

Kod szkolenia: **J/TOOLS**

Tytuł szkolenia: **Warsztat narzędziowy dobrego programisty**

Dni: 3

Opis:

Adresaci Szkolenia:

Szkolenie adresowane jest zarówno do początkujących programistów Java jak i developerów z większym doświadczeniem, którzy pragną rozbudować swój warsztat pracy poprzez zwiększenie swojej efektywności za pomocą automatyzacji wielu codziennych zadań "okołoprogramistycznych" przy zachowaniu kompleksowego podejścia do wysokiej jakości własnej pracy.

Adresatami szkolenia są również osoby decyzyjne i odpowiedzialne w swoich firmach za organizację pracy w zespołach i wyznaczanie standardów pracy w skali firmy.

Cel szkolenia:

Celem szkolenia jest zdefiniowanie i wdrożenie u siebie dobrego warsztatu pracy programisty znanego z innych zawodów cechujących się wysokim poziomem wyspecjalizowania.

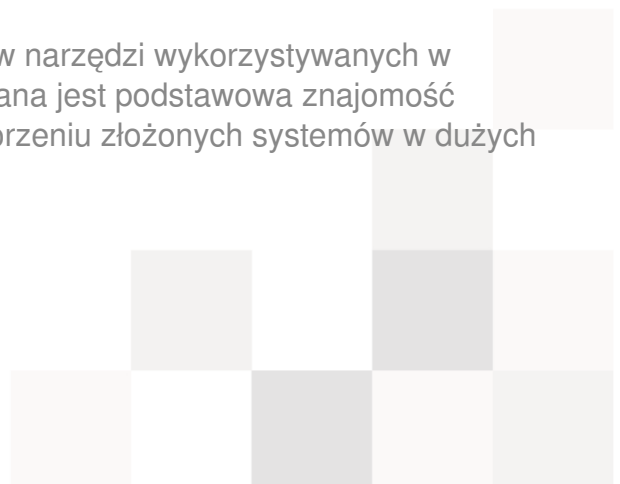
Warsztat ten wesprze u uczestnika m.in.:

- jakość pracy
- optymalizację zadań powtarzalnych poprzez ich automatyzację, np. wyzwolenie się z piekła release'owania
- produktywność i kreatywność poprzez zmniejszenie obciążenia czynnościami żmudnymi
- świadomość dobrych praktyk zarówno na poziomie kodu, systemu, pracy zespołu i procesów dostarczania.

Wymagania:

Szkolenie w sposób syntetyczny przedstawia zestaw narzędzi wykorzystywanych w codziennej pracy programisty Java, dlatego wymagana jest podstawowa znajomość programowania w tym języku. Doświadczenie w tworzeniu złożonych systemów w dużych zespołach będzie dodatkowym atutem.

Parametry szkolenia:

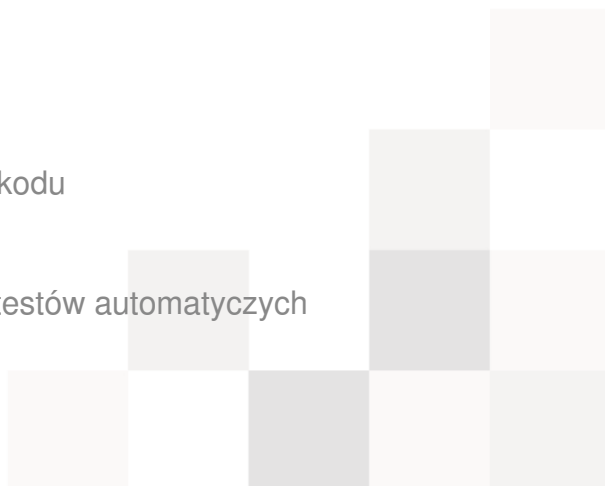


3*8 godzin (3*7 godzin netto) w proporcji: 80% warsztaty, 20% dyskusje i wykłady.

Wielkość grupy: maks. 8-10 osób.

Program szkolenia:

1. Wprowadzenie
 - I. Czy można mówić o rzemiośle programisty?
 - II. Czym charakteryzuje się dobry programista?
2. Wersjonowanie kodu na przykładzie gita
 - I. czemu nie SVN?
 - II. lokalne i zdalne repozytoria
 - III. branche'owanie i merge'owanie
 - IV. cherry-picking
3. Budowanie aplikacji za pomocą Apache Maven
 - I. zarządzanie zależnościami
 - II. parent POMy i dziedziczenie
 - III. pluginy
 - IV. profile
 - V. tworzenie repozytorium bibliotek na przykładzie Sonatype Nexus
 - i. instalacja
 - ii. konfiguracja
 - iii. integracja z Mavenem w pomie
4. Praca z IDE - Eclipse lub IntelliJ
 - I. najczęściej wykorzystywane funkcje
 - II. customizacja środowiska
 - III. użyteczne pluginy
 - IV. skróty klawiszowe
5. Logowanie zdarzeń i informacji na przykładzie log4j
 - I. konfiguracja appenderów, loggerów i layoutów
 - II. poprawne korzystanie z poziomów logowania
 - III. techniki redukcji narzutu logowania na wydajność
6. Test-Driven Development jako podejście do tworzenia oprogramowania
 - I. idea TDD
 - II. JUnit
 - III. Mockito
 - IV. Pokrycie kodu testami, np. EmmaPlugin
7. Dbanie o jakość kodu
 - I. CheckStyle
 - II. FindBugs
 - III. SonarQube
 - IV. Integracja w/w z Maven
8. Code review, czyli wspólne dbanie o jakość kodu
 - I. strategie code review
 - II. Gerrit
9. Dbanie o jakość aplikacji z wykorzystaniem testów automatycznych



- I. w zależności od potrzeb grupy: Selenium i/lub Robot Framework i/lub Soap UI
- 10. Continuous Integration i Continuous Deployment z wykorzystaniem Jenkins
 - I. czemu potrzebujemy CI i CD?
 - II. konfiguracja zadań
 - III. integracja z git
 - IV. integracja z maven
 - V. pluginy
- 11. Praca z zadaniami i błędami na przykładzie Atlassian Jira
 - I. typowe workflow
 - II. integracja z git
- 12. Zorientowanie na zadania z pomocą Mylyn
 - I. Zadania jako główna jednostka pracy
 - II. Zarządzanie zadaniami Jira
 - III. Wykonywanie zadań z jednoczesnym ich rozliczaniem i obsługą (1 komentarz, 1 narzędzie, 3 czynności)
 - IV. Integracja z Eclipse

