Jakarta EE
Aplikacje SPA wykorzystujące REST i JSON

...

Natalia Bidzińska
Rafał Czajka
Jan Czerlunczakiewicz
Patryk Ernest
Grzegorz Karkowski
Maciej Kobiec

Jakarta EE

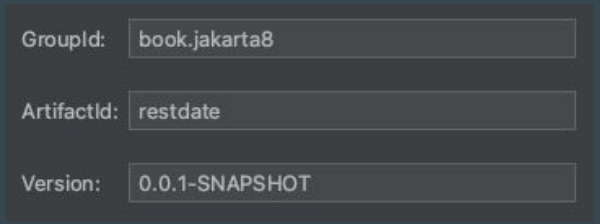
- Platforma oprogramowania zbudowana jako narzędzie do tworzenia oprogramowania biznesowego.
- Pojawiła się w 2019 r., bezpośrednio nawiązywała i była kompatybilna z Java EE 8.
- Po przejęciu platformy przez Eclipse zmieniła nazwę na Jakarta i od wersji Jakarta EE 9 straciła wsteczną kompatybilność.
- W praktyce jest to zbiór specyfikacji odpowiedzialnych za m. in. komunikację z bazą danych, łączność sieciową czy wstrzykiwanie zależności.
- Specyfikacje wchodzące w skład platformy to m. in. *Servlet, ServerPages, JSON Processing, Bean Validation, Persistence*. Pełna lista znajduje się tutaj.

Serwer

- Jakarta EE daje możliwość współpracy z licznymi serwerami, np. *Apache Tomcat, Jelly, Glassfish, Wildfly*.
- Referencyjną implementacją jest w tym przypadku *Glassfish 6*.
- Serwer ma tworzyć środowisko pozwalające na uruchomienie poszczególnych części składowych platformy.
- Umożliwia połączenie z bazą danych, uwierzytelnianie oraz autoryzowanie informacji, obsługę kolejek komunikatów.

RESTful Server w Jakarta EE - konfiguracja

1. Utwórz projekt, wybierając opcję *Maven Project*. W konfiguracji ustaw przedstawione obok dane.
2. Upewnij się, że używasz Java JDK w wersji 8.
3. Otwórz plik *pom.xml*, dodaj odpowiednie zależności i sprawdź, czy zawartość pliku jest zgodna z zaprezentowaną.




```
<dependency>
    <groupId>javax</groupId>
    <artifactId>javaee-api</artifactId>
    <version>8.0</version>
</dependency>
```

```
<project ...>
    <build>
        <plugins>
            <plugin>
                <artifactId>maven-compiler-plugin</artifactId>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

RESTful Server w Jakarta EE - konfiguracja

4. Przekonwertuj projekt, wybierając opcję *Configure → Convert to Faceted Form.*
5. Upewnij się, że w projekcie istnieje folder *src/main/webapp/WEB-INF* i zawiera pliki *glassfish-web.xml, beans.xml, web.xml.*
6. Utwórz klasę *RestDate.*
7. Uruchom serwer, zainstaluj na nim aplikację wybierając opcję *Run as → Run on Server* i uruchom w terminalu polecenie
   `curl -X GET http://localhost:8080/restdate/webapi/d`
8. Gotowe! W odpowiedzi powinieneś otrzymać aktualną datę i czas ze strefą czasową.

```java
package book.jakarta8.restdate;
import java.time.ZonedDateTime;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;

