

Spring Fundamentals

WEB API and REST Controllers



SoftUni Team
Technical Trainers



SoftUni



Software University

<https://softuni.bg>

1. REST API

- RESTful Design
- HTTP GET, POST, PUT, DELETE, PATCH Examples

2. REST with Spring

3. Rest Template

4. DOM Manipulations

5. FETCH

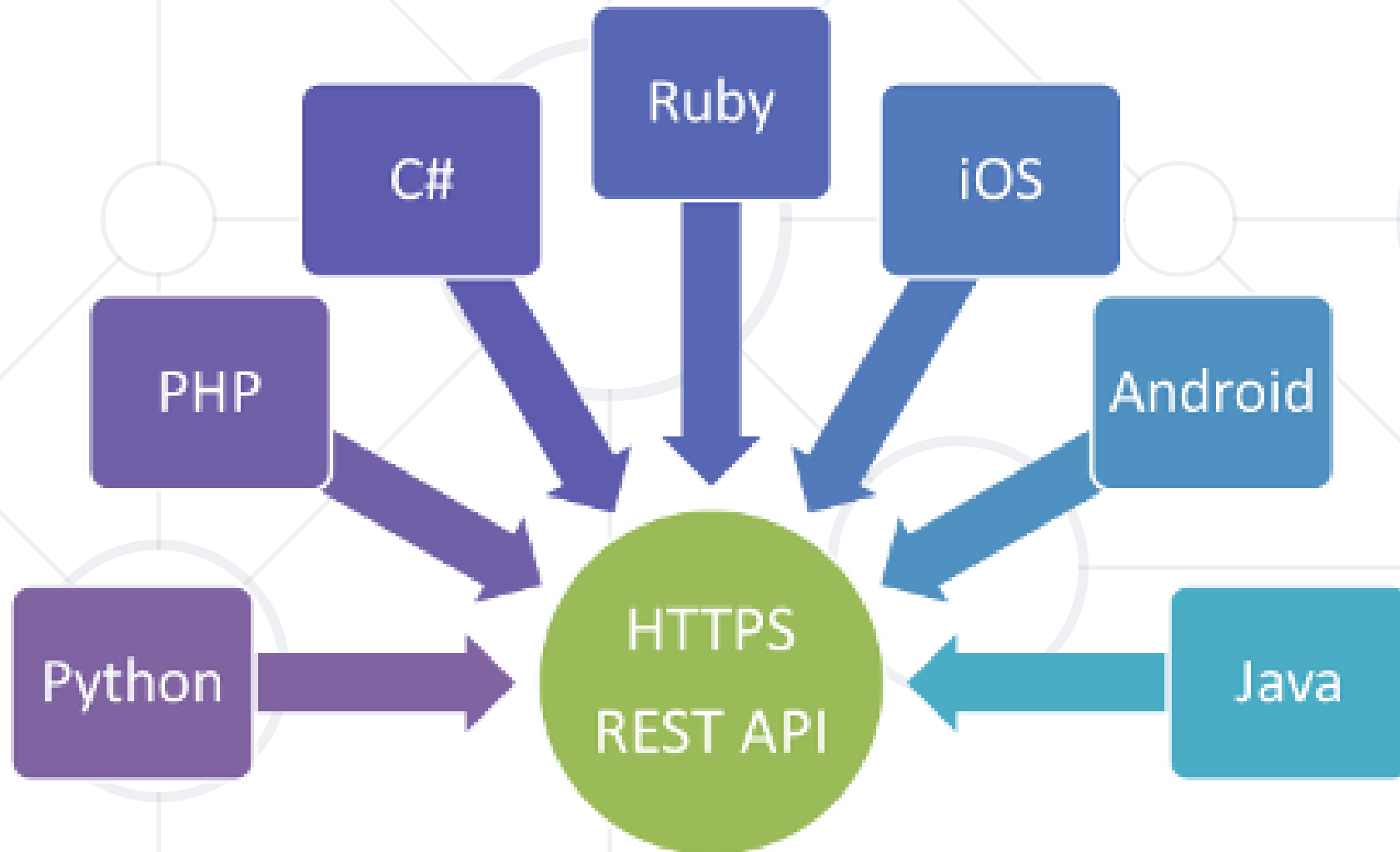
sli.do

#java-web



REST API

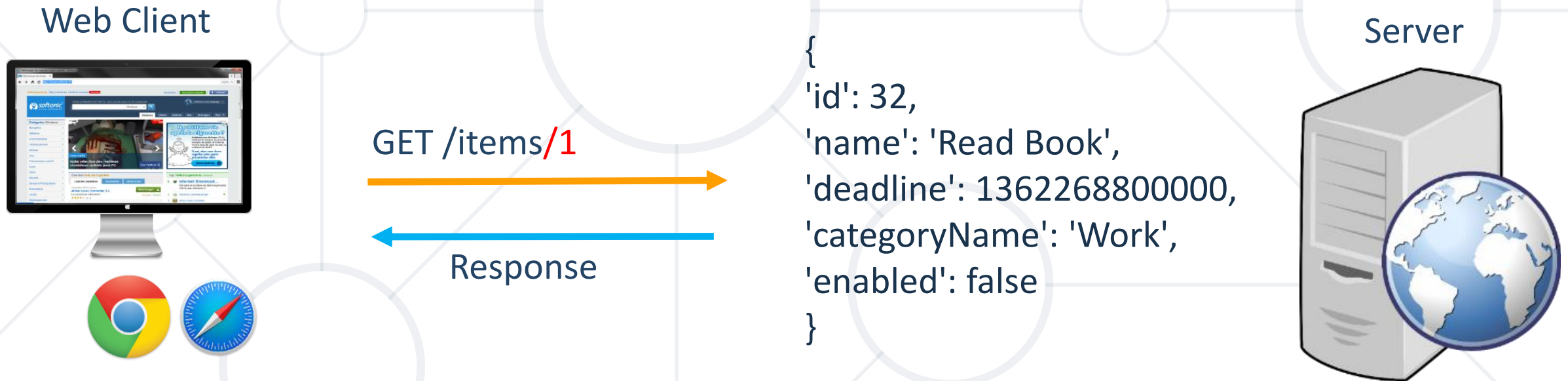
RESTful Design



- True RESTful API, is a **web service** must adhere to the following six **REST architectural constraints**
 - Use of a **uniform interface (UI)**
 - **Client-server based**
 - **Stateless** operations
 - RESTful **resource caching**
 - **Layered system**
 - **Code on demand**

- **Simple Object Access Protocol (SOAP)**
 - Standardized protocol that **sends messages** using other protocols such as **HTTP** and **SMTP**
 - The SOAP specifications are official web standards, maintained and developed by the World Wide Web Consortium (W3C)
- **Remote Procedure Call (RPC)**
 - A way to describe a mechanism that lets you **call a procedure in another process** and **exchange data by message passing**

- Used to retrieve single data entities



- Used to retrieve data arrays



GET /items



Response



```
[  
{  
  'id': 32,  
  'name': 'Read Book',  
  'deadline': 1362268800000,  
  'categoryName': 'Work',  
  'enabled': false  
},  
...  
]
```

Server



- Used to save data

Web Client



```
{  
  'id': 32,  
  'name': 'Read Book',  
  'deadline': 1362268800000,  
  'categoryName': 'Work',  
  'enabled': false  
}
```

