

# Java Persistence API

The background of the slide features a light blue gradient. Overlaid on this are several thick, white diagonal lines that run from the bottom-left towards the top-right, creating a modern, geometric design.

# Agenda

---

- ORM – wstęp
- Ogólne informacje o JPA
- Mapowanie encji
- Mapowanie asocjacji
- Przegląd API
- Podsumowanie i pytania

# ORM – O co chodzi?

---

- Podejście pierwotne – JDBC
- Rozwiązania – OODBMS, ORM
- Zalety
  - Programujemy i projektujemy w pełni obiektowo
  - Unikamy niskopoziomowego kodu SQL
  - Jesteśmy niezależni od bazy danych
  - Optymalizacja
  - I tak musielibyśmy to napisać

# Cechy JPA

---

- JPA to specyfikacja
  - Implementacje: Hibernate (JBoss/RedHat), TopLink (Oracle), ...
- Przenośność
- Działa w JSE
- Klasy modelu biznesowego nie zależą od API
- Zapis mapowania w postaci XML lub adnotacji

# Proste mapowanie encji

---

```
import javax.persistence.*;

@Entity
public class Address implements Serializable {
    @Id @GeneratedValue
    private Long id;
    private String streetAddress;
    private String postalCode;
    private String city;
    private String phone;

    // get/set methods
}
```

---

```
EntityManager em = ...;
Address addr = new Address();
addr.setStreetAddress(...);
em.persist(addr);
```

# Mapowanie od drugiej strony

---

```
...
@Entity @Table(name="ADDRESSES")
public class Address implements Serializable {
    @Id @GeneratedValue
    private Long id;
    @Column(name="STREET", length=150, nullable=false)
    private String streetAddress;
    private String postalCode;
    private String city;
    @Column(unique=true, nullable=false)
    private String phone;
    @Transient
    private String temporary;
    ...
}
```

Użycie: identycznie!

# Mapowanie asocjacji

---

- Referencje na inne obiekty
  - Liczności
  - Kaskada
  - Leniwe pobieranie wartości
- Dziedziczenie



# EntityManager – tworzenie

---

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="1.0" ...>
  <persistence-unit name="sok">
    <provider>org.hibernate.ejb.HibernatePersistence</provider>
    <jta-data-source>java:/sokDS</jta-data-source>
    <properties>
      <property name="hibernate.dialect"
        value="org.hibernate.dialect.MySQL5Dialect"/>
      <property name="hibernate.hbm2ddl.auto"
        value="create-drop"/>
      <property name="jboss.entity.manager.factory.jndi.name"
        value="java:/sokEMF"/>
    </properties>
  </persistence-unit>
</persistence>
```



# EntityManager – dostę

---

- EJB way

- `EntityManager entityManager = (EntityManager) context.lookup("java:comp/env/persistence/em");`
- `@PersistenceContext`  
`private EntityManager entityManager;`

- Seam way

- `<persistence:managed-persistence-context name="entityManager" auto-create="true" persistence-unit-jndi-name= "java:/sokEMF"/>`
- `@In private EntityManager entityManager;`

# Interfejs EntityManager

---

- `<T> T find(Class<T> entityClass, Object primaryKey)`  
`em.find(Person.class, 20L)`
- `<T> T getReference(Class<T> entityClass, Object primaryKey)`
- `void persist(Object entity)`
- `<T> T merge(T entity)`
- `void refresh(Object entity)`
- `Query createNamedQuery(String name)`
- ...

# EJB Query Language

---

- Obiektowy SQL
- `SELECT p FROM Person p WHERE p.email=:email`
- Zapytania nazwane
  - `@NamedQuery(  
 name = "Paper.findByReviewer",  
 query = "SELECT p FROM Paper p, IN (p.reviews) r WHERE  
 r.reviewer=:reviewer" )  
@Entity  
public class Paper implements Serializable {...}`
  - `entityManager.createNamedQuery("Paper.findByReviewer").set  
Parameter("reviewer", pc).getResultList();`

# Wydajność

---

- Pobieranie leniwe kontra zachłanne
- Moment zapisu do bazy – `em.flush()`
- Second level cache (Hibernate)  
`<property name="hibernate.cache.use_second_level_cache" value="true"/>`  
`<property name="hibernate.cache.provider_class" value="org.hibernate.cache.EhCacheProvider"/>`


`@Entity`

`@Cache(usage=CacheConcurrencyStrategy.READ_ONLY)`

`public class PredefinedUniversity implements Serializable  
{...}`

# Podsumowanie

---

- Łatwy i przejrzysty dostęp do RDBMS
- Łatwa integracja z pozostałą częścią JEE
- Zaawansowane możliwości mapowania ORM
- Wysoka wydajność implementacji
- Kiedy JPA nie wystarcza:  HIBERNATE
  - `Session hibernateSession = (Session) em.getDelegate();`
- Literatura: C. Bauer, G. King, *Java Persistence with Hibernate*. Manning, 2006.

