuery(AJAX): [Previous](#) | [Next](#)

Nowe akcje kontrolera

- Ostatecznie należy stworzyć nowe akcje w kontrolerze: `Previous()` i `Next()`
 - Gdy repozytorium zwróci `null`, strona z `Details` ma pozostać na tych samych danych studenta.

```
public ActionResult Next(int id)
{
    Student stud = repository.GetNextStudent(id);
    if(stud==null)
        return RedirectToAction(nameof(Details), new { id });
    else
        return RedirectToAction(nameof(Details), new { id = stud.Id });
}
public ActionResult Previous(int id)
{
    Student stud = repository.GetPreviousStudent(id);
    if (stud == null)
        return RedirectToAction(nameof(Details), new { id });
    else
        return RedirectToAction(nameof(Details), new { id = stud.Id });
}
```

StudentController.cshtml

Działanie 1/3

- Akcja Index.

WebApiController Home Privacy Students

Index

[Create New](#)

Id	Index	Name	
0	11111	Smith	Details
1	22222	Kowal	Details
2	33333	Schneider	Details

Network

Filter: ☐ Invert ☐ Hide data URLs

All | Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest Other

☐ Has blocked cookies ☐ Blocked Requests ☐ 3rd-party requests

Name	Stat..	Type	Initiator	Size	T.	Waterfall
Student	200	doc...	Other	3.6 kB	1...	
bootstrap.min.css	200	styl...	Student	(memo...	0...	
site.css	200	styl...	Student	(memo...	0...	
jquery.min.js	200	script	Student	(memo...	0...	
bootstrap.bundle....	200	script	Student	(memo...	0...	
site.js?v=4q1jwFh...	200	script	Student	(memo...	0...	
data:image/svg+...	200	svg...	bootstra...	(memo...	0...	

Działanie 2/3

- Akcja Details (0).

The screenshot shows a web browser at `https://localhost:5001/Student/Details/0`. The page displays the details of a student with the following information:

Property	Value
Id	0
Index	11111
Name	Smith

Navigation links include "Back to List", "StudentController: Prev", and "Next". Below these are "JavaScript(AJAX): Previous | Next" and "jQuery(AJAX): Previous | Next". The "Next" link in the StudentController section is highlighted with a green box. A blue arrow points from this link to the bottom right of the slide.

The Chrome DevTools Network tab is open, showing a list of requests. The first request, labeled "0", is highlighted with a red box. Its details are as follows:

Name	Stat..	Type	Initiator	Size	T.	Waterfall
0	200	doc...	Other	5.5 kB	3...	[Waterfall chart]
bootstrap.min.css	200	styl...	0	(memo...	0...	[Waterfall chart]
site.css	200	styl...	0	(memo...	0...	[Waterfall chart]
jquery.min.js	200	script	0	(memo...	0...	[Waterfall chart]
bootstrap.bundle....	200	script	0	(memo...	0...	[Waterfall chart]
site.js?v=4q1jwFh...	200	script	0	(memo...	0...	[Waterfall chart]

