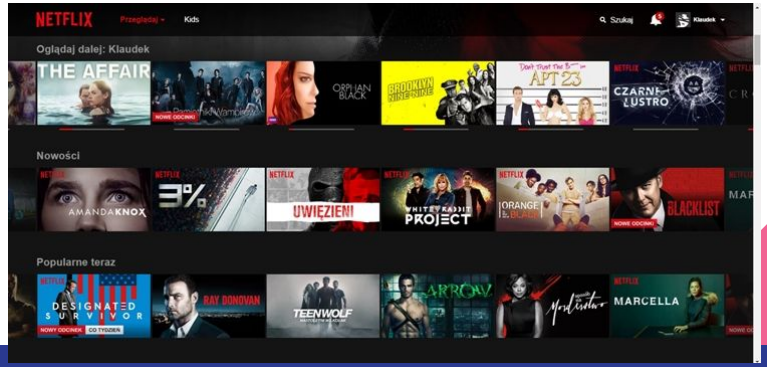
Rozdział V

Building Single-Page Web Applications with REST and JSON

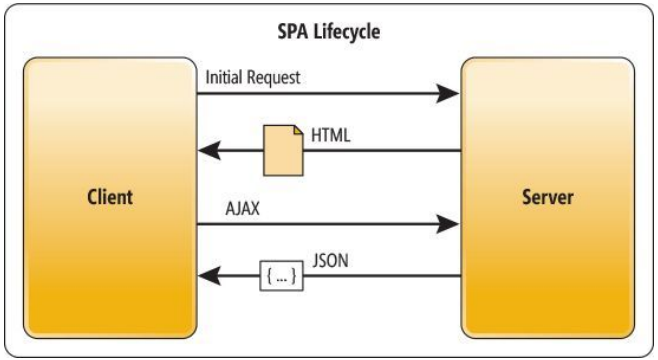
Piotr Sularz
Iwona Poręba
Artur Michura
Konrad Konicki
Jan Wiśniewski
Tomasz Mysłajek
Radosław Kraj
Krzysztof Michalski

Single-Page Web Applications (SPAs)

- dla całej aplikacji ładowana jest tylko jedna strona
- interakcja z użytkownikiem oraz logika frontendowa realizowana za pomocą JavaScript
- komunikacja między stroną a serwerem odbywa się za pomocą AJAX (Asynchronous JS and XML) z wykorzystaniem formatowania JSON
- wiele frameworków JSowych usprawniających logikę front-endu (AngularJS, React, Vue.js …)
- przykładowe SPA:
  - Gmail
  - Netflix
  - Google Maps



Single-Page Web Applications



W jednostronicowej aplikacji internetowej (SPA) mamy dwa rodzaje danych:

1) **zawartość statyczną**, obejmującą strony HTML, arkusze stylów, obrazy i pliki skryptów
2) **zawartość dynamiczną**, czyli dane dynamiczne w formacie JSON.

Ważne jest, aby wiedzieć, że w SPA wszystko jest obsługiwane przez interfejs REST, nawet zawartość statyczna.

Jeśli wpiszemy w przeglądarce adres URL:

-> do serwera wysyłane jest polecenie **GET/some/url/path**, a to samo polecenie GET z różnymi ścieżkami wysyłane jest dla arkuszy stylów, skryptów i obrazów, które strona chce załadować.

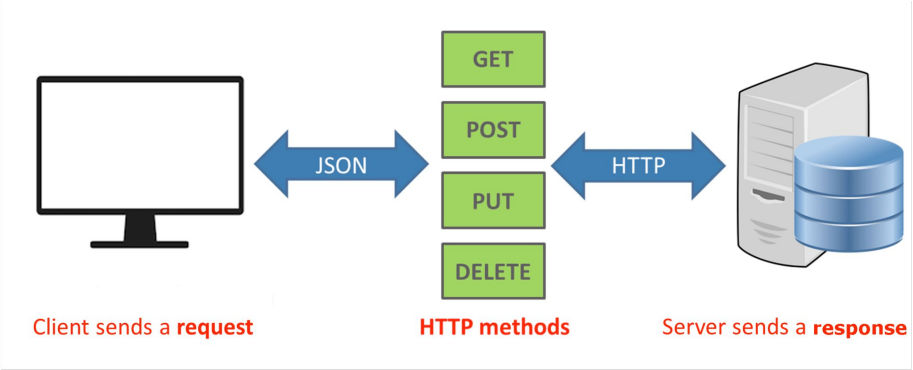
-> Jeśli przesyłamy formularz, wysyłane jest polecenie **POST/some/url/path** z przesłanymi danymi jako treść wiadomości (message body).

JSF vs REST

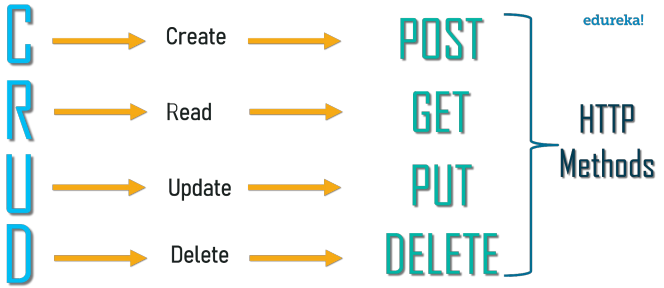
W środowisku JSF (JavaServer Faces) wszystkie GET-y i POST-y oraz nawigacja po stronie mogą być obsługiwane przez JSF, ale w przypadku interfejsu REST cofamy się o krok i w sposób jawny zajmujemy się GET-ami i POST-ami, a nawigację po stronie omijamy, ładując tylko jedną statyczną stronę HTML. Dynamiczna zawartość strony jest obsługiwana przez JavaScript.

REST

Jest to styl architektoniczny dla operacji internetowych. Klienci używają predefiniowanego zestawu operacji lub metod HTTP na danych — GET, POST, PUT, DELETE (oraz innych) — do komunikacji z serwerami. Komunikacji jest bezstanowa - natychmiast po wykonaniu przez serwer operacji i/lub zwróceniu danych, serwer zapomina o kroku komunikacji.



Communication verbs



- GET - pobiera określony zasobu według podanego identyfikator. Nie ma możliwości zmiany danych, tylko pobrania ich.
- DELETE - usuwa określony zasób według identyfikatora. W przypadku wykonania tego samego zasobu drugi raz nie wykonywana jest żadna akcja - jest ignorowana.
- POST - tworzy nowy zasób. Wykorzystywany w formularzach, dane przesyłane są zazwyczaj za pomocą JSON lub XML.
- PUT - aktualizuje dany zasób na podstawie podanego identyfikatora. Jeśli nie ma takiego zasobu to zachowuje się jak POST i tworzy nowy zasób.

Others verbs

HEAD - do pobierania metadanych o zasobie (informacje o nim, ale nie sam zasób)

TRACE - aby zobaczyć, co dzieje się z danymi w drodze na serwer.

PATCH - aktualizacja części danych

OPTIONS - sprawdzenie udostępnionych metod.

CONNECT - stworzenie tunelu poprzez serwery proxy

JSON

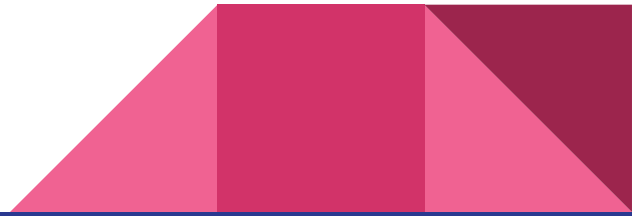
- W aplikacjach REST nie ma wprost określonego formatu przesyłania danych między klientem a serwerem.
- Dla stron internetowych używa się formatu HTML, a inne pliki takie jak zdjęcia czy skrypty są przesyłane w oryginalnej formie.
- Jednak dla danych tekstowych współcześnie najczęściej stosowanym formatem jest JSON, czyli format oparty na zapisie obiektów w języku JavaScript.

