Metodyka tworzenia portali biznesowych

WYKŁAD 13

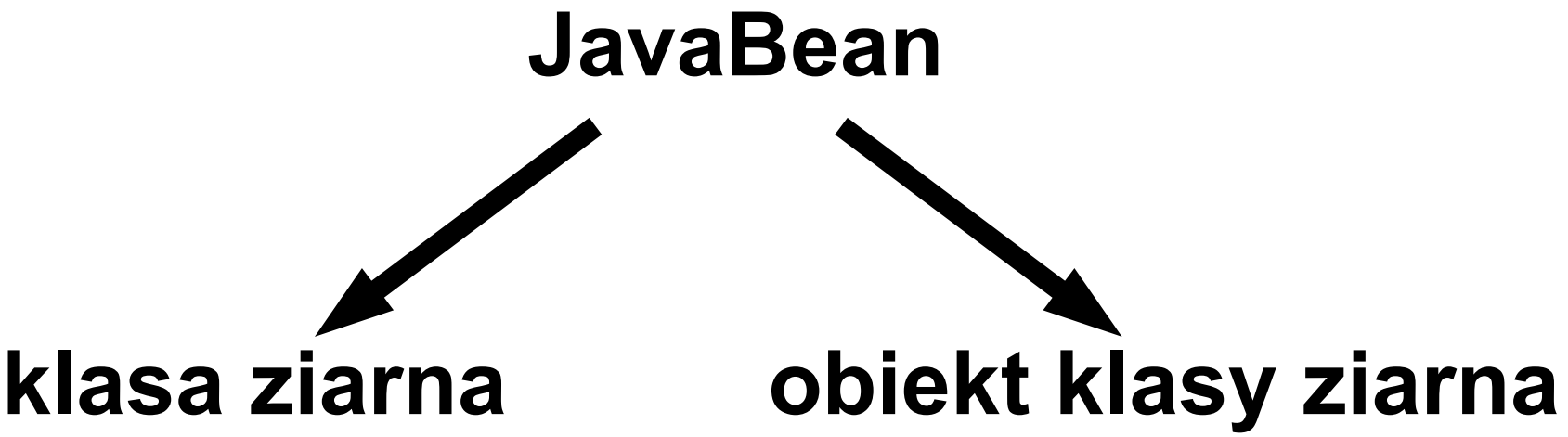
Programowanie komponentowe

- **Komponent** - moduł programowy:
  - odizolowany od środowiska i innych komponentów,
  - wystarczająco samodzielny,
  - zaopatrzony w odpowiednią specyfikację, określającą jego wymagania i możliwości,
  - udostępniający swoją funkcjonalność za pomocą jednoznacznie zdefiniowanego interfejsu,
  - zdolny do współdziałania z pozostałymi elementami systemu lub z innymi komponentami.

JavaBean

- *Bean* – ziarno
- JavaBean – programowy komponent "wielokrotnego użytku", którego właściwości i funkcjonalność mogą być odczytywane i/lub zmieniane uniwersalnymi środkami programistycznymi.
- Klasy-ziarna są typowymi klasami języka Java, które stosują odpowiedni interfejs programistyczny.

JavaBean (cd.)

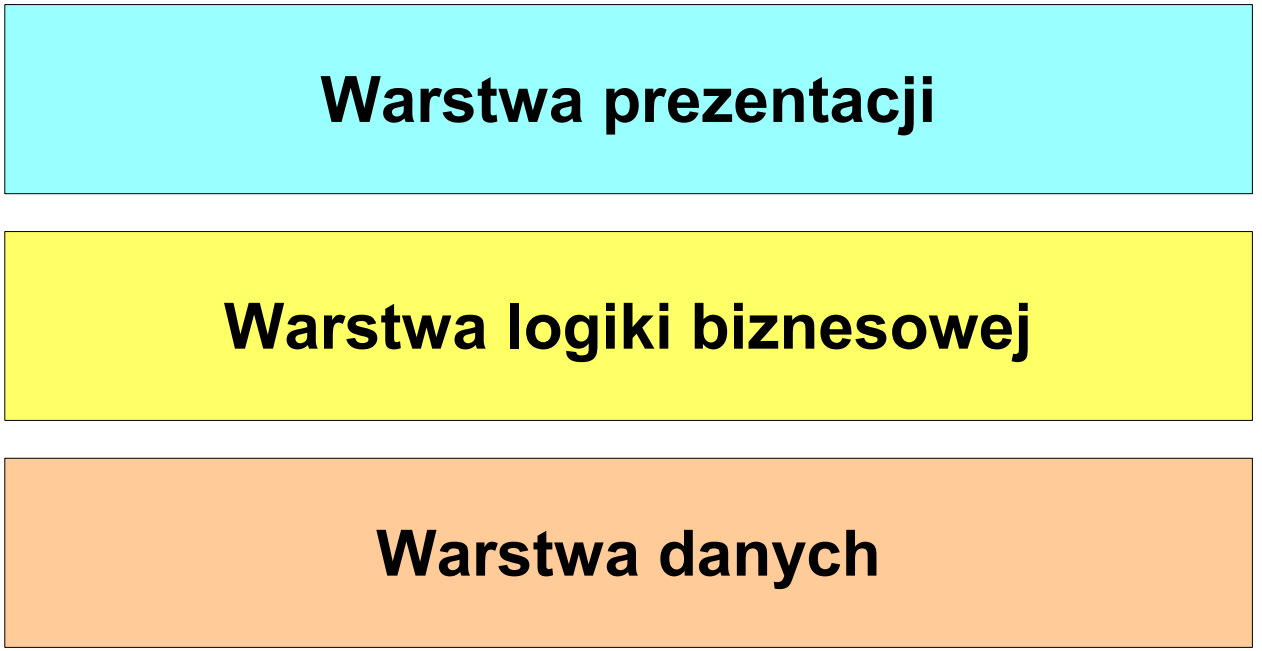


Rozróżnienie wynika z kontekstu.

Właściwości i akcesory

- Ziarna posiadają właściwości (atrybuty).
- Dostęp do właściwości zapewniają metody klasy-ziarna nazywane akcesorsami.
- *getter* – akcesor pobierający właściwość
- *setter* – akcesor ustawiający właściwość
- Rodzaje właściwości:
  - proste (właściwości posiadające jedną wartość)
  - indeksowane (właściwości posiadające wiele wartości umieszczonych w tablicy)

Aplikacje trójwarstwowe



Aplikacje trójwarstwowe

- Warstwa prezentacji
  - odpowiada za interakcję z użytkownikiem
  - może być zbudowana zgodnie ze wzorcem projektowym MVC (Model-View-Controller)

Aplikacje trójwarstwowe

- Warstwa logiki biznesowej
  - odpowiada za całość logiki biznesowej aplikacji
  - może komunikować się z innymi systemami
  - jest istotna z punktu widzenia wydajności aplikacji

Aplikacje trójwarstwowe

- Warstwa danych
  - przechowuje i udostępnia dane
  - najczęściej oparta jest o bazę danych

Model MVC

- MVC (Model-View-Controller) – wzorzec projektowy zakładający podział aplikacji na trzy współpracujące ze sobą grupy komponentów:
  - komponenty modelu,
  - komponenty widoku,
  - komponenty kontrolera.

Model MVC

- Komponenty modelu
  - implementują całość logiki biznesowej
- Komponenty widoku
  - implementują interfejs użytkownika
- Komponenty kontrolera
  - implementują kontrolę przepływu sterowania