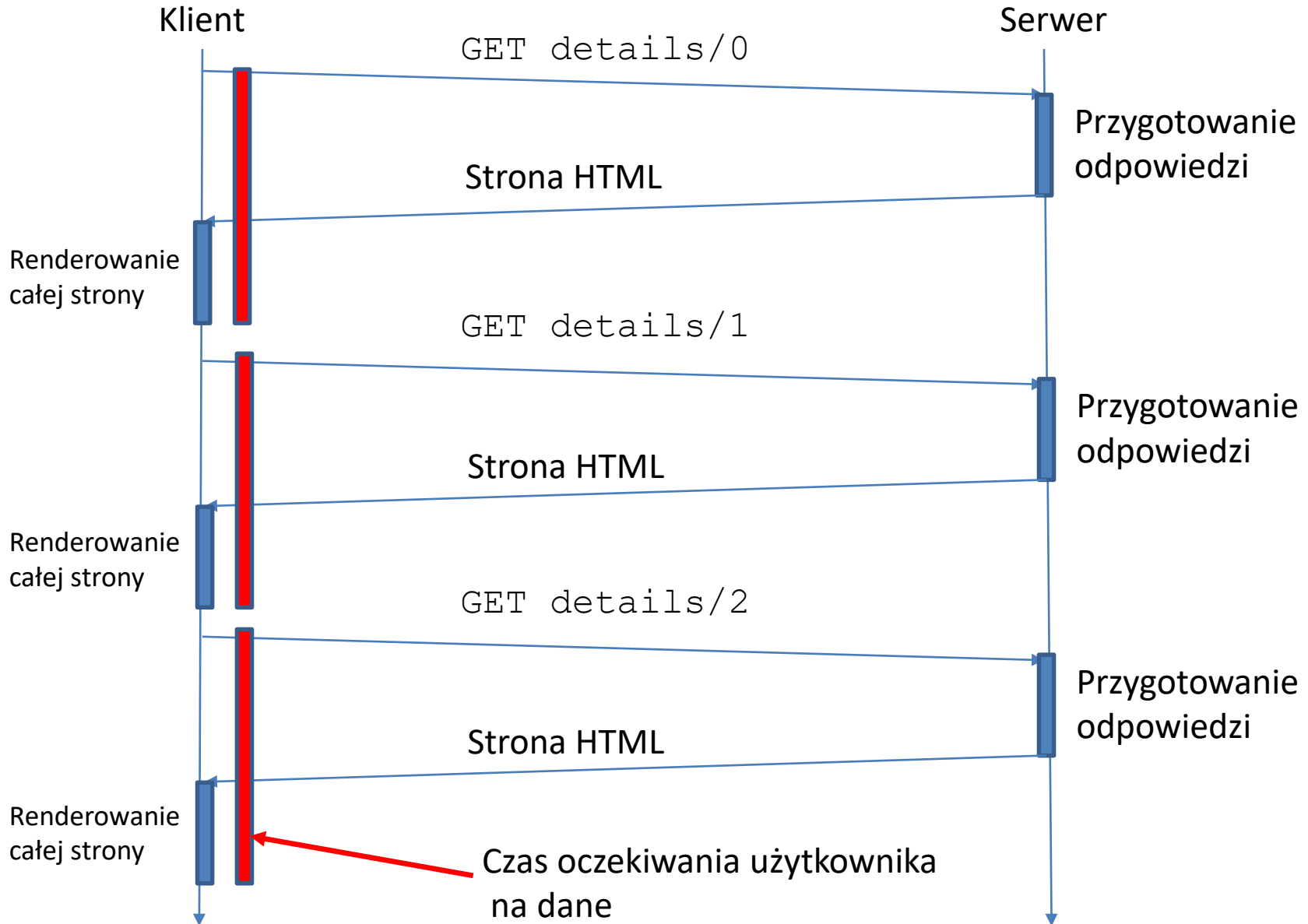
Działanie 3/3

- Pierwsze żądanie to przekierowanie
- Drugie to cała strona HTML z nowym studentem.
 - Jednak w zasadzie zmieniły się tylko dane (pomarańczowa ramka)
 - Pojawia się opóźnienie związane z ładowaniem, renderowaniem całej strony HTML

The screenshot shows a web browser at the URL `https://localhost:5001/Student/Details/1`. The page displays student details for ID 1, Index 22222, and Name Kowal. A yellow box highlights these details. The browser's developer tools are open to the Network tab, showing a list of requests. A red box highlights the first request, which is a 70 B document from the 'Other' initiator. The second request is a 5.4 kB document from the '5001/St...' initiator. The page also shows navigation links like 'Back to List', 'StudentController: Prev | Next', and 'JavaScript(AJAX): Previous | Next'.

Name	Stat..	Type	Initiator	Size	T.	Waterfall
0	302	doc...	Other	70 B	3...	
1	200	doc...	5001/St...	5.4 kB	5...	
bootstrap.min.css	200	styl...	1	(disk c...	2...	
jquery.min.js	200	script	1	(memo...	0...	
site.css	200	styl...	1	(disk c...	1...	
bootstrap.bundle....	200	script	1	(memo...	0...	
site.js?v=4q1jwFh...	200	script	1	(memo...	0...	

Klasyczna komunikacja



AJAX

- AJAX – (ang. Asynchronous JavaScript and XML) asynchroniczny JavaScript i XML
 - Czyli kod (funkcja) JavaScript, która w sposób asynchroniczny wyśle żądanie do serwera i wykona działanie po otrzymaniu odpowiedzi w czasie, gdy przeglądarka cały czas prezentuje bieżącą stronę WWW użytkownikowi i reaguje na jego działania.
- Używany protokół HTTP(S)
- Możliwość uruchomienia połączenia z klienta do serwera w sposób asynchroniczny:
 - Nawet wiele zapytań jednocześnie
 - Pytania mogą być do różnych domen WWW
 - Odpowiedzią nie musi być strona HTML
 - Może to być strumień w formacie JSON, np. z kontrolera API
- To kod w JavaScriptcie decyduje jak zinterpretować otrzymany strumień danych.

Rozwiązanie z AJAX-em

- Skorzystamy z wcześniej napisanego kontrolera API dla studenta
- **Dodamy do niego dwie akcje** analogiczne jak dla StudentController-a z innymi endpointami:
 - /api/student/prev/{id}
 - /api/student/next/{id}
- W widoku Details należy dołączyć np. przyciski, które zostaną obsłużone w JavaScriptcie (kod HTML być wcześniej, str 26) w funkcjach jsPrev() i jsNext().
 - Aby mogły dostać się do elementów DOM najlepiej nadać im identyfikatory.

