|  |  |
| --- | --- |
| **===HIBERNATE===** | |
| |  |  |  |  | | --- | --- | --- | --- | |  | Teoria |  | Adnotacja | |  | @Adnotacja | pole XML |  | Klasa | |  | link |  | Nazwa klasy | |  |  |  | Akcesor | |  |  |  | Interface | |  |  |  | String | | |
| **JPA** | * To standard typu ORM (Object-RelationalMapping). * ORM oznacza mapowanie z modelu obiektowego (tu dostarczonego przez Java)  @Entity class Employee{  @Id  int id;  String name;  int salary; } |
| **Hibarnate** | * Zbiór implementacji standardu JPA |
| **@Entity** | * Adnotacja oznaczająca że klasa jest encją i posiada tablicę w bazie danych * Należy umieścić adnotację nad nazwą klasy * Klasa musi mieć publiczne gettery i settery   @Entity class Cat{  @Id  private Long id;  private String name;  public getId(){  return this.id;  }  public setId(Long id){  this.id = id;  }  public getName(){  return this.name;  }  public setName(String name){  this.name = name;  }  }   * Tablica ma taką samą nazwę jak klasa * Kolumny tablicy mają takie same nazwy jak pola |
| **@Id** | * Oznacza że pole jest unikatowym identyfikatorem w tablicy w bazie danych * Najczęściej są to typy Long dla baz relacyjnych i String dla baz NoSql |
| **@GeneratedValue** | Wskazuje że pole oznaczone adnