

## Program szkolenia:

# REST i Microservices w PHP

### Informacje:

<b>Nazwa:</b>	<b>REST i Microservices w PHP</b>
<b>Kod:</b>	<b>PHP-rest</b>
<b>Kategoria:</b>	PHP
<b>Grupa docelowa:</b>	developerzy
<b>Czas trwania:</b>	4 dni
<b>Forma:</b>	50% wykłady / 50% warsztaty

---

Szkolenie stanowi syntezę wzorców, technik i narzędzi potrzebnych aby od początku do końca pracować nad kodem systemów projektowanych w architekturze microservices.

Niemal wszystkie frameworki nie narzucają struktury organizacji kodu. Dlatego proponowalibyśmy rozpoczęcie implementacji z wykorzystaniem Symfony i jego komponentów. Następnie, gdy zespół będzie miał praktyczne doświadczenie i wiedzę odnośnie potrzebnych narzędzi, użycie innych narzędzi (np. aplikacji Symfony opartych i micro-kernele) będzie znacznie prostsze.

### Zalety szkolenia:

- Zintegrowane podejście do: skalowania, bezpieczeństwa, failover, CQRS
- Całościowe i szersze spojrzenie na problemy
- Dostęp do wiedzy eksperckiej

## Szczegółowy program:

### 1. REST / HTTP API

#### 1.1. Wprowadzenie

##### 1.1.1. Request

##### 1.1.2. Response

#### 1.2. Nazewnictwo

##### 1.2.1. Rzeczowniki

##### 1.2.2. Czasowniki

##### 1.2.3. Liczba pojedyncza / liczba mnoga

#### 1.3. Formaty API

##### 1.3.1. JSON

##### 1.3.2. XML

#### 1.4. Metody HTTP

##### 1.4.1. GET

##### 1.4.2. POST

##### 1.4.3. PUT

##### 1.4.4. DELETE

##### 1.4.5. PATCH

#### 1.5. Typowe zagadnienia

##### 1.5.1. Lokalizacja API

##### 1.5.2. Wersjonowanie API

##### 1.5.3. Filtrowanie zasobów

##### 1.5.4. Partial response

##### 1.5.5. Stronicowanie

1.5.6. Sortowanie

1.5.7. Obsługa błędów

1.6. Bezpieczeństwo

1.6.1. Logowanie

1.6.2. Uwierzytelnianie

1.6.3. Rate limits

1.7. Cache

1.7.1. Wprowadzenie

1.8. Klienci API

## 2. Mikroserwisy

2.1. Wprowadzenie

2.1.1. Prawo Conwaya

2.1.2. Charakterystyka mikroserwisów

2.1.3. Korzyści i koszty

2.2. Modelowanie

2.2.1. Założenia

2.2.1.1. Decentralizacja zarządzania danymi

2.2.2. Podejście bounded-context

2.2.3. Antywzorce mikroserwisów

2.2.3.1. Nanoservice

2.3. Komunikacja

2.3.1. Komunikacja synchroniczna

2.3.2. Komunikacja asynchroniczna

2.3.3. Formaty

2.4. Security

## 2.5. Infrastruktura

### 2.5.1. Deployment

### 2.5.2. Skalowanie

### 2.5.3. Osiąganie wysokiej dostępności

### 2.5.4. Monitoring działania

## 2.6. Testowanie

## 2.7. Dokumentowanie mikroserwisów

# 3. Frameworki

## 3.1. Wprowadzenie do Symfony

### 3.1.1. Dlaczego używać Symfony?

#### 3.1.1.1. Jakie problemy rozwiązuje?

#### 3.1.1.2. Ekosystem

### 3.1.2. Jak Symfony może pomóc?

#### 3.1.2.1. Symfony Framework

#### 3.1.2.2. Komponenty Symfony

## 3.2. Podstawy Symfony

### 3.2.1. Tworzenie projektów

#### 3.2.1.1. Instalator Symfony

#### 3.2.1.2. Composer

#### 3.2.1.3. Sprawdzanie wymagań środowiska

### 3.2.2. Konfiguracja serwera HTTP

#### 3.2.2.1. Nginx

#### 3.2.2.2. Apache

#### 3.2.2.3. Wbudowany w PHP serwer HTTP

### 3.2.3. Struktura projektów

3.2.3.1. Aplikacje