/**
 * REST Web Service */
@Path("/d")
public class RestDate {
    @GET
    @Produces("text/plain")
    public String stdDate() {
        return ZonedDateTime.now().toString();
    }
}
```

Single-Page Web Application

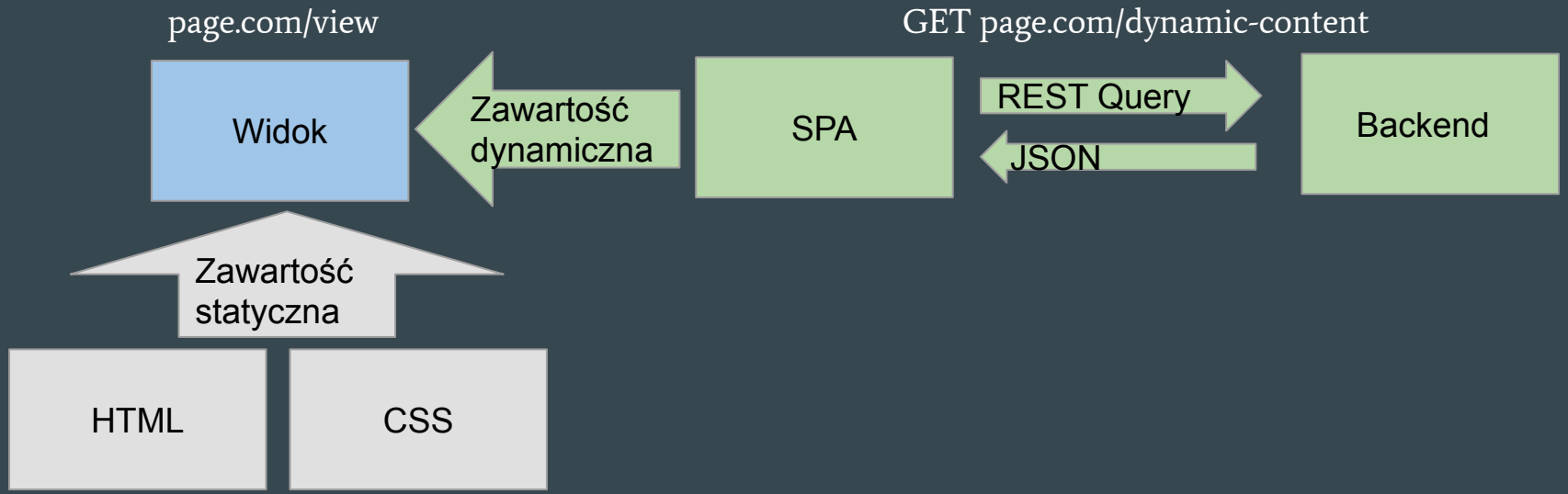
Wprowadzenie do zagadnienia

Akronim SPA oznacza “Single Page (Web) Application”, czyli współczesne podejście do tworzenia elastycznych i przyjemnych w użytkowaniu aplikacji.

Aplikacja zawarta jest w jednym pliku .HTML która dynamicznie za pomocą skryptów wykonywanych po stronie klienta steruje wyświetlaną zawartością.

Dzięki takiemu podejściu można wydajnie optymalizować ogólną wydajność aplikacji, a skalowalność aplikacji wzrasta.

Przykład działania



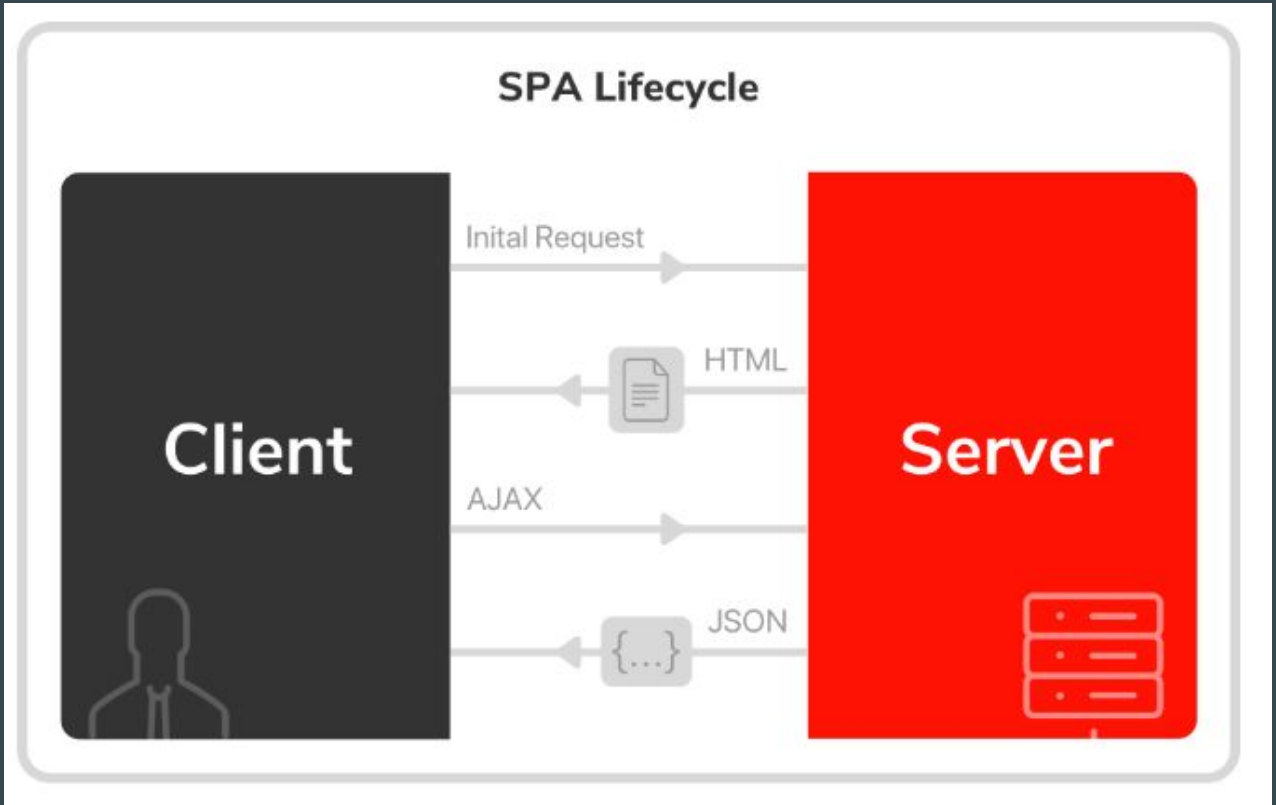
Dlaczego SPA jest popularne?

- W odróżnieniu od MPA (Multiple Pages Application), przy wejściu w aplikację nie ma potrzeby pobierania całego dokumentu. Wystarczy pobrać statyczną treść, a następnie sterować nią i doładowywać zawartość za pomocą skryptów.
- Z punktu widzenia użytkowania takiego interfejsu wrażenia i odczucia są dużo płynniejsze.
