

## Program szkolenia:

# Tworzenie aplikacji w Ruby on Rails z wykorzystaniem zwinnych metodyk

## Informacje:

<b>Nazwa:</b>	<b>Tworzenie aplikacji w Ruby on Rails z wykorzystaniem zwinnych metodyk</b>
<b>Kod:</b>	<b>RoR-Rails</b>
<b>Kategoria:</b>	Ruby on Rails
<b>Grupa docelowa:</b>	developerzy
<b>Czas trwania:</b>	4 dni
<b>Forma:</b>	30% wykłady / 70% warsztaty

Podczas tego szkolenia uczestnicy dowiedzą się, w jaki sposób szybko tworzyć wysokiej jakości aplikacje webowe w oparciu o technologię Ruby on Rails.

Poznają oni kluczowe możliwości najnowszej wersji Rails - 3.2, jak również elementy języka Ruby w wersji 1.9 i nauczą się, jak wykorzystać tę wiedzę do tworzenia aplikacji, krok po kroku - od idei do wdrożenia.

W oparciu o tworzoną od podstaw aplikację uczestnicy poznają najważniejsze cechy tej technologii.

Szkolenie ma formę wykładów zakończonych warsztatami, główny nacisk położony jest na ćwiczenia praktyczne.

W procesie tworzenia projektu przedstawione zostaną elementy zwinnych (agile) metod tworzenia oprogramowania oraz praktyk takich jak refaktoring, DRY i innych.

## Zalety szkolenia:

- Nauka poprzez ćwiczenia praktyczne
- Poznanie pełnego procesu wytwarzania oprogramowania w oparciu o metodyki zwinne (TDD, BDD, automatyzacja)
- Zawiera elementy Domain Driven Design
- Poznanie narzędzi i bibliotek, które są obecnie na preferowane przez programistów Ruby on Rails
- Szybkie tworzenie w pełni testowalnych aplikacji internetowych

## Szczegółowy program:

### 1. Wstęp

#### 1.1. Środowisko Developerskie

##### 1.1.1. Instalacja - Windows

##### 1.1.2. Instalacja - Linux

##### 1.1.3. RubyMine IDE

### 2. Aplikacje webowe

#### 2.1. Lightweight vs enterprise

### 3. Zwinne projektowanie

#### 3.1. Podejście Agile

#### 3.2. User Story

#### 3.3. Interakcja z klientem

#### 3.4. Automatyzacja testów

##### 3.4.1. Jednostkowe

##### 3.4.2. end2end

#### 3.5. Zwinne modelowanie

##### 3.5.1. Elementy Domain Driven Design

##### 3.5.1.1. Building Blocks

### 4. Język Ruby

#### 4.1. Filozofia

#### 4.2. Różnice pomiędzy dynamicznie a statycznie typowanymi językami programowania

#### 4.3. Podstawowe konstrukcje języka (Metody, Klasy, Moduły, Bloki, Sterowanie, Wyrażenia regularne, Yield)

#### 4.4. Podstawowe typy (Teksty, Liczby, Tablice, Mapy)

#### 4.5. Programowanie zorientowane obiektowo

4.6. Metaprogramowanie

4.7. Podejście funkcyjne

4.8. Idiomy języka Ruby

4.9. Wzorce

4.9.1. Wzorce zawarte w składni

4.9.2. Best practices

## 5. Framework Rails

5.1. Podstawy

5.1.1. Architektura aplikacji

5.1.2. Architektura Model View Controller

5.1.3. RESTful

5.1.4. Rails Models and Action Pack

5.2. Pierwsza aplikacja

5.2.1. Komendy linii poleceń

5.2.2. Generatory (scaffolding)

5.2.3. Zasada "konwencja ponad konfiguracją"

5.3. Model

5.3.1. Rails ActiveRecord

5.3.2. Zarządzanie schematem bazy danych przy pomocy migracji

5.3.3. ORM - walidacja, callbacki

5.3.4. Zasięgi (scope)

5.3.5. Techniki optymalizacyjne

5.3.6. Zasady designu obiektowego

5.3.6.1. Enkapsulacja

5.3.6.2. DRY

5.3.6.3. Low Coupling

5.3.6.4. High Cohesion

5.4. Widoki

5.4.1. Rails ActionView

5.4.2. Layouty, partiale, helpery

5.4.3. Formularze

5.4.4. HTML (silnik szablonów: ERB/ HAML - do wyboru)

5.4.5. CSS (Sass)

5.4.6. Javascript

5.4.6.1. JQuery

5.4.7. Coffeescript

5.4.7.1. Ajax

5.4.7.2. Wydajność

5.4.8. Best Practices

5.5. Kontrolery

5.5.1. Rails ActionController

5.5.2. Wzorce Decorator/Presenter i Facade

5.5.3. Zasada 'skinny controller, fat model'

5.5.4. Obsługa różnych formatów (xml, json, csv, pdf)

5.6. Routing

5.6.1. RESTful urls

5.6.2. Tworzenie zdalnego API

5.7. Rozsyłanie maili

5.7.1. Rails ActionMailer

5.8. Zadania w tle

5.8.1. delayed\_job

5.9. Autentykacja i bezpieczeństwo

5.9.1. Bezpieczne mechanizmy Rails

5.9.2. Zarządzanie użytkownikami przy pomocy devise

5.9.3. Kontrolowanie dostępu

5.10. Narzędzia

5.10.1. assets pipeline - kompilator statycznych zasobów

5.10.2. Bundler - zarządzanie zależnościami

5.10.3. Rake - automatyzacja procesu budowania

## 6. Testowanie

6.1. Testowanie jednostkowe

6.1.1. Zalety testowania w izolacji

6.1.2. Podstawowe pojęcia i techniki

6.1.3. Test-Driven Development

6.1.3.1. Narzędzia wspomagające TDD: guard, spork

6.1.4. Testowanie JavaScript

6.2. RSpec

6.2.1. Testowanie modeli

6.2.2. Testowanie kontrolerów

6.2.3. Testowanie widoków

6.2.4. Testowanie route'ów

6.2.5. Przygotowanie stanu (fixture, factory)

6.2.6. Mockowanie

6.3. Testowanie integracyjne/akceptacyjne

6.3.1. Metodyka Behavior-Driven Development

6.3.1.1. Technika modelowania wymagań poprzez User Story

6.3.2. Scenariusze akceptacyjne

6.3.3. Współpraca z Ekspertem Domenowym

6.3.4. Cucumber - narzędzie wspierające BDD

6.3.4.1. Produktywne techniki tworzenia scenariuszy

6.3.5. Capybara - testowanie przez warstwę prezentacji

6.3.5.1. Pułapki i najlepsze praktyki tworzenia testowalnych widoków

## 7. Deployment i automatyzacja

7.1. Deployment w chmurze

7.1.1. Heroku

7.2. Własny server

7.2.1. capistrano

7.3. Zaawansowana automatyzacja

7.3.1. Developerskie środowisko produkcyjne

7.3.2. VirtualBox, Vagrant, Chef

7.4. Continuous integration

## 8. Wskazówki praktyczne

8.1. Zasady S.O.L.I.D. w praktyce

8.2. Pułapki wydajności

8.3. Bezpieczeństwo