

Kod Szkolenia: J/ARCH

Tytuł Szkolenia: Architektura systemów (Java i integracja)

Adresaci szkolenia:

Szkolenie adresowane jest do osób, które chciałyby zapoznać się z praktycznymi aspektami tworzenia architektury. Dla wszystkich osób, które chcą otworzyć przed sobą nowe możliwości w zakresie realizacji zadań związanych z wyższymi kompetencjami architekta. Jak również dla osób pragnących osiągnąć wyższą świadomość konsekwencji płynących z dobieranych rozwiązań, w celu podejmowania lepszych decyzji.

Szkolenie jest odpowiednie zarówno dla programistów jak i projektantów, analityków, czy już aktualnie architektów chcących usystematyzować wiedzę i wymienić doświadczenia.

Cel szkolenia:

Celem szkolenia jest zdobycie wiedzy niezbędnej do tworzenia i weryfikacji architektury oraz umiejętności rozpatrywania potencjalnych rozwiązań z punktu widzenia parametrów systemowych. Jak również poznanie języka UML, w zakresie modelowania architektury i umiejętności tworzenia modeli architektonicznych.

Szkolenie kładzie duży nacisk na osiągnięcie wysokiej świadomości konsekwencji związanych z doбором rozwiązań, technologii, wzorców i innych decyzji architektonicznych. W oparciu o tą świadomość ćwiczymy i budujemy umiejętność podejmowania i weryfikacji decyzji architektonicznych poruszając się w realiach nieklarownych wizji systemu i dużej ilości założeń architektonicznych. Rozpatrując decyzje z punktu widzenia korzyści i wad, oraz omawiając sposoby weryfikacji zarówno decyzji jak i założeń.

Szerokim tematem szkolenia są również wzorce oraz modelowanie w UML, gdzie poznanie języka UML jest tylko środkiem, a jako cel wyznaczone jest nabycie umiejętności tworzenia modeli architektonicznych bazując na wymaganiach klienta.

Wymagania:

Szkolenie wprowadza do zagadnień architektury od podstaw. Zarówno w tematyce samej architektury, wzorców, technologii, jak i UML. W związku z tym szkolenie nie posiada żadnych wymagań wstępnych stawianych uczestnikom.

Parametry szkolenia:

4*7 lub 5*7 (wersja 4 i 5 dni) wykładów i warsztatów w proporcji 1/3. Proporcje warsztatów różnią się w zależności od aktualnej tematyki (wyższe przy UML, niższe przy technologiach poszczególnych warstw aplikacji)

Wielkość grupy: maks. 8-10 osób.

Polecane szkolenia poprzedzające:

Wszystko co związane z Javą, wzorcami i UML może być traktowane jako szkolenie poprzedzające. Na architekturze dotykamy mnóstwa tematów na wysokim poziomie abstrakcji (konsekwencje dla parametrów systemowych – wady, zalety), zatem głębsza znajomość poszczególnych technologii może być pomocna, aczkolwiek nie jest wymagana.

Program szkolenia:

1. Podstawy Architektury

- Czym jest architektura
 - i. Architektura a projekt
 - ii. Cele tworzenia architektury
- Kim jest architekt i jaką pełni rolę
 - i. Kim jest architekt - różne poziomy
 - 1. Technolog
 - 2. Strateg
 - 3. Polityk
 - ii. Co robi architekt
 - iii. Potrzeba istnienia architekta a skala projektu
- Proces architektoniczny
- Dokumentacja architektoniczna
- Zarządzanie ryzykiem

2. Parametry systemowe

- Czym są parametry systemowe

- Jak poprawnie definiować wymagania niefunkcjonalne
 - Parametry systemowe
 - i. Wygoda użytkownika (Usability)
 - ii. Bezpieczeństwo (Security)
 - iii. Wydajność (Performance)
 - 1. Przepustowość (Throughput)
 - 2. Czas odpowiedzi (Response Time)
 - 3. Czas reakcji (Responsivness)
 - iv. Dostępność (Availability)
 - v. Niezawodność (Reliability)
 - vi. Skalowalność (Scalability)
 - vii. Różne wymiary elastyczności systemu
 - 1. Rozszerzalność (Extensibility)
 - 2. Reużywalność (Reusability)
 - 3. Przenaszalność (Portablity)
 - 4. Elastyczność (Flexibility)
 - viii. Realizowalność (Realizability)
 - ix. Planowalność (Planability)
 - x. Testowalność (Testability)
 - xi. Utrzymanie (Maintainability)
 - xii. Serwisowalność (Serviceability)
 - xiii. Zarządzalność (Managebility)
 - Wymiary systemu
 - i. Wymiary związane z infrastrukturą
 - 1. Pojemność (Capacity)
 - 2. Redundantność/Replikacja (Redundancy)
 - 3. Modułowość (Modularity)
 - ii. Wpływ wymiarów na parametry systemu
 - iii. Inne wymiary systemu
 - 1. Tolerancja (Tolerance)
 - 2. Obciążenie (Workload)
 - 3. Niejednorodność/Jednorodność (Homo/Heterogenity)
 - Priorytety parametrów systemu
 - i. Skąd wynikają priorytety?
 - ii. Problemy priorytetowania
- ### 3. Wzorce architektoniczne

- Wprowadzenie do wzorców
 - i. Definicja wzorca
 - ii. Cechy i zalety wzorców
 - iii. Rodzaje wzorców
- Wzorce architektoniczne
 - i. Problemy architektury komponentowej
 - ii. Wzorce podziału odpowiedzialności
 - 1. MVC (Model View Controll)
 - 2. Web-centric
 - 3. Application-centric
 - 4. Enterprise
 - a. Wymagania systemów Enterprise
 - b. Architektura Enterprise w JEE
 - 5. Architektura wielowarstwowa (Layers Pattern)
 - a. Architektura wielowarstwowa w JEE
 - iii. Wzorce EAI (Enterprise Application Integration)
 - 1. SOA (Service Oriented Architecture)
 - 2. ESB (Enterprise Service Bus; Szyna Danych; Broker Integracyjny)
 - 3. MOM (Message Oriented Middleware)
 - iv. Wzorce infrastruktury
 - 1. Redundancja Ścieżek
 - 2. Skalowanie pionowe
 - 3. Skalowanie poziome (Replikacja)
 - 4. Równoważenie obciążenia (Load Balancing)
 - 5. Klastry (Clustering)
 - a. HA (High Availability)
 - b. Failover
 - 6. Forward Proxy Cache
 - v. Wzorce blokowania zasobów
 - 1. Blokowanie optymistyczne (Optimistic Lock)
 - 2. Blokowanie pesymistyczne (Pessimistic Lock)
 - a. Blokada domyślna (Implicit Lock)
 - b. Blokowanie gruboziarniste (Coarse-Grained Lock)
 - vi. Inne wzorce architektoniczne
 - 1. Dependency Injection
 - 2. Dependency Inversion

3. Stable Dependency Principle
 4. Szablon wzorców POSA
 5. Szablon wzorców PEAA
 6. Szablon wzorców Core J2ee
 7. Szablon wzorców EAI (wzorce JMS)
4. Prototypowanie
 - Po co prototypować
 - Prototyp Proof of Concept
 - Prototyp ewolucyjny
 - Antywzorzec Lava Flow
 5. Metodyki wytwarzania oprogramowania a architektura
 - Metodyka kaskadowa
 - USDP (UP) – Unified Software Development Process
 - i. Założenia
 - ii. Wymiary
 - iii. Fazy
 1. Rozpoczęcie (Inception)
 2. Opracowanie (Elaboration)
 3. Budowa (Construction)
 4. Wdrożenie (Transition)
 - RUP – Rational Unified Process
 - SynTone Architecture Methodology
 - Metodyki Agile
 - i. Extreme Programming (XP)
 - ii. Scrum
 - Podejście hybrydowe
 6. Architektura warstwy klienta i prezentacji
 - Podział klientów
 - i. Klient gruby
 - ii. Klient Cienki
 1. RIA
 - Przechowywanie sesji
 - Technologie klienta grubego
 - i. Swing
 - ii. SWT
 - iii. RCP

- Klient gruby zanurzony w kliencie cienkim
 - i. Applet
 - ii. Java Web Start
 - iii. Java FX
- Technologie klienta cienkiego
 - i. HTML Statyczny
 - ii. HTML Dynamiczny
 1. Servlety + JSP + JSTL
 2. Portlety
 3. JSF (Java Server Faces)
 4. Ajax
 - a. Java Script
 - b. Prototype
 - c. Ajax4JSF
 - d. RichFaces
 - e. GWT
 5. Wsparcie JavaScript
 - a. JSON
 - b. jQuery
- 7. Architektura warstwy biznesowej
 - Przetwarzanie rozproszone
 - Komunikacja zdalna a lokalna
 - Optymalizacja komunikacji sieciowej
 - Protokoły komunikacyjne
 - i. CORA i IIOP
 - ii. Web Services
 1. SOAP i REST
 2. WSDL
 3. Repozytoria usług: UDDI, ebXML
 4. Specyfikacje JEE
 - a. JAX-P (Java API for XML Processing)
 - b. SAAJ (SOAP with Attachments API for Java)
 - c. JAX-B (Java API for XML Binding)
 - d. JAX-R (Java API for XML Registries)
 - e. JAX-WS (Java API for XML Web Services)
 5. Web Service Orchestration (wsparcie procesów biznesowych)