POST /**items**

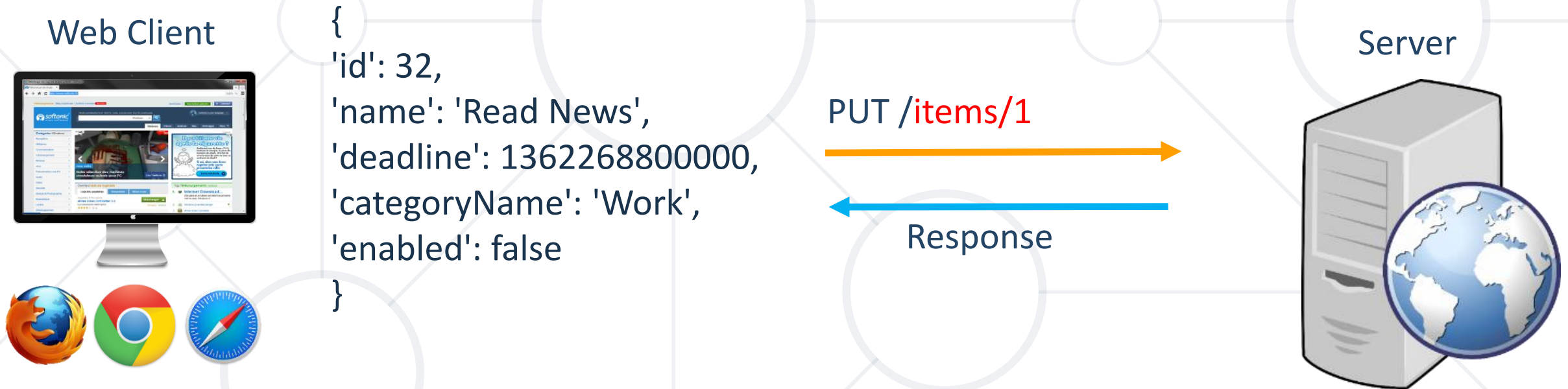


Response

Server



- Used to update data.



HTTP DELETE

- Used to delete data.

Web Client



Server



DELETE /items/delete/1



Response

OK Response



REST with Spring

Creating REST API with Spring

- Returning plain-text in MVC controller:

```
@GetMapping('/info/{id}')
@ResponseBody
public Student getInfo(@PathVariable Long id){
    ...
    return new Student().setName("Joro");
}
```

- Setting the correct Response Code

```
@GetMapping('{id}/info')
@ResponseStatus(HttpStatus.OK)
public GameInfoView getInfo(@PathVariable Long id){

    GameInfoView gameInfo = this.gameService.getInfoById(id);

    return new Gson().toJson(gameInfo);
}
```

- **@RestController** is essentially **@Controller + @ResponseBody**

```
@RestController
public class OrderController {

    @GetMapping('{id}/info')
    public ResponseEntity<Game> getGame(@PathVariable Long id){
        ...
    }
}
```


- Controlling the entire response object

```
@GetMapping('{id}/title')  
public ResponseEntity<Game> getTitle(...){  
    ...  
    return new ResponseEntity(gameService.getGame(id));  
}
```

- The **ResponseEntity<>** object allows you **to change the response body**, response headers and response code

- Maven Dependency

```
<dependency>  
  <groupId>org.springframework.boot</groupId>  
  <artifactId>spring-boot-starter-data-rest</artifactId>  
</dependency>
```

- Spring Data REST **scans your project** and **provides REST API** for your application **using HAL** as media type

- You can configure repository settings using the **@RepositoryRestResource** annotation:

```
@RepositoryRestResource(path = 'gameIssues')
public interface IssueRepository extends
    JpaRepository<Issue, Long> {
    Issue getById(@Param('id') Long id);
    List<Issue> getAllByOrderByDateDesc();
}
```



Rest Template

- Accessing **a third-party REST service** inside a Spring application revolves around the use of the Spring **RestTemplate class**
- Class is **designed to call REST services**
- Its **main methods** are closely tied to **REST's underpinnings**, which are the **HTTP protocol's methods: HEAD, GET, POST, PUT, DELETE**
- **Recommended** to use the non-blocking, **reactive WebClient**.
- RestTemplate will be **deprecated in a future version**

- **getForObject(url, classType)**
 - Retrieves a **representation by doing a GET on the URL.**
 - The response (if any) is unmarshalled to given class type and returned
- **getForEntity(url, responseType)**
 - Retrieve a **representation as ResponseEntity** by doing a GET on the URL

- **exchange(requestEntity, responseType)**
 - Executes the specified **request** and **returns** the response as **ResponseEntity**
- **execute(url, httpMethod, requestCallback, responseExtractor)**
 - Executes the **httpMethod** to the given URI template and preparing the request with the **RequestCallback**

- **postForObject(url, request, classType)**
 - **POSTs** the given object **to the URL** and **returns the representation** found in the response **as given class type**
- **postForEntity(url, request, responseType)**
 - **POSTs** the given object **to the URL** and **returns the response as ResponseEntity**

- **postForLocation(url, request, responseType)**
 - **POSTs** the given object **to the URL** and **returns** the value of the **Location header**
- **exchange(url, requestEntity, responseType)**
- **execute(url, httpMethod, requestCallback, responseExtractor)**

HTTP PUT and HTTP DELETE

- **put(url, request)**
 - PUTs the given request object to URL
- **delete(url)**
 - Deletes the resource at the specified URL



DOM Manipulations

- Create with `document.createElement`

```
let p = document.createElement('p');
```

- Append text to the `<p>` element

```
let text = document.createTextNode('Random Text');
```

```
p.appendChild(text);
```

- Text added to `textContent` will be escaped.
- Text added to `innerHTML` will be parsed and turned into actual HTML elements beware of XSS attacks!