Notacja JSON

- Obiekty są głównymi elementami notacji JSON. Przechowują dane w formie par klucz-wartość, czyli własności obiektu.
  Przykład: `{ [własność 1], [własność 2], … }`
- Własność obiektu jest zapisywana w następujący sposób:
  `"klucz" : [wartość]` gdzie wartość może być ciągiem znaków, liczbą całkowitą lub zmiennoprzecinkową, wartością logiczną, innym obiektem lub tablicą albo wartością pustą `null`.
- Tablica jest kolekcją innych elementów, które mogą mieć różne typy danych i jest zapisywana w następujący sposób: `[element1, element2, ...]`

Przykład obiektu JSON

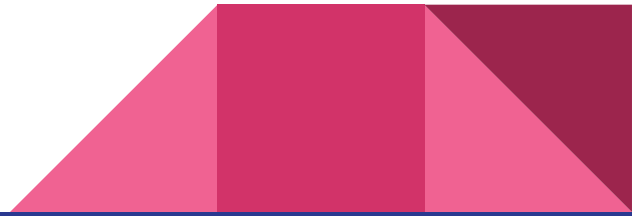
```
{
   "ID" : 5616,
   "name" : "John Doe"
   "children" : [
{ "name": "Sue Ann Doe" },
{ "name": "Patt Doe" }
   ],
   "weight": 145.5,
   "spouse": true,
   "cars": null,
   "record": [ 12734, "QBA", true ]
 }
```



Pliki statyczne

Pliki służące do personalizowania wyglądu strony, jej zachowań ale również do załączania obrazów, pdf itp.

- **html** (src/main/webapp/static/)
- **css** (src/main/webapp/static/css)
- **js** (src/main/webapp/static/js)



Linkowanie plików statycznych

```
...
<head>
  <meta charset="UTF-8">
  <title>RESTful Dates</title>
  <link rel="stylesheet" type="text/css"
        href="css/styles.css" />
  <script src="js/jquery-3.3.1.min.js"></script>
</head>
...
```

Przykład - część I

```
...
 <body>
     Date format: <input id="dateFormat" type="text" />
     <div class="clearfloat"></div>
     <button id="submitButton">Get date!</button>
     <div class="clearfloat"></div>
     <div id="errOutput" class="err"></div>
     <div class="clearfloat"></div>
     <div id="dateOutput"></div>
 </body>
 ...
```

Przykład - część II

```
/**
 * REST Web Service
 */
@Path("/")
public class RestDate {
    @GET
    @Path("date")
    @Produces("application/json")
    public Response date(
            @QueryParam("dateFormat") @DefaultValue("")
            String dateFormat) {
        ZonedDateTime zdt = ZonedDateTime.now();
        String outStr = "";
        String errMsg = "";
        try {
            outStr = ("".equals(dateFormat) ?
               zdt.toString() :
               zdt.format(DateTimeFormatter.
                    ofPattern(dateFormat)));
            errMsg = "";
        } catch(Exception e) {
// ************
}
```

Przykład - część III

```
$(function() {
      $('#dateFormat').val("yyyy-MM-dd HH:mm:ss.SSSXXX");
      $('#submitButton').click(function(){
      var fmt = $('#dateFormat').val();
      var url = "../webapi/date";
      $.ajax({
      method: "GET",
      url: url,
      data: { dateFormat: fmt }
      })
      .done(function(msg) {
      $('#errOutput').html(msg.errMsg);
      $('#dateOutput').html("Current date/time: " +
      msg.date);
      })
      .fail(function(jqXHR, textStatus, errorThrown) {
 //*********************************************//
});});})
```

DZIĘKUJEMY ZA UWAGĘ