

# TWORZENIE APLIKACJI Z UŻYCIEM SPRING FRAMEWORK

KOD: JSPR

# Profil uczestnika

Uczestnik:

- posiada minimum roczne doświadczenie w programowaniu w języku Java;
- zna podstawy tworzenia aplikacji w Java SE / EE;
- zna podstawowe założenia platformy Java EE;
- chce poznać efektywne sposoby używania Spring Framework w tworzeniu aplikacji.

# Korzyści ze szkolenia

1. **Duża intensywność szkolenia** – w niedługim czasie uczestnik poznaje najbardziej użyteczne aspekty Spring Framework, umożliwiające efektywne tworzenie aplikacji w technologii Java EE i Java SE.
2. **Aplikacje tworzone są w uznanym standardzie open source, powszechnie używanym do tworzenia aplikacji korporacyjnych** – Spring Framework to de facto standard w technologii Java.
3. **Programista potrafi rozwiązywać problemy podczas tworzenia aplikacji** – w trakcie szkolenia uczymy, w jaki sposób radzić sobie z potencjalnymi problemami, które mogą pojawić się w trakcie tworzenia projektu. Pokażemy sprawdzone techniki i triki programistyczne.

# Parametry szkolenia

CZAS TRWANIA: 2 dni – 16 godzin

FORMA ZAJĘĆ: Ćwiczenia - 60%, wykład – 40%.

WIELKOŚĆ GRUPY: ok. 10 osób.

## Szczegółowy program

Moduły szkoleniowe	Nabyte wiedza i umiejętności, poruszane zagadnienia
<b>Wprowadzenie</b>	<ul style="list-style-type: none"><li>• Zalety i wady platformy Java EE</li><li>• Zadania postawione przed Spring Framework<ul style="list-style-type: none"><li>○ Kontrola tworzenia obiektów</li><li>○ Tworzenie aplikacji internetowych</li><li>○ Wsparcie dla trwałości danych</li><li>○ Programowanie aspektowe</li><li>○ Integracja z innymi bibliotekami</li></ul></li></ul>
<b>Spring Framework Core</b>	<ul style="list-style-type: none"><li>• Programowanie komponentowe oparte o interfejsy</li><li>• JavaBeans i POJO – nowe spojrzenie</li><li>• Spring jako fabryka</li><li>• Inversion of control</li><li>• Konfiguracja XML</li><li>• Konfiguracja z użyciem adnotacji</li><li>• Programistyczne korzystanie z fabryki</li><li>• Singletony i prototypy</li><li>• Inicjalizacja stanu początkowego obiektu</li></ul>
<b>Wstrzykiwanie zależności (Dependency Injection)</b>	<ul style="list-style-type: none"><li>• Koncepcja wstrzykiwania zależności</li><li>• Składanie obiektów</li><li>• Automatyczna konfiguracja (autowiring)</li><li>• Aliasy</li></ul>

	<ul style="list-style-type: none"><li>• Kolejność tworzenia obiektów</li><li>• Cykl życia obiektów</li><li>• Zdarzenia kontenera</li></ul>
<b>Walidacja</b>	<ul style="list-style-type: none"><li>• Walidatory</li><li>• Obiekty błędów</li><li>• ValidationUtils</li><li>• Komunikaty błędów i ich lokalizacja</li></ul>
<b>Programowanie aspektowe</b>	<ul style="list-style-type: none"><li>• Koncepcja programowania aspektowego i zastosowanie</li><li>• Adnotacja @AspectJ</li><li>• Deklarowanie aspektów</li><li>• Deklarowanie punktów przecięcia<ul style="list-style-type: none"><li>○ Wyrażenia AspectJ</li></ul></li><li>• Deklarowanie advice<ul style="list-style-type: none"><li>○ „przed” (before)</li><li>○ „po” (after)</li><li>○ „po wyrzuceniu wyjątku” (after throwing)</li><li>○ „przed i po” (around)</li></ul></li><li>• Introdukcje</li></ul>
<b>Trwałość danych</b>	<ul style="list-style-type: none"><li>• Wzorzec DAO</li><li>• Hierarchia DaoSupport</li><li>• Hierarchia DataAccessHierachy</li><li>• JDBC DAO, JDBC Template, Row Mapper</li><li>• Hibernate DAO</li><li>• Transakcje</li></ul>