StudentApiController.cshtml

```
[HttpGet("prev/{id}")]
public Student GetPrev(int id) =>
    repository.GetPreviousStudent(id);
[HttpGet("next/{id}")]
public Student GetNext(int id) =>
    repository.GetNextStudent(id);
```

Details.cshtml

```
<div>
    JavaScript(AJAX):
    <button onclick="jsPrev()"> Previous </button> |
    <button onclick="jsNext()"> Next </button>
</div>
```

Details.cshtml

```
<dl class="row">
    <dt class="col-sm-2">
        @Html.DisplayNameFor(model => model.Id)
    </dt>
    <dd id="id" class="col-sm-10">
        @Html.DisplayNameFor(model => model.Id)
    </dd>
    <dt class="col-sm-2">
        @Html.DisplayNameFor(model => model.Index)
    </dt>
    <dd id="index" class="col-sm-10">
        @Html.DisplayNameFor(model => model.Index)
    </dd>
    <dt class="col-sm-2">
        @Html.DisplayNameFor(model => model.Name)
    </dt>
    <dd id="name" class="col-sm-10">
        @Html.DisplayNameFor(model => model.Name)
    </dd>
</dl>
```

JavaScript z Ajax-em - kod

- Obiekt XMLHttpRequest.

```
@section Scripts{
<script>
    function jsAjax(word, suffix) {
        const xhr = new XMLHttpRequest();
        xhr.onload = function () {
            if (this.status === 200) {
                try {
                    const stud = JSON.parse(this.responseText);
                    //console.log(stud);
                    document.getElementById("id").innerHTML = stud.id;
                    document.getElementById("index").innerHTML = stud.index;
                    document.getElementById("name").innerHTML = stud.name;
                } catch (e) {
                    console.warn('There was an error in JSON. Could not parse.');
```

Details.cshtml

JavaScript z Ajax-em - opis

- Tworzymy obiekt `xhr` typu `XMLHttpRequest`.
 - XML w nazwie oznacza, że powinien być to dokument w formacie XML
 - W praktyce może być dowolny strumień danych
 - Obecnie w przypadku obiektów jest używany format JSON
 - Do prostych obiektów (cały graf byłoby trudno przesłać)
- W polu `onload` tworzymy funkcję, która zostanie wywołana, jeśli komunikacja zakończy się (poprawnie lub niepoprawnie).
 - W tej funkcji `this` oznacza obiekt `xhr`.
 - Pole `this.status` oznacza kod odpowiedzi
 - Pole `this.responseText` oznacza tekst **ładunku** (payload).
 - Kod 204 oznacza pusty ładunek, czyli kontroler API ASP .Net tak koduje wartość `null`.
 - Wówczas pokazujemy alert z odpowiednim napisem.
 - Odpowiedź JSON zamienia jest na **obiekt** JavaScript za pomocą metody `JSON.parse()`.
 - Dane z obiektu przepiszemy do odpowiednich obiektów DOM.
 - W przypadku niepowodzenia komunikacji na konsolę Javascriptu wypisujemy odpowiedni komunikat
- Przygotowujemy dane do żądania korzystając z obecnej zawartości elementu o identyfikatorze „id”, jednak może on mieć różne białe znaki przed i za wartością liczbową więc usuwamy je metodą `trim()`.
- Metoda `xhr.open()` przygotowuje strumień żądania.
 - Podajemy w niej typ żądania oraz URL.
 - Można jeszcze nim manipulować np. dodając zawartość lub parametry w nagłówkach żądania.
- Ostateczne wysłanie żądania w sposób asynchroniczny następuje po wywołaniu metody `xhr.send()`.
 - Przeglądarka działa niezależnie
 - Po odebraniu odpowiedzi wywołana zostanie metoda zapamiętana w `xhr.onload`.
- Ogólna metoda `jsAjax(word, suffix)` wymaga podania słowa `word`, które pojawi się w alercie, gdy nie ma poprzedniego/następnego elementu oraz końcówki `suffix` adresu URL na który ma być wysłane żądanie.

JavaScript z Ajax-em – działanie 1/4

- Akcja Index.

WebApiController Home Privacy Students

Index

[Create New](#)

Id	Index	Name	
0	11111	Smith	Details
1	22222	Kowal	Details
2	33333	Schneider	Details

Network tab:

Name	Stat..	Type	Initiator	Size	T.	Waterfall
Student	200	doc...	Other	3.6 kB	1...	
bootstrap.min.css	200	styl...	Student	(memo...	0...	
site.css	200	styl...	Student	(memo...	0...	
jquery.min.js	200	script	Student	(memo...	0...	
bootstrap.bundle...	200	script	Student	(memo...	0...	
site.js?v=4q1jwFh...	200	script	Student	(memo...	0...	
data:image/svg+...	200	svg...	bootstra...	(memo...	0...	

