

Kod Szkolenia: J/TOOLS

Tytuł Szkolenia: Narzędzia wspomagające tworzenie projektów w Java

Adresaci Szkolenia:

Szkolenie adresowane jest do programistów tworzących w języku Java niezależnie od platformy JME/JSE/JEE pragnących udoskonalić swój warsztat pracy.

Cel szkolenia:

Celem szkolenia jest zapoznanie uczestników z narzędziami i metodami pracy mającymi na celu usprawnienia wykonywania czynności związanych z developmentem. Zakres szkolenia obejmuje zarówno teoretyczne podstawy inżynierii oprogramowania jak i praktyczne narzędzia wchodzące w skład warsztatu profesjonalisty. Poznane techniki mają znaczny wpływ na zwiększenie wydajności pracy o jakości produkowanego kodu.

Na życzenie modyfikujemy program szkolenia pod inne ze znanych narzędzi developerskich.

Wymagania:

Od uczestników szkolenia wymagana jest znajomość języka Java, mile widziana wiedza ogólna o Java Enterprise Edition, podstawowa znajomość środowiska Eclipse.

Parametry szkolenia:

2*7 godzin wykładów i warsztatów w proporcji 1/1. W czasie warsztatów oprócz prostych ćwiczeń będzie budowane kompleksowe środowisko developerskie.
Wielkość grupy: max 8-10 osób.

Program szkolenia:

1. Ant – podstawowe narzędzie budowania projektu
 - 1.1. Kompilacja źródeł
 - 1.2. Budowanie projektu
 - 1.3. Integracja Ant z SVN
 - 1.4. Zarządzanie zależnościami przy pomocy Apache Ivy
2. Maven jako kompleksowa platforma projektu
 - 2.1. Ant czy Maven – jak świadomie wybrać odpowiednie narzędzie dopasowane do problemu
 - 2.2. Standardowa struktura projektu Maven
 - 2.2.1. Tworzenie projektów wielomodułowych (w tym JEE)
 - 2.3. Kluczowe elementy cyklu budowania projektu
 - 2.3.1. Automatyczne testowanie (JUnit)
 - 2.3.2. Tworzenie bieżących wersji (snapshot)
 - 2.3.3. Deploy projektu do środowiska integracyjnego
 - 2.3.4. Release nowej wersji projektu
 - 2.4. Zarządzanie zależnościami
 - 2.4.1. Strategie tworzenia hierarchii POM
 - 2.5. Tworzenie korporacyjnego repozytorium bibliotek
 - 2.6. M2 – Eclipse plugin
 - 2.6.1. Dostrojenie ustawień pluginu
 - 2.6.2. Zwiększenie ergonomii i produktywności: jednoczesne korzystanie z budowania Maven i hotdeploying na server aplikacji wraz z możliwością debugowania po stronie servera
3. Praca grupowa z CVS/SVN
 - 3.1. Konfiguracja pluginów Eclipse
 - 3.2. Praca z repozytorium
 - 3.2.1. Podstawowe czynności (commit, update)
 - 3.2.2. Pliki lokalne
 - 3.2.3. Rozwiązywanie konfliktów (merging)
 - 3.3. Zarządzanie wersjami kodu
 - 3.3.1. Tags w celu oznaczenia miejsca rozwoju
 - 3.3.2. Branches w celu współbieżnego rozwoju wersji
 - 3.4. Zorientowanie na zadania z pomocą Mylyn
 - 3.4.1. Zadania jako główna jednostka pracy
 - 3.4.2. Zarządzanie zadaniami Bugzilla/Jira
 - 3.4.3. Wykonywanie zadań z jednoczesnym ich rozliczaniem i obsługą CVS/SVN (1 komentarz, 1 narzędzie, 3 czynności)
4. Standaryzacja i zwiększenie jakości kodu dzięki Checkstyle
 - 4.1. Nieodzwonne standardy kodowania
 - 4.2. Standardowe konwencje nazewnicze (JSE, JEE)
 - 4.3. Metryki jakości kodu
 - 4.4. Konfiguracja pluginu
5. Profesjonalne logowanie zdarzeń i informacji na przykładzie log4j
 - 5.1. Konfiguracja appenderów
 - 5.2. Poprawne korzystanie z poziomów logowania
 - 5.3. Techniki redukcji narzutu logowania na wydajność
 - 5.4. Chainsaw – narzędzie do przeglądania logów
6. Teoria pragmatycznego podejścia do testów z wykorzystaniem JUnit
 - 6.1. Wstęp do metodyki Test Driven Development i podejścia Design by Contract
 - 6.1.1. Tworzenie scenariuszy testowych na podstawie Use Case
 - 6.1.2. Testowanie na zasadzie „białe skrzynki”
 - 6.1.3. Testowanie na zasadzie „czarnej skrzynki”