- iii. Sockets (własny protokół)
 - iv. RMI (Remote Method Invocation)
 - v. EJB i RMI-IIOP
 1. Serwer aplikacji
 2. Usługi serwera aplikacji
 3. Kryteria wyboru serwera
 - Rodzaje komponentów EJB
 - i. Sesyjne
 1. Statefull
 2. Stateless
 3. Singleton (od EJB3.1 – JEE6)
 - ii. MDB
 - iii. Encyjne (do EJB2.x)
 - Porównanie implementacji i komunikacji EJB 2.x a EJB 3.x
 - JNDI
8. Architektura warstwy integracji i zasobów
- Technologie utrwalania danych
 - i. Bazy relacyjne
 1. JDBC
 2. Entity EJB
 3. JDO (także bazy obiektowe i inne)
 4. JPA
 - ii. LDAP (bazy hierarchiczne)
 - iii. JCR – Java Content Repository (systemy CMS)
 - iv. JCA – Java Connector Architecture (systemy EIS)
 - v. Wzorzec DAO
 - vi. Wzorzec Domain Store
 - Komunikacja asynchroniczna
 - i. JMS (Java Message Service)
 1. Topic
 2. Queue
 - ii. MDB EJB (Message Driven Bean)
 - Systemy „Legacy”
 - Screen Scrapping
9. Wzorce projektowe a architektura (**tylko wersja 5cio dniowa**)
- Jak wzorce projektowe mogą wpływać na architekturę

- Wybrane wzorce Core J2EE
 - i. Warstwa Prezentacji
 - 1. Intercepting Filter (flexibility)
 - 2. Context Object (maintenance, flexibility)
 - 3. Composite View (flexibility)
 - 4. Service To Worker (flexibility)
 - ii. Warstwa Biznesowa
 - 1. Business Delegate (maintenance)
 - 2. Service Locator (maintenance)
 - 3. Session Façade (performance, flexibility)
 - 4. Transfer Object (performance)
 - 5. Value List Handler (performance, scalability)
 - 6. Application Service (flexibility, maintenance)
 - 7. Business Object (flexibility, maintenance)
 - iii. Warstwa Integracji
 - 1. DAO (flexibility)
 - 2. Domain Store (flexibility)
- Wybrane wzorce GOF
 - i. Factory Method (flexibility)
 - ii. Abstract Factory (reliability, flexibility)
 - iii. Builder (reliability, flexibility)
 - iv. Prototype (performance)
 - v. Singleton (performance)
 - vi. Façade (performance, flexibility)
 - vii. Command (flexibility)
 - viii. Strategy (flexibility)
 - ix. Adapter (flexibility)
 - x. Mediator (flexibility)
 - xi. Observer (performance, flexibility)
 - xii. Template Method (reliability, flexibility)
 - xiii. Bridge (reliability, flexibility)
 - xiv. Memento (reliability)
 - xv. Visitor (flexibility)
 - xvi. Flyweight (memory performance)
 - xvii. Chain of responsibility (flexibility)
 - xviii. Proxy (flexibility, performance)

xix. Decorator (flexibility)

10. Wprowadzenie do UML

- Czym jest modelowanie
- Czym jest a czym nie jest UML
- Rozwój UML
- Podstawowe elementy UML
 - i. Podstawowe kwalifikatory
 - 1. Klasa (Class)
 - 2. Interfejs (Interface)
 - 3. Obiekt (Object)
 - 4. Aktor (Actor)
 - 5. Przypadek Użycia (Use Case)
 - 6. Komponent (Component)
 - 7. Węzeł (Node)
 - ii. Relacje (Relationships)
 - 1. Asocjacja (Association)
 - 2. Asocjacja (Association)
 - 3. Zależność (Dependency)
 - 4. Realizacja (Realization)
 - iii. Diagramy (Diagrams)
 - iv. Komentarze (Note)
 - v. Mechanizmy rozszerzenia
 - 1. Stereotypy (Stereotype)
 - 2. Etykiety (Tagged Values)
 - 3. Ograniczenia (Constraints)
- Diagram a model UML
- Zastosowania UML