JavaScript z Ajax-em – działanie 2/4

- Akcja Details (0).

The screenshot shows a web browser at the URL `https://localhost:5001/Student/Details/0`. The page displays the details of a student with the following information:

Property	Value
Id	0
Index	11111
Name	Smith

Navigation links include "Back to List", "StudentController: Prev", and "Next". Below these are "JavaScript(AJAX): Previous" and "Next", and "jQuery(AJAX): Previous" and "Next". A green box highlights the "Next" link in the "JavaScript(AJAX)" section, with a blue arrow pointing to the network traffic in the developer tools.

The developer tools show the "Network" tab with a list of requests. The first request, labeled "0", is highlighted with a red box. It is a document request (type "doc...") initiated by "Other", with a size of 5.5 kB and a status of 200. The waterfall chart shows a single request line.

Name	Stat..	Type	Initiator	Size	T.	Waterfall
0	200	doc...	Other	5.5 kB	3...	[Waterfall chart]
bootstrap.min.css	200	styl...	0	(memo...	0...	[Waterfall chart]
site.css	200	styl...	0	(memo...	0...	[Waterfall chart]
jquery.min.js	200	script	0	(memo...	0...	[Waterfall chart]
bootstrap.bundle....	200	script	0	(memo...	0...	[Waterfall chart]
site.js?v=4q1jwFh...	200	script	0	(memo...	0...	[Waterfall chart]

JavaScript z Ajax-em – działanie 3/4

- Jesteśmy w ramach jednego głównego żądania Details/0.
- Pojawiło się jedno **dodatkowe żądanie**
`https://localhost:5001/api/student/next/0`
- Liczba przesłanych danych bardzo się zmniejszyła z 5,5 KB do 135 B.
- Przeglądarka nie odświeża całej strony, tylko 3 elementy DOM.

The screenshot shows a web browser at `https://localhost:5001/Student/Details/0`. The page title is 'Details' and the sub-header is 'Student'. The student details are: Id: 1, Index: 22222, Name: Kowal. There are navigation links: 'Back to List', 'StudentController: Prev | Next', 'JavaScript(AJAX): Previous | Next', and 'jQuery(AJAX): Previous | Next'. The network tab is open, showing a list of resources. The 'next/0' request is highlighted, showing a size of 135 B. The word 'kontynuacja' is written next to the highlighted request.

Name	Stat	Type	Initiator	Size	T	Waterfall
0	200	doc...	Other	5.4 kB	1...	
bootstrap.min.css	200	styl...	0	(memo...	0...	
site.css	200	styl...	0	(memo...	0...	
jquery.min.js	200	script	0	(memo...	0...	
bootstrap.bundle...	200	script	0	(memo...	0...	
site.is?v=4n1wFh	200	script	0	(memo...	0...	
0	200	xhr	0:122	135 B	5...	kontynuacja

JavaScript z Ajax-em – działanie 4/4

- Po **trzecim** wciśnięciu klawisza „Next” (na ostatnim studencie w bazie)

WebApiController Home Privacy Student

Details

Student

Id 2
Index 33333
Name Schneider

[Back to List](#)

StudentController: [Prev](#) | [Next](#)

JavaScript(AJAX): [Previous](#) | [Next](#)

jQuery(AJAX): [Previous](#) | [Next](#)