Creating DOM Elements

```
let list = document.createElement('ul');  
let liPeter = document.createElement('li');  
liPeter.textContent = 'Peter';  
list.appendChild(liPeter);  
  
let liMaria = document.createElement('li');  
liMaria.innerHTML = '<b>Maria</b>';  
list.appendChild(liMaria);  
document.body.appendChild(list);
```

```
▼ <ul>  
  <li>Peter</li>  
  ▼ <li>  
    <b>Maria</b>  
  </li>  
</ul>
```

- To remove an HTML element, you must know the his parent

```
<div id='div1'>  
  <p id='p1'>This is a paragraph.</p>  
  <p id='p2'>This is another paragraph.</p>  
</div>
```

```
let parent = document.getElementById('div1');  
let child = document.getElementById('p1');  
parent.removeChild(child);
```



jQuery and DOM

- **text()** - reads and writes text

```
let text = $('#theElement').text();  
$('#theElement').text('New text for element.');
```

- **html()** - returns the HTML of a given element

```
let html = $('#theElement').html();  
$('#theElement').html('New text for element.');
```

- **val()** - gets and sets value

```
let theValue = $('#theFormField').val();  
$('#theFormField').val('New value');
```


- **attr()** - reads and writes attributes of HTML elements. Also can take an object as parameter

```
let attrValue = $('#theFormField').attr('height');  
$('#theFormField').attr({height : attrValue});
```

- **removeAttr()** - removes an attribute from an HTML element

```
$('#theFormField').removeAttr('height');
```

- **wrap()** - wraps the selected element in another HTML element

```
$('#someElement').wrap('<div style='border: 1px  
solid black;'></div>');
```

- **replaceWith()** - replaces the selected HTML element with a new one

```
$('#theElement').replaceWith('<div style='border: 1px solid black;'></div>');
```

- **remove()** - removes the selected HTML element from the DOM

```
$('#theElement').remove();
```

- **empty()** - removes all child elements of the selected HTML element

```
$('#theElement').empty();
```



Handling Events

Browser Events and DOM Events

Handling Events in JS

- Browsers send events to notify the JS code of interesting things that have taken place



```
<div id='text'>Some text</div>
```

```
let div = document.getElementById('text');  
div.onmouseover = function(event) {  
    event.target.style.border = '3px solid green';  
}  
div.onmouseout = function() {  
    this.style.border = ''; // this === event.target  
}
```

Event Types in DOM API

▪ Mouse events

click
mouseover
mouseout
mousedown
mouseup

▪ Touch events

touchstart
touchend
touchmove
touchcancel

▪ DOM / UI events

load
unload
resize
dragstart / drop

▪ Keyboard events

keydown
KeyPress
keyup

▪ Focus events

focus (got focus)
blur (lost focus)

▪ Form events

input
change
submit
reset

- Attach an event to an element.

```
let textbox = document.createElement('input');  
textbox.type = 'text';  
textbox.value = 'I am a text box';  
document.body.appendChild(textbox);  
  
textbox.addEventListener('focus', focusHandler);
```

- Remove an event.

```
function focusHandler(event) {  
    textbox.value = 'Event handler removed';  
    textbox.removeEventListener('focus', focusHandler);  
}
```

Multiple Events

- The **addEventListener()** method also allows you to add many events to the same element, without overwriting existing events:

```
element.addEventListener('click', function);  
element.addEventListener('click', myFunction);  
element.addEventListener('mouseover', mySecondFunction);  
element.addEventListener('mouseout', myThirdFunction);
```

- Note that you don't use the 'on' prefix for the event; use 'click' instead of 'onclick'.





Fetch API

Fetch API

- Fetch provides a generic definition of Request and Response objects
- Fetch API allows you to make network requests similar to **XMLHttpRequest** (XHR).
- The response of a **fetch()** is a Stream object.