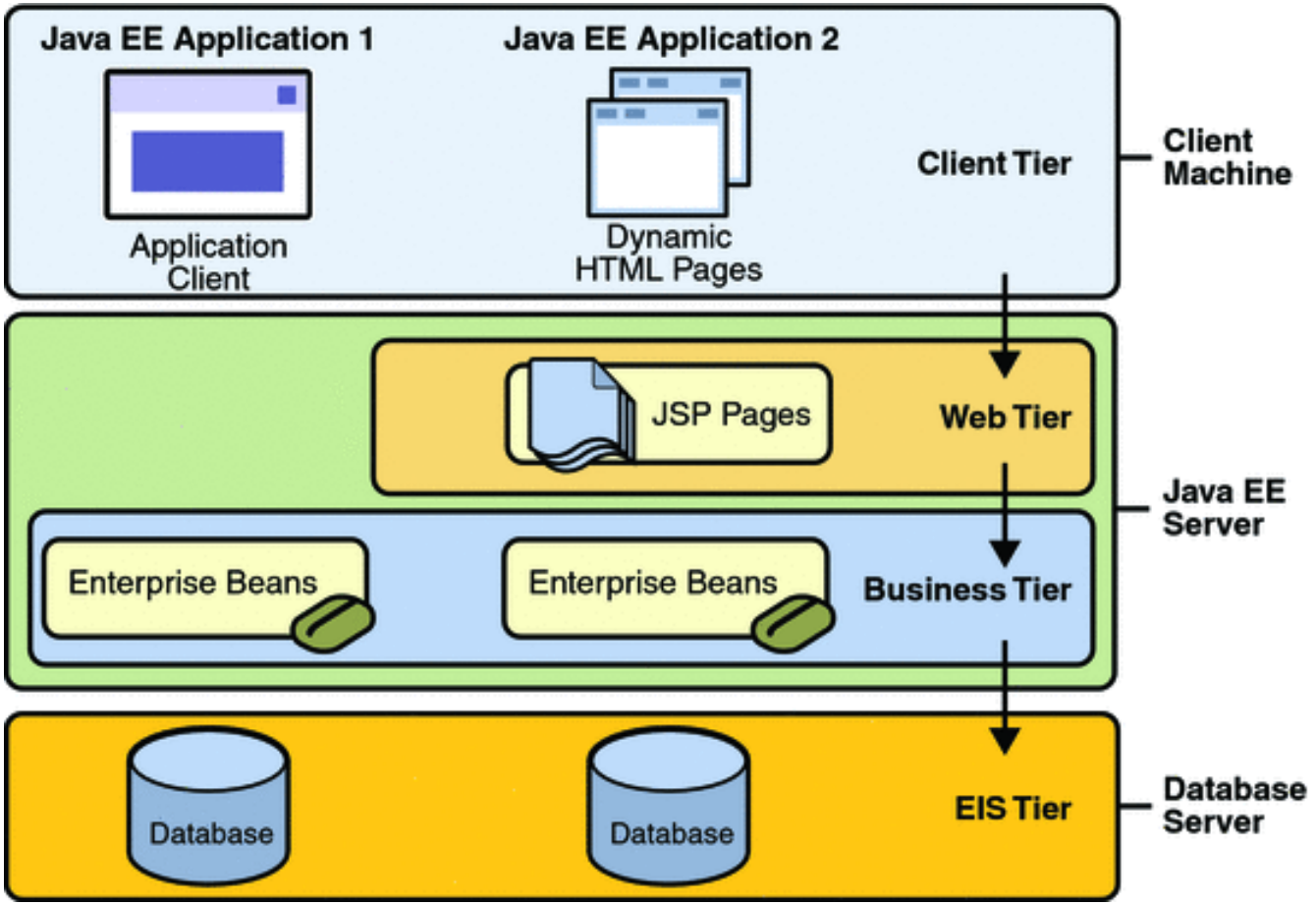
Platforma Java Enterprise Edition (Java EE)

- Platforma Java Enterprise Edition (Java EE), to platforma umożliwiająca tworzenie aplikacji webowych i korporacyjnych.
- Aplikacje te są zazwyczaj wielowarstwowe.
- Aktualna wersja: Java EE 8.

Warstwy technologii J2EE

- Warstwa kliencka (przeglądarka, aplikacja napisana w języku Java).
- Warstwa webowa (serwlety i strony JSP).
- Warstwa biznesowa (komponenty Enterprise JavaBeans).
- Warstw danych (np. bazy danych).