Komunikat z witryny localhost:5001:
No next element
OK

Network x » + 1

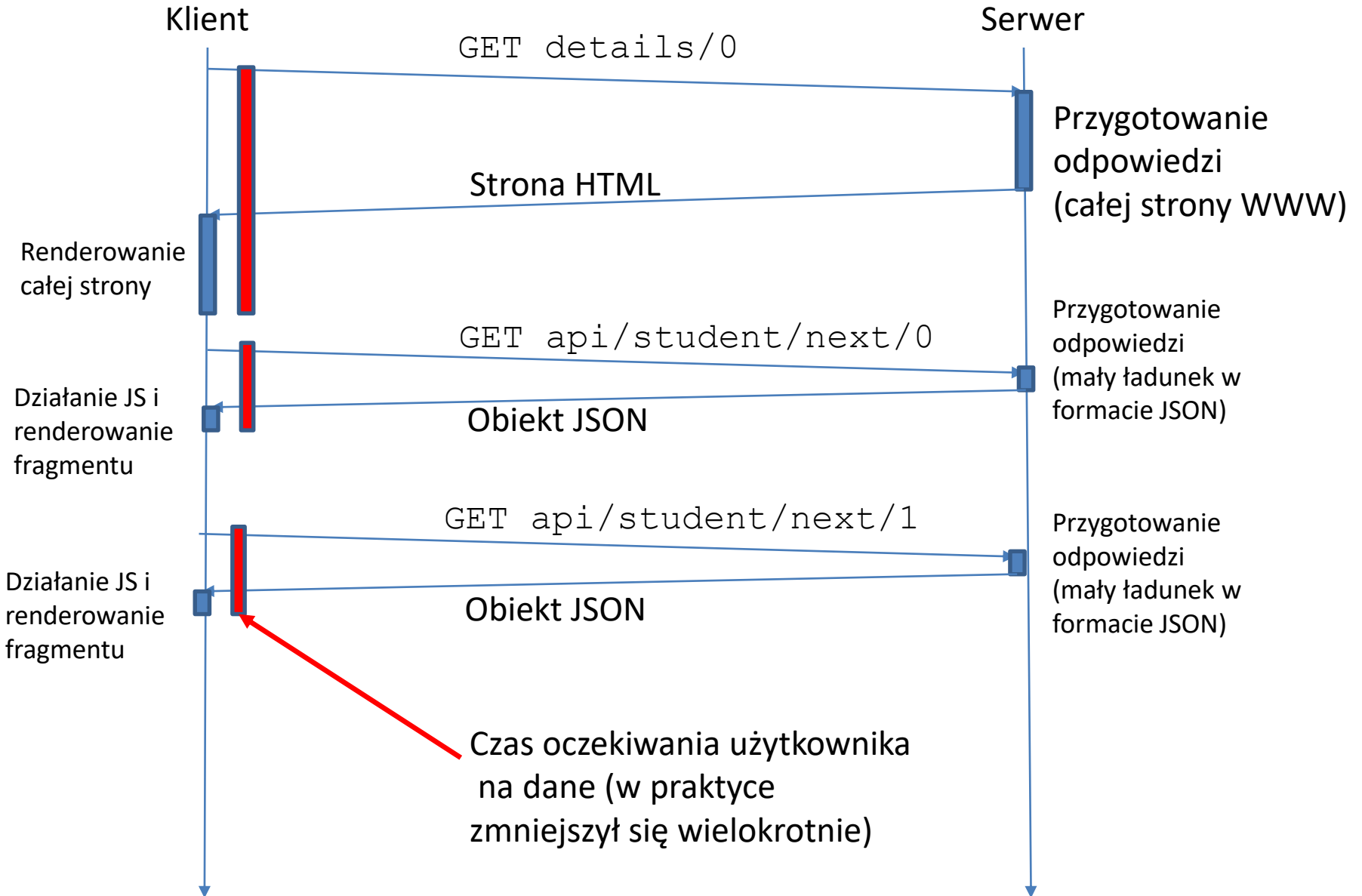
log ☐ Disable cache No throttling ☐ Invert ☐ Hide data URLs
Font Doc WS Wasm Manifest Other

☐ Has blocked cookies ☐ Blocked Requests ☐ 3rd-party requests

Name	Stat..	Type	Initiator	Size	T.	Waterfall
0	200	doc...	Other	5.4 kB	1...	
bootstrap.min.css	200	styl...	0	(memo...	0...	
site.css	200	styl...	0	(memo...	0...	
jquery.min.js	200	script	0	(memo...	0...	
bootstrap.bundle...	200	script	0	(memo...	0...	
site.js?v=4q1jwFh...	200	script	0	(memo...	0...	
0	200	xhr	0:122	135 B	5...	
1	200	xhr	0:122	147 B	3...	
2	204	xhr	0:122	46 B	P...	

kontynuacja

Komunikacja z AJAX-em



Dodatek - Ajax w jQuery

- Najniższe przyciski w widoku Details przygotowane są dla funkcji napisanych w jQuery.
 - Zapis bardziej skompresowany niż bezpośrednio w Javascriptcie

```
function jqAjax(word, suffix) {  
    $.ajax({  
        type: "GET",  
        url: "/api/student/" + suffix + "/" + $("#id").html().trim(),  
        success: function (stud, textStatus, jqXHR) { // codes 200..299  
            if (jqXHR.status === 204) { // 204 No Content, so NULL  
                window.alert("No " + word + " element");  
                return;  
            }  
            $('#id').html(stud.id);  
            $('#index').html(stud.index);  
            $('#name').html(stud.name);  
        }  
    })  
    .fail(function (jqXHR, textStatus) { // codes 400..499  
        console.warn("Recived " + jqXHR.status + " in response code.");  
    });  
}  
  
function jqPrev(){  
    jqAjax("previous", "prev");  
}  
  
function jqNext() {  
    jqAjax("next", "next");  
}  
</script>
```