Fetch API (Demo) (1)

```
@GetMapping('/')
public ModelAndView index(ModelAndView modelAndView) {
    modelAndView.setViewName('index');
    return modelAndView;
}

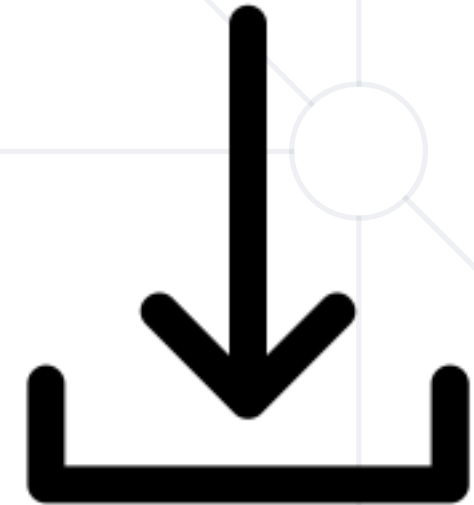
@GetMapping(value = '/fetch', produces = 'application/json')
@ResponseBody
public Object fetchData() {
    return new ArrayList<Product>() {{
        add(new Product(){
            setName('Chewing Gum');
            setPrice(new BigDecimal(1.00));
            setBarcode('133242556222');
        });
        ...
    }};
}
```

HomeController.java

```
public class Product {
    private String name;
    private BigDecimal price;
    private String barcode;

    // Getters & Setters
    ...
}
```

Product.java



- Now let's head to the view
 - There is no need for a separate .js file for one-time use

```
...
<div class='container-fluid'>
  <h1 class='text-center mt-5 display-1'>Data Fetch</h1>
  <div class='data-container mt-5'></div>
  <div class='button-holder mt-5'>
    <button id='fetch-button' class='btn btn-info'>Fetch Data</button>
    <button id='clear-button' class='btn btn-secondary'>Clear Data</button>
  </div>
</div>
<script>
  // jQuery Event handlers
  $('#fetch-button').click(() => {...}); // Fetch and render the data
  $('#clear-button').click(() => $('#data-container').empty()); // Clear the data
</script>
```

index.html

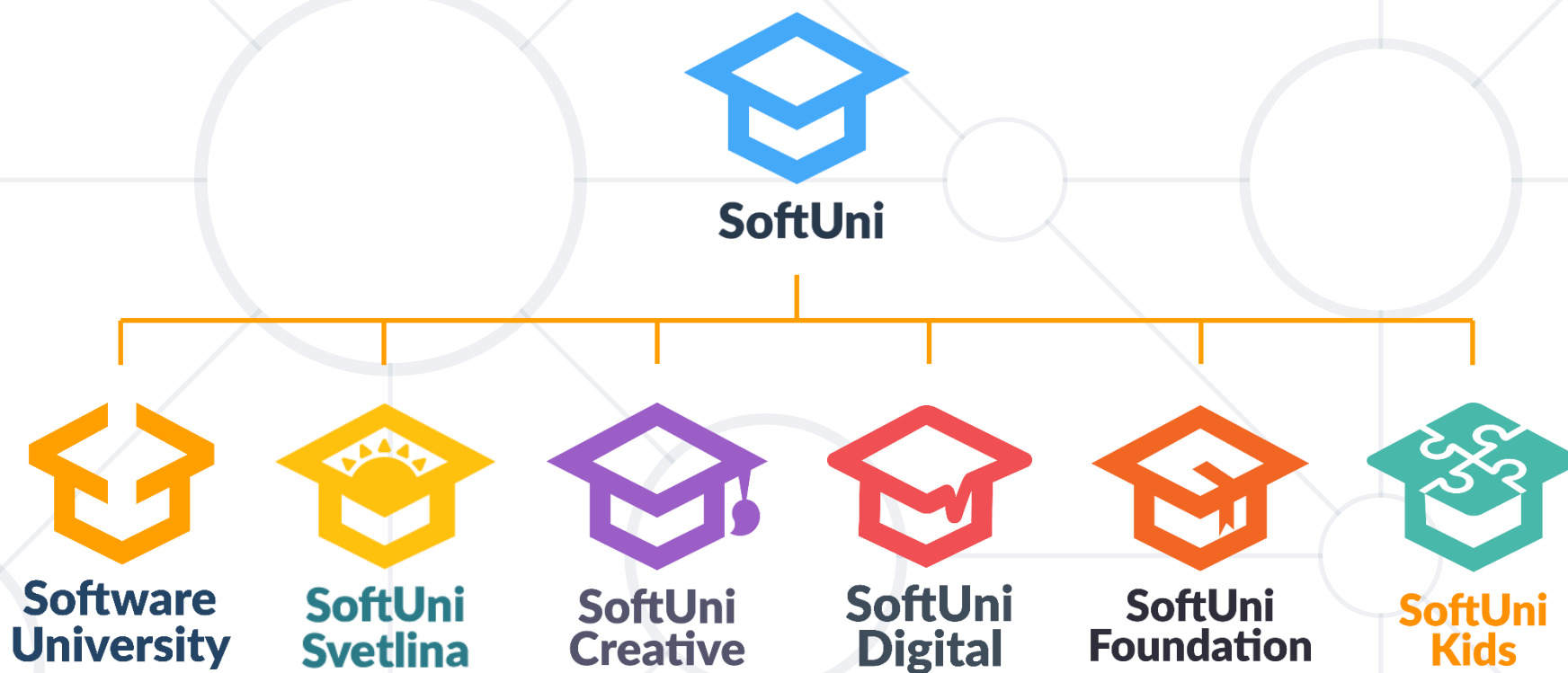
Fetch API (Demo) (3)

```
$('#fetch-button').click(() => {  
  fetch('http://localhost:8000/fetch') // Fetch the data (GET request)  
  .then((response) => response.json()) // Extract the JSON from the Response  
  .then((json) => json.forEach((x, y) => { // Render the JSON data to the HTML  
    if (y % 4 === 0) {  
      $('#data-container').append('<div class=row d-flex justify-content-around mt-4>');  
    }  
  
    let divColumn =  
      '<div class=col-md-3>' +  
      '<h3 class=text-center font-weight-bold>' + x.name + '</h3>' +  
      '<h4 class=text-center>Price: $' + x.price + '</h4>' +  
      '<h4 class=text-center>Barcode: $' + x.barcode + '</h4>' +  
      '</div>';  
  
    $('#data-container .row:last-child').append(divColumn);  
  }));  
});
```