Warstwy technologii J2EE



Źródło: https://docs.oracle.com/javaee/5/tutorial/doc/bnaay.html

Serwery Java EE

- Serwery kompatybilne z Java EE 8
  - GlassFish
  - IBM WebSphere Application Server
  - WildFly (*dawniej JBoss AS*)
  - Tomcat (*częściowo kompatybilny*)

Enterprise JavaBeans

- Komponenty Enterprise JavaBeans (EJB) umożliwiają tworzenie aplikacji rozproszonych na bazie komponentów oferujących:
  - skalowalność,
  - przetwarzanie transakcyjne,
  - bezpieczeństwo.

Enterprise JavaBeans

- Rodzaje komponentów EJB:
  - komponenty sesyjne (*Session Beans*),
  - komponenty sterowane komunikatami (*Message-diven Beans*),
  - komponenty encyjne (*Entity Beans*) – zastąpione przez Java Persistence API.

Enterprise JavaBeans

- Komponenty EJB działają w kontenerze EJB, który zajmuje się ich tworzeniem, usuwaniem i przypisywaniem do żądań klientów.

Komponenty sesyjne EJB

- Komponenty sesyjne EJB:
  - mają na celu obsługiwanie żądań klientów,
  - działają w ten sposób, że w danym momencie co najwyżej jeden klient jest przypisany do danego egzemplarza komponentu,
  - są dostępne przez protokół IIOP, przez co mogą być wykorzystywane przez klientów implementujących mechanizm CORBA.

Komponenty sesyjne EJB

- Komponenty sesyjne stanowe:
  - wykorzystywane są do konwersacji z określonym klientem,
  - stan zapisywany jest w polach instancji komponentu stanowego, obiektów powiązanych oraz obiektów osiągalnych z komponentu,
  - oznaczane są adnotacją **@Stateful**

Komponenty sesyjne EJB

- Komponenty sesyjne bezstanowe:
  - nie zawierają żadnych informacji o konwersacji z określonym klientem,
  - aby być dostępnymi zdalnie muszą implementować interfejsy biznesowe oznaczone adnotacją @Remote
  - kontener wybiera instancję komponentu bezstanowego, która obsłuży wywołanie metody,
  - oznaczane są adnotacją **@Stateless**

Komponenty EJB sterowane komunikatami

- Komponenty EJB sterowane komunikatami:
  - obsługują zgłaszane komunikaty pozwalające klientom na jednostronną komunikację z serwerem,
  - działają wg następującego mechanizmu:
    - klient wysyła komunikat i kontynuuje przetwarzanie, nie czekając na odpowiedź,
    - kontener po stronie serwera przekazuje komunikat do komponentów oczekujących na wiadomości.

Szkielety Java EE

- Apache Struts
  - szkielet tworzenia aplikacji dla platformy Java EE z wykorzystaniem serwletów i JSP
- Spring Framework
  - szkielet tworzenia aplikacji dla platformy Java EE
  - powstał jako alternatywa dla programowania aplikacji z użyciem Enterprise JavaBeans
  - wspiera tworzenie warstwy prezentacji oraz warstwy biznesowej

Mapowanie obiektowo-relacyjne

- **Mapowanie obiektowo-relacyjne** (*ang. Object-Relational Mapping ORM*) - sposób odwzorowania obiektowej architektury aplikacji na bazę danych o relacyjnym charakterze.
- Przykład:
  - JPA (JavaPersistence API) - standard ORM dla języka Java.

Hibernate

- **Hibernate** - narzędzie ORM dla języka Java.
- Hibernate wykorzystuje do opisu struktury danych język XML.

Hibernate

- Przykład pliku konfiguracyjnego:

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
 <hibernate-configuration>
     <session-factory>
        <property name="connection.url">jdbc:mysql://localhost/kredyt</property>
        <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="connection.username">root</property>
        <property name="connection.password">qwerty</property>
         <mapping resource="Kredyt.hbm.xml"/>
    </session-factory>
 </hibernate-configuration>
```

Hibernate

- Przykład pliku mapowania:

```
<hibernate-mapping>
   <class name="Kredyt" table="kredyt1">
      <id name="id" column="ID"/>
      <property name="oszczednosci" column="Oszczednosci"/>
      <property name="majatek" column="Majatek"/>
      <property name="dochod" column="Dochod"/>
      <property name="ryzyko" column="Ryzyko"/>
   </class>
</hibernate-mapping>
```