Details.cshtml

Visual Studio 2019 – projekt „API kontroler”

- Gotowy szablon dla projektu tylko z kontrolerem API



Internetowy interfejs API platformy ASP.NET Core

Szablon projektu służący do tworzenia aplikacji platformy ASP.NET Core z przykładowym kontrolerem obsługującym usługę HTTP RESTful. Tego szablonu można także użyć dla widoków i kontrolerów platformy ASP.NET Core MVC.

```
app.UseEndpoints(endpoints =>
{
    endpoints.MapControllers(); // enough for API controllers
});
```

Publiczne RESTful kontrolery


- Zamiast parsować dokument HTML i szukać potrzebnych danych, lepiej użyć kontrolerów API (jeśli twórca strony z danymi takie udostępnia)
- Dostęp do kontrolerów RESTfull powinno się połączyć z autoryzacją i autentykacją.
- Część portali posiada dostępne publiczne serwery RESTfull
 - Dla programistów
 - Często wymagają rejestracji
 - Dla dużej liczby zapytań są płatne
 - Głównie operacja GET
- Przykład: Google Maps
 - <https://developers.google.com/maps/documentation>

Wady/zalety kontrolerów API

- Zalety:
 - Mniej danych jest przesyłanych
 - Szybsze działanie strony
 - Projekt można łatwiej podzielić na część backend-ową z kontrolerami API oraz część frontend-ową z AJAX-em
- Wady:
 - 1) Potrzebny kod w Javascriptcie
 - 2) trudno automatycznie podlinkować stronę (URL się nie zmienia)
- Rozwiązanie wad:
 - Ad 1) Istnieją framework-i ułatwiające użycie AJAX-a
 - Ad 1) Albo nawet rozwiązania typu Blazor Web Assembly, gdzie można pisać w C#
 - Ad 2) rozwiązanie mieszane: kontroler MVC i kontrolerAPI, oraz przycisk pobierania linku do strony, który będzie obsługiwany przez kontroler MVC.

Blazor, aplikacje SPA

Tworzenie aplikacji Blazor – solucja BlazorStudent

 **Blazor App**
Project templates for creating Blazor apps that run on the server in an ASP.NET Core app or in the browser on WebAssembly (wasm). These templates can be used to build web apps with rich dynamic user interfaces (UIs).

C# Linux macOS Windows Cloud Web

.NET 5.0



Blazor Server App

A project template for creating a Blazor server app that runs server-side inside an ASP.NET Core app and handles user interactions over a SignalR connection. This template can be used for web apps with rich dynamic user interfaces (UIs).

- Układ podobny do innego typu projektów
- Istnieją pliki z rozszerzeniem `.razor`
- Pliki w folderze `Shared` o trochę innych nazwach.
- Pliki konfiguracyjne takie same, różnica:

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddRazorPages();
    services.AddServerSideBlazor();
    services.AddSingleton<WeatherForecastService>();
}
```

```
app.UseEndpoints(endpoints =>
{
    endpoints.MapBlazorHub();
    endpoints.MapFallbackToPage("/_Host");
});
```

Startup.cs

Solution 'BlazorStudent' (1 of 1 project)

- BlazorStudent
 - Connected Services
 - Dependencies
 - Properties
 - wwwroot
 - Data
 - IServiceService.cs
 - MemoryStudentService.cs
 - Student.cs
 - WeatherForecast.cs
 - WeatherForecastService.cs
 - Pages
 - Student
 - Create.razor
 - Edit.razor
 - Index.razor
 - _Host.cshtml
 - Counter.razor
 - Error.cshtml
 - FetchData.razor
 - Index.razor
 - Shared
 - MainLayout.razor
 - NavMenu.razor
 - SurveyPrompt.razor
 - _Imports.razor
 - App.razor
 - appsettings.json
 - Program.cs
 - Startup.cs

Projekt typu Blazor

- Aplikacja SPA – Single Page Application
 - Na początku ładowane do przeglądarki wszystkie widoki itd.
 - Nawigacja między stronami to komunikacja między serwerem a stroną z niewielką liczbą informacji
 - Formularze (kod HTML) nie są wysyłane wielokrotnie, tylko dane z/do nich
- Standardowo SPA to kod napisany w JavaScriptcie.
- Blazor to możliwość pisania w C#, który zostanie zamieniony na kod wywołania metod Javascript w ramach silnika
`blazor.server.js`
- Do wymiany danych używa WebSocket-ów.
 - Dwukierunkowa komunikacja
 - Raczej małe paczki danych
 - Co kilka sekund sprawdzenie, czy serwer działa
- Przykład: Bazując na wygenerowanym szkieletcie kodu dodanie operacji CRUD na kolekcji studentów.

Oczekiwany rezultat

BlazorStudent

[Home](#)
[Counter](#)
[Fetch data](#)
[Students](#)

Students

[Create new student](#)

Id	Index	Name		
0	11111	Smith	Edit	<button>Delete</button>
1	22222	Kowal	Edit	<button>Delete</button>
2	33333	Schneider	Edit	<button>Delete</button>

[Home](#)
[Counter](#)
[Fetch data](#)
[Students](#)

Edit student

Id

Index

Name

Update

BlazorStudent

[Home](#)
[Counter](#)
[Fetch data](#)
[Students](#)

Create student

Index

Name

Create

Przygotowanie danych

- Implementacja klasy `Student`, `IStudentService`, `MemoryStudentService` i wstrzyknięcie do kontenera serwisów.