- 6.2. Podstawowe zasady projektowe i architektoniczne zwiększające testability (podatność na testy) kodu
 - 6.2.1. Wsparcie technik OO
 - 6.2.2. Wstrzykiwanie zależności
 - 6.2.3. Mock Objects na przykładzie EasyMock
 - 6.2.4. Stubs
 - 6.2.5. Fakes
- 6.3. Typy testów i ich odpowiednie zastosowanie
 - 6.3.1. Testowanie jednostkowe
 - 6.3.2. Testowanie integracyjne
- 6.4. Kontrola pokrycia kodu testami (code coverage) z użyciem EclEmma
- 6.5. Testowanie starych systemów – sposoby na izolowanie funkcjonalności poddawanej testom
- 6.6. Podejście oparte o specyfikacje
 - 6.6.1. Podstawowe pojęcia (predykat, niezmiennik)
 - 6.6.2. Biblioteka T2
- 6.7. Testowanie GUI aplikacji webowych z wykorzystaniem JsUnit i Selenium
- 6.8. Błyskawiczne testowanie EJB 3.1 w lekkich kontenerach (GlassFish)
- 6.9. TestNG jako alternatywa dla JUnit
- 7. WTP (Eclipse Web Tools Platform) jako podstawowe narzędzie developera JEE
 - 7.1. Składowe platformy
 - 7.2. Konfiguracja pluginu
 - 7.2.1. Konfiguracja serverów
 - 7.2.2. Przygotowanie do efektywnej pracy z wykorzystaniem hotdeploy i debugowania kodu po stronie servera
 - 7.3. Praktyczne wykorzystanie komponentów platformy pod kątem warstw aplikacji korporacyjnej
 - 7.3.1. Moduł EJB
 - 7.3.1.1. Wsparcie dla JPA
 - 7.3.1.2. Wsparcie dla JMS
 - 7.3.2. Moduł Web
 - 7.3.2.1. Wsparcie dla HTML, JS, CSS
 - 7.3.2.2. Wsparcie dla JSF
 - 7.3.3. Moduł Webservices
 - 7.3.3.1. Wsparcie dla generowania usługi
 - 7.3.3.2. Wsparcie dla generowania klienta
- 8. Wdrożenie podejścia Continuous Integration w celu śledzenia stanu projektu
 - 8.1. Hudson – server CI
 - 8.2. Maven jako narzędzie budowania
 - 8.3. JUnit jako narzędzie testowania jednostkowego i integracyjnego
- 9. Praktyczne sposoby zwiększenia produktywności dzięki wykorzystaniu mniej znanych funkcji Eclipse i systemu operacyjnego
 - 9.1. Kustomizacja środowiska (perspektywy i widoki)
 - 9.2. Małe lecz cenne wtyczki
 - 9.3. Skróty klawiszowe optymalizujące czas wykonania powtarzalnych i żmudnych czynności
 - 9.4. Rozszerzenia Windows usprawniające różne aspekty pracy