- Dzięki korzystaniu z asynchronicznych metod komunikacji z serwerem można zastosować różne sztuczki UX/UI, jak np. wyświetlenie spinnera na czas ładowania listy.
- Utrzymanie, debugowanie i rozwój projektu jest łatwiejszy dla developerów tak zrealizowanego systemu.

Krótki rys historyczny

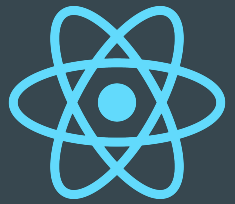
- Powstają pierwsze SPA jako aplikacje Flash i aplety języka Java.
- W 2006 wydany zostaje Asynchroniczny JavaScript i XML (AJAX), który rewolucjonizuje możliwości tworzenia aplikacji webowych.
- W niedalekiej przyszłości świat FrontEnd developerów wzbogaci się o javascriptowe frameworki, ułatwiające wytwarzanie coraz to lepszego jakościowo oprogramowania.
- Nowe możliwości zmieniły również panujące trendy w projektowaniu stron pod kątem UX/UI. Coraz większy nacisk kładzie się na optymalizację procesów oraz komfort i wrażenia użytkownika końcowego.
- W końcu, SPA staje się najczęściej preferowanym podejściem do wytwarzania aplikacji webowej.

Schemat działania SPA



SPA - popularne biblioteki i frameworki

SPA po dziś dzień jest najpopularniejszym sposobem tworzenia stron i aplikacji internetowych. Wpływ na to miało powstanie wielu bibliotek i frameworków przeznaczonych do ich budowania. Najpopularniejszymi z nich są:



React

Vue

Angular

Svelte

REST

- Nie jest protokołem!!
- Jest to zbiór dobrych praktyk tworzenia architektury aplikacji rozproszonych (wzorzec).
- Najczęściej kojarzony z HTTP, choć nie jest z nim ściśle związany.
- Nie jest standardem, lecz zbiorem ograniczeń.

REST ogólne zasady

Zasoby	Wyeksponowane przez łatwo zrozumiałe identyfikatory katalogów (URI)
Obiekty	Reprezentowane i przekazywane w postaci JSON
Zapytania	Wykorzystują jawnie metody HTTP (GET, POST, PUT, DELETE)
Interakcje	Są całkowicie bezstanowe. Kontekst klienta nie jest przechowywany na serwerze między żądaniami

Zasady tworzące REST

1. Odseparowanie interfejsu użytkownika od operacji na serwerze
2. Bezstanowość
3. Cacheability
4. Endpointy
5. Separacja warstw

Metody uwierzytelniania w REST API

- Basic
- API Key
- OAuth

Metoda uwierzytelniania - Basic - szczegóły

username : password

B64 encode

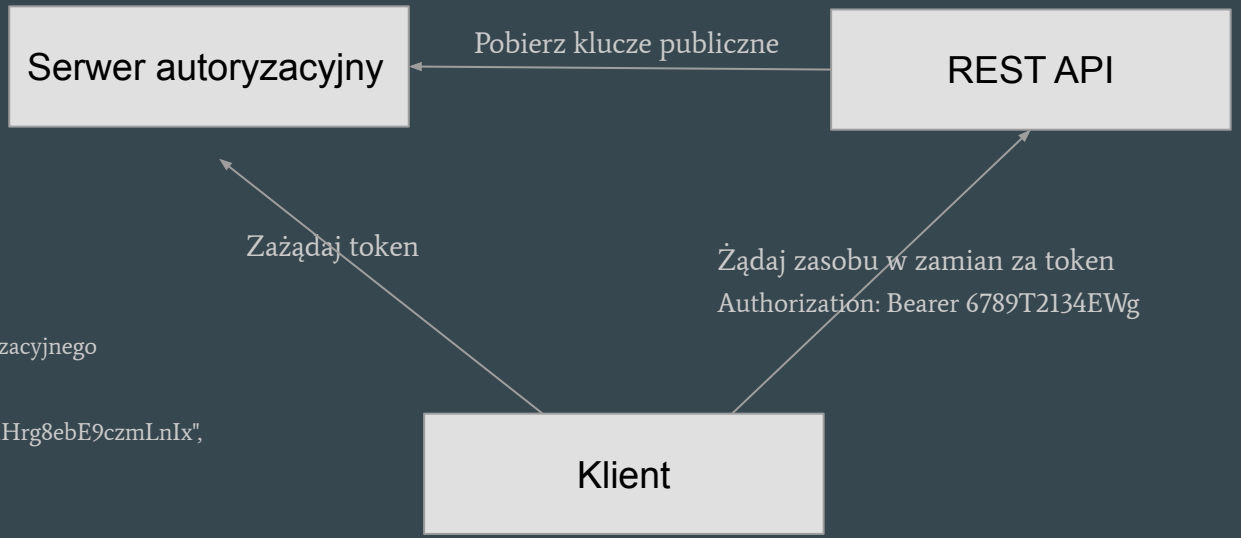
Authorization: Basic bG9sOnNlY3VyZQ==

Metoda uwierzytelniania - Klucz Api

Unikalny Klucz

Authorization: Apikey
6789T2134EWgvbvbiosgnioIO

Metoda uwierzytelniania - Oauth2



Przykładowa odpowiedź z serwera autoryzacyjnego
HTTP/1.1 200 OK


```
  {
   "access_token": "eyJbGciOiJSUzI1N....3c1Hrg8ebE9czmLnIx",
   "expires_in": 900,
   "refresh_expires_in": 1800,

"refresh_token":"7J9WwnrraQ...F8DLI6mmWeMJJ7LlSbOR”,
   "token_type": "bearer",
   "not-before-policy": 0,
   "session_state": "cf158946-eb2b-4daa-869f-fec7eb6eb015",
   "scope": ""
 }
```

REST Wady

- Bezstanowość
- Redundancja przesyłanych informacji

REST Zalety

- Łatwy w implementacji
- Używa pamięci podręcznej
- Wymusza serializację danych formatem XML lub JSON
- Skalowalność

JSON

- JSON (ang. JavaScript Object Notation) to format zapisu i wymiany danych.
- Jest to format tekstowy. Uważany jest za format czytelny zarówno dla maszyn, jak i dla ludzi.
- Jeżeli dane są przesyłane za pomocą protokołu http, to nagłówek Content-Type: application/json informuje nas, że w body są przesyłane dane w tym formacie.

Prawidłowy dokument w formacie JSON

```
{
    “name”:”Jan”,
    “surname”:”Kowalski”,
    “client_id”:1001,
    “age”:55,
    “is_admin”:true
}
```

Zalety formatu JSON

- Czytelny zarówno dla maszyn, jak i dla ludzi.
- Chroni przed atakami CSRF.
- Zajmuje mniej miejsca (używanych jest mniej znaków, niż np. xml).
- Łatwo przedstawić wartość null oraz bool.
- Łatwy do parsowania.

Asynchroniczne zapytania do serwera - historia

SSR

Renderowanie strony HTML po stronie serwera i przesyłanie gotowego pliku do przeglądarki

VS

SPA

Asynchroniczne zapytania do serwera pobierające i uaktualniające tylko część zawartości strony

Asynchroniczne zapytania do serwera - Ajax vs Fetch

Ajax

Jeden z pierwszych sposobów na asynchroniczne pobieranie zawartości stron internetowych

Fetch

Obecnie JavaScript zawiera natywne API do zapytań asynchronicznych o nazwie *Fetch*

Asynchroniczne zapytania do serwera - Ajax vs Fetch

Ajax

```
let xhr = new XMLHttpRequest();
xhr.open('GET', 'path');

xhr.onreadystatechange = function () {
    const DONE = 4;
    const OK = 200;
    if (xhr.readyState === DONE) {
        if (xhr.status === OK) {
            console.log(xhr.responseText);
        } else {
            console.log('Error: ' + xhr.status);
        }
    }
};

xhr.send(null);
```

Fetch

```
async function receiveData() {
    try {
        const res = await fetch('path');
        const data = await res.text();
        console.log(data);
    } catch (error) {
        console.log('Error:' + error);
    }
}

receiveData();
```

Asynchroniczne zapytania do serwera - Axios

Obecnie bardzo popularnym rozwiązaniem jest biblioteka Axios. Jest to alternatywa dla Fetch API. Nie jest to natywny sposób wykonywania zapytań, udostępnia jednak bardzo proste w obsłudze API i posiada wiele pomocnych funkcjonalności takich jak: interceptory, anulowanie zapytań czy globalna konfiguracja.

```
axios.get('https://api.example.com/data')
    .then((response) => {
        console.log(response.data);
    });
```

```
axios.post('/login', {
        firstName: 'Finn',
        lastName: 'Williams'
    })
    .then((response) => {
        console.log(response);
    }, (error) => {
        console.log(error);
    });
```

Czy każda aplikacja zawarta na jednej stronie to SPA?

Chociaż w pierwszej chwili można by pomyśleć, że to prawda, to jednak tak nie jest.

Zdarza się, że SPA mylona jest z witrynami typu “one page”, czyli takimi, które pozbawione są nawigacji oraz podstron, a sama zawartość umieszczona jest w segmentach, do których można ewentualnie scrollować.

Single Page Application umożliwia dodanie podstron, które znajdować się będą pod innym URL’em, co w przypadku rozbudowanych aplikacji ułatwia nawigację.

Zalety Single Page Application

Z punktu widzenia użytkownika:

- Możliwość wyświetlenia splashscreenów, spinnerów, skelletonów etc. w oczekiwaniu na dane,
- Zwiększona wydajność gdy podstawowe dane zostaną załadowane.

Z punktu widzenia dewelopera:

- Separacja warstwy klienckiej od serwerowej,
- Odciążenie serwera,
- Możliwość migracji na inny typ aplikacji.

Popularne serwisy zbudowane w oparciu o SPA

- Facebook
- YouTube
- Twitter
- GitHub
- Gmail
- Google Maps
- Airbnb
- Netflix
- Pinterest
- Paypal







Dziękujemy za uwagę!

...

Natalia Bidzińska
Rafał Czajka
Jan Czerlunczakiewicz
Patryk Ernest
Grzegorz Karkowski
Maciej Kobiec