```
public class Student
{
    9 references
    public int Id { get; set; }
    12 references
    public int Index { get; set; }
    12 references
    public string Name { get; set; }
}
```

```
public interface IStudentService
{
    2 references
    List<Student> Students { get; }
    2 references
    Student this[int id] { get; }
    3 references
    Student AddStudent(Student student);
    2 references
    Student UpdateStudent(Student student);
    2 references
    void DeleteStudent(int id);
}
```

```
public class MemoryStudentService : IStudentService
{
    private readonly Dictionary<int, Student> items;
    0 references
    public MemoryStudentService()
    {
        items = new Dictionary<int, Student>();
        new List<Student>
        {
            new Student{ Index=11111, Name="Smith"},
            new Student{ Index=22222, Name="Kowal"},
            new Student{ Index=33333, Name="Schneider"}
        }.ForEach(s => AddStudent(s));
    }
    2 references
    public Student this[int id] => items.SingleOrDefault(x=>x.Key==id).Value;

    2 references
    public List<Student> Students => items.Values.ToList();

    3 references
    public Student AddStudent(Student student)
    {
        if (student.Id == 0)
        {
            int key = items.Count;
            while (items.ContainsKey(key)) { key++; };
            student.Id = key;
        }
        items[student.Id] = student;
    }
}
```

```
services.AddSingleton<WeatherForecastService>();
services.AddSingleton<IStudentService, MemoryStudentService>();
```

Dodanie opcji do menu

```
<NavLink class="nav-link" href="fetchdata">
  <span class="oi oi-list-rich" aria-hidden="true"></span> Fetch data
</NavLink>
</li>
<li class="nav-item px-3">
  <NavLink class="nav-link" href="student/index">
    <span class="oi oi-list-rich" aria-hidden="true"></span> Students
  </NavLink>
</li>
</ul>
```

Operacje CRUD dla Studenta w Blazorze

- Plik `Index.razor` - ważne miejsca
 - Jako strona Razora
 - `@inject` – jak parametry konstruktora, ale dodatkowo jak właściwość
 - `@onclick` – wiązanie do zdarzenia
 - `@code` – ciało klasy wytworzonej dla tej strony
 - Metoda `OnInitialized()` uruchamiana przed pokazaniem strony
 - Może być asynchroniczna i po uzyskaniu wyniku strona jest ponownie tworzona (`FetchData.Razor`)

```
<button @onclick="@(>OnDelete(student.Id))">
    Delete
</button>
</td>
</tr>
</tbody>
</table>

@code {
    private List<BlazorStudent.Data.Student> students;

    protected override void OnInitialized()
    {
        //base.OnInitialized();
        students = StudentService.Students;
    }

    private void OnDelete(int id)
    {
        StudentService.DeleteStudent(id);
        Navigator.NavigateTo("/student/index", true);
    }
}
```

```
@page "/student/index"
@inject BlazorStudent.Data.IStudentService StudentService;
@inject NavigationManager Navigator

<h1>Students</h1>
```

Plik Edit.razor

```
@page "/students/edit/{id:int}"
@inject BlazorStudent.Data.IStudentService StudentService
@inject NavigationManager Navigator

<h1>Edit student</h1>

<EditForm Model="student" OnSubmit="@SubmitFunction">
    <div class="form-group">
        <label>Id</label>
        <input id="id" class="form-control" name="id" value="@student.Id" readonly />
    </div>
    <div class="form-group">
        <label>Index</label>
        <InputNumber id="index" class="form-control" name="index" @bind-Value="student.Index"> </InputNumber>
    </div>
    <div class="form-group">
        <label>Name</label>
        <InputText id="name" class="form-control" name="name" @bind-Value="student.Name"> </InputText>
    </div>
    <button type="submit" class="btn btn-primary">Update</button>
</EditForm>