11. Modelowanie architektury w UML

- Diagram komponentów (component diagram)
 - i. Komponent (component)
 - ii. Komponenty zagnieżdżone
 - iii. Interfejs (interface)
 - 1. Interfejs wymagany (required interface)
 - 2. Interfejs dostarczany (provided interface)
 - iv. Złączenie (assembly)
- Diagram wdrożenia (deployment diagram)

- i. Węzeł (node)
- ii. Łącze (communication path)
 1. Łącze kierunkowe
 2. Liczność łącza

12. Zaawansowane aspekty modelowania architektury w UML

- Zaawansowane elementy diagramu komponentów (component diagram)
 - i. Porty
 - ii. Konektory
 - iii. Realizacja komponentu
- Zaawansowane elementy diagramu wdrożenia (deployment diagram)
 - i. Instancyjne diagramy wdrożenia
 - ii. Niskopoziomowe diagramy wdrożenia
 - iii. Szablony architektoniczne
 - iv. Model wdrożenia na diagramach wdrożenia
 1. Po do model wdrożenia
 2. Artefakt
 3. Stereotypy artefaktów
 - a. <<file>>
 - b. <<document>>
 - c. <<library>>
 - d. <<executable>>
 - e. <<script>>
 - f. <<source>>
 4. Specyfikacja konfiguracji (deployment specification)
 5. Relacje między artefaktami
 - a. Kompozycji (composition)
 - b. Zależności (dependency)
 6. Instalacja artefaktów (deployment) <<deploy>>
 7. Manifestacja (manifestation) <<manifest>>
 - v. Diagram pakietów (package diagram)
 1. Pakiet
 2. Zagnieżdżanie (nest, nesting)
 3. Przestrzeń nazw
 4. Importowanie (package import)
 - a. <<import>>
 - b. <<access>>

5. Łączenie (merge)
6. Diagramy pakietów i modelowanie warstw architektury
 - a. Przekięcie warstw i poziomów

13. Przejście z architektury do projektu

- Warstwy i komponenty a realizacja projektu
- Warstwy i komponenty a model projektowy
- Uwzględnienie ograniczeń architektury w projekcie
 - i. Na modelu statycznym
 - ii. Na modelu dynamicznym

14. Bezpieczeństwo

- Mechanizmy bezpieczeństwa
 - i. Uwierzytelnianie (Authentication)
 - ii. Autoryzacja (Authorization)
 - iii. Kontrola dostępu (Access Control)
 - iv. Logowanie
 - v. Audyt
 - vi. Szyfrowanie danych
 - vii. Szyfrowanie transmisji
 - viii. Integralność i przywracanie danych (backup)
 - ix. Certyfikaty
- Bezpieczeństwo w Javie
 - i. JAAS
 - ii. Servlety
 - iii. Spring ACEGI
 - iv. EJB
- Serwery SSO (Single Sign On)
- Zarządzanie bezpieczeństwem
- Podstawowe rodzaje ataków
 - i. DOS i DDOS
 - ii. SQL Injection
 - iii. Cross Site Scripting (XSS)
 - iv. Cross Site Request Forgery

15. Transakcje

- ACID
- Poziomy izolacji
 - i. SERIALIZABLE

- ii. REPEATABLE_READ
- iii. READ_COMMITTED
- iv. READ_UNCOMMITTED
- Efekty uboczne obniżania poziomu izolacji
 - i. Fantomy (Phantoms)
 - ii. Niepowtarzalny odczyt (Unrepeatable read)
 - iii. Brudny odczyt (Dirty read)
- Wpływ transakcji na system
- Transakcje rozproszone (JTA, 2PC)
- Transakcje kompensacyjne
 - i. Na czym polega kompensacja
 - ii. 3PC
- Kontrola obciążenia systemu transakcjami
 - i. Zasięg i rozmiar transakcji
 - ii. Transakcje biznesowe a systemowe
 - iii. Unikanie transakcji
 - 1. Kompensacja
 - 2. Memento
 - 3. Rezygnacja z transakcji biznesowych
 - a. Konflikty zapisu danych
 - i. Blokowanie optymistyczne
 - ii. Blokowanie pesymistyczne
 - 4. Buforowanie danych
- Transakcje a EJB
 - i. Container Managed Transaction
 - 1. Required
 - 2. Requires new
 - 3. Mandatory
 - 4. Supports
 - 5. Not supported
 - 6. Never
 - ii. Bean Managed Transaction
 - iii. Client Managed Transaction
- Transakcje długoterminowe w Webservice

16. Weryfikacja i ocena architektury

- Po co weryfikować?

sages

- Zespół weryfikujący
- Techniki weryfikacji i oceny
- Proces weryfikacji
- Raport z weryfikacji