- What is the REST Controllers
- Rest Templates
- How to manipulate DOM
 - Creating and appending html elements
- Using JQuery and Fetch



Questions?



SoftUni Diamond Partners

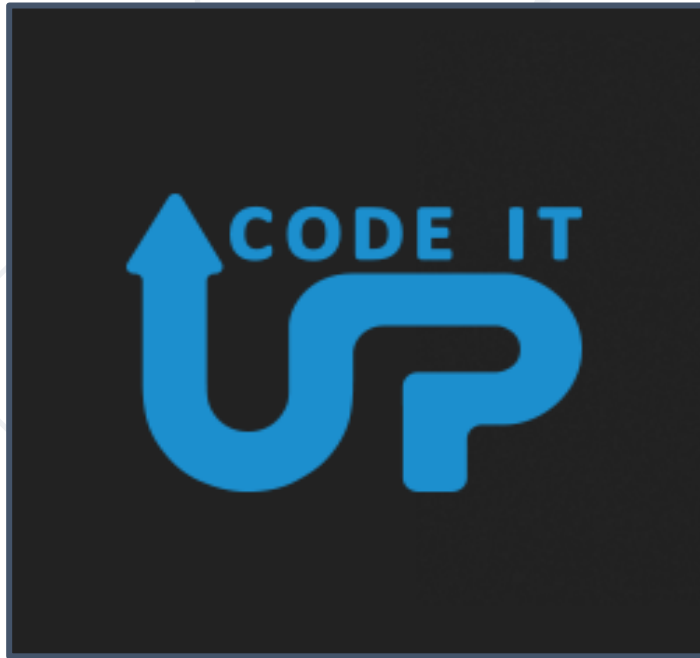


SCHWARZ



**SUPER
HOSTING
.BG**





VIRTUAL RACING SCHOOL



- Software University – High-Quality Education, Profession and Job for Software Developers
 - softuni.bg, about.softuni.bg
- Software University Foundation
 - softuni.foundation
- Software University @ Facebook
 - facebook.com/SoftwareUniversity
- Software University Forums
 - forum.softuni.bg



- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is copyrighted content
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://about.softuni.bg/>
- © Software University – <https://softuni.bg>