@code {
    [Parameter]
    public int Id { get; set; }

    private BlazorStudent.Data.Student student;

    protected override void OnInitialized()
    {
        //base.OnInitialized();
        student = StudentService[Id];
    }

    private void SubmitFunction()
    {
        StudentService.UpdateStudent(student);
        Navigator.NavigateTo("/student/index");
    }
}
```

Obserwacja działania

- Najpierw w przeglądarce
 - Działa jak na początkowych slajdach
- W narzędziu developerskim (F12)
 - Ustawić: Disable cache
 - Przeładować: Reload
 - `blazor.server.js` – SPA silnik
 - `_blazor?id=SDFR....` - websocket

Name
<input type="checkbox"/> counter
<input type="checkbox"/> negotiate?negotiateVersion=1
<input type="checkbox"/> open-iconic.woff
<input checked="" type="checkbox"/> blazor.server.js
<input checked="" type="checkbox"/> bootstrap.min.css
<input checked="" type="checkbox"/> site.css
<input checked="" type="checkbox"/> BlazorStudent.styles.css
<input checked="" type="checkbox"/> open-iconic-bootstrap.min.css
<input type="checkbox"/> _blazor?id=2fn2RRAfzMHIQ7KsBqUdxQ
<input type="checkbox"/> favicon.ico

Name	Headers	Messages	Initiator	Timing
<input type="checkbox"/> counter	All	Enter regex, for example: (web)?socket		
<input type="checkbox"/> negotiate?negotiateVersion=1				
<input type="checkbox"/> open-iconic.woff				
<input checked="" type="checkbox"/> blazor.server.js				
<input checked="" type="checkbox"/> bootstrap.min.css				
<input checked="" type="checkbox"/> site.css				
<input checked="" type="checkbox"/> BlazorStudent.styles.css				
<input checked="" type="checkbox"/> open-iconic-bootstrap.min.css				
<input checked="" type="checkbox"/> _blazor?id=2fn2RRAfzMHIQ7K...				
<input type="checkbox"/> favicon.ico				

Data	Leng...	Time
↑ {"protocol":"blazorpac...	38	21:04:21.938
↓ Binary Message	3 B	21:04:21.940
↑ Binary Message	497 B	21:04:21.946
↓ Binary Message	29 B	21:04:21.950
↓ Binary Message	4.1 kB	21:04:21.954
↑ Binary Message	26 B	21:04:21.960
↓ Binary Message	229 B	21:04:21.960
↓ Binary Message	93 B	21:04:21.965
↑ Binary Message	44 B	21:04:21.967
↑ Binary Message	3 B	21:04:37.265
↓ Binary Message	3 B	21:04:37.619

Obserwacja komunikacji

- Częste trzybajtowe komunikaty
 - Czy serwer działa
- W przypadku różnych formularzy 2-3 kB dane
- Komunikacja dwustronna:
 - Zielona strzałka w górę: wysyłanie danych
 - Czerwona strzałka w dół: odbieranie informacji

x Headers Messages Initiator Timing		
All Enter regex, for example: (web)?socket		
Data	Leng...	Time
↑ Binary Message	3 B	21:10:38.266
↓ Binary Message	3 B	21:10:40.608
↑ Binary Message	378 B	21:10:44.485
↓ Binary Message	148 B	21:10:44.486
↑ Binary Message	26 B	21:10:44.488
↓ Binary Message	3.0 kB	21:10:44.499
↑ Binary Message	78 B	21:10:44.502
↓ Binary Message	3 B	21:10:55.617
↑ Binary Message	66 B	21:10:56.662
↓ Binary Message	2.5 kB	21:10:56.697
↑ Binary Message	26 B	21:10:56.699
↓ Binary Message	133 B	21:10:56.701
↑ Binary Message	26 B	21:10:56.702
↓ Binary Message	3 B	21:11:11.622
↑ Binary Message	3 B	21:11:12.265
↓ Binary Message	3 B	21:11:27.622
↑ Binary Message	3 B	21:11:28.264
↓ Binary Message	3 B	21:11:43.619
↑ Binary Message	3 B	21:11:44.264
↓ Binary Message	3 B	21:11:59.620
↑ Binary Message	3 B	21:12:00.264

Projekt typu WebAssembly App



Blazor WebAssembly App

A project template for creating a Blazor app that runs on WebAssembly. This template can be used for web apps with rich dynamic user interfaces (UIs).

- Brak części serwerowej
- Brak komunikacji poprzez websocket
- Dane najlepiej pozyskiwać z serwera API
 - np. w formacie JSON
 - W sposób asynchroniczny (zamienione zostanie na AJAX)
- Blazor - Podsumowanie:
 - Ciekawa technologia, pozwala pisać „kod po stronie klienta” w C#
 - Zapomnieć o JavaScript, chociaż powstaje kod JS
 - Ciągłe w fazie rozwoju, ale Microsoft bardzo w niego inwestuje.
 - Wspiera dwa różne podejścia (Blazor Server App i Blazor WebAssembly App)