3.2.3.2. Środowiska

3.2.3.3. Pliki uruchamialne

3.2.3.4. Bundle

3.2.3.5. Cache

3.2.3.6. Logi

3.2.3.7. Zależności

3.2.4. Konfiguracja

3.2.4.1. PHP

3.2.4.2. YML

3.2.4.3. Adnotacje

3.2.4.4. XML

3.2.5. Web Debug Toolbar

3.3. Cykl życia aplikacji

3.3.1. Http Kernel

3.3.1.1. Request

3.3.1.2. Response

3.3.1.3. PSR-7

3.3.1.4. App Kernel

3.3.1.5. Multi Kernel

3.3.1.6. Micro Kernel

3.3.2. Kontrolery

3.3.2.1. Konwencje i dobre praktyki

3.3.2.2. Przekierowania

3.3.2.3. Strony błędów

#### 3.3.2.4. Konwertery Parametrów

### 3.3.3. Routing

#### 3.3.3.1. Konfiguracja

#### 3.3.3.2. Konwencje nazewnictwa

#### 3.3.3.3. Wymagania

#### 3.3.3.4. Debugowanie

### 3.4. Formularze, Walidacja i Tłumaczenia

#### 3.4.1. Formularze

##### 3.4.1.1. Tworzenie formularzy

##### 3.4.1.2. Obsługa wysłanych danych

#### 3.4.2. Walidacja

##### 3.4.2.1. Konfiguracja

##### 3.4.2.2. Ograniczenia

##### 3.4.2.3. Grupy

##### 3.4.2.4. Funkcje walidujące

##### 3.4.2.5. Walidacja formularzy

### 3.5. Wstrzykiwanie zależności

#### 3.5.1. Wprowadzenie

##### 3.5.1.1. Dlaczego powinniśmy wstrzykiwać zależności?

##### 3.5.1.2. Wstrzykiwanie przez setter

##### 3.5.1.3. Wstrzykiwanie przez konstruktor

#### 3.5.2. Kontener usług

##### 3.5.2.1. Konfiguracja

##### 3.5.2.2. Definiowanie usług

##### 3.5.2.3. Debugowanie

3.5.2.4. Tagi i zaawansowane modyfikowanie konfiguracji kontenera

3.6. HTTP Cache

3.6.1. Wprowadzenie

3.6.1.1. Typy Cache

3.6.1.2. Nagłówki HTTP

3.6.1.3. Modele Walidacji i Wygasania

3.6.1.4. Edge Side Includes

3.6.2. Cache w Symfony

3.6.2.1. Http Kernel

3.6.2.2. App Cache

3.6.2.3. Edge Side Includes

3.7. Logowanie i uwierzytelnianie

3.7.1. Komponent Security

3.7.1.1. Konfiguracja

3.7.1.2. Logowanie

3.7.1.3. Uwierzytelnianie

3.7.1.4. Zabezpieczanie kontrolerów

3.7.2. Zaawansowane zagadnienia komponentu Security

3.7.2.1. Access Control Lists

3.7.2.2. Security Voters

3.7.2.3. Logowanie bezstanowe

3.8. Testowanie, wydajność

3.8.1. Testowanie

3.8.1.1. Web Test Case w PHPUnit

3.8.1.2. Test Client w PHPUnit

3.8.1.3. Dostęp do kontenera usług w testach

3.8.1.4. Korzystanie z profilera w testach

3.8.1.5. Behat extension

3.8.2. Wydajność

3.8.2.1. Dobre praktyki

3.8.2.2. Optymalizacja Composer'a

3.8.2.3. Profilowanie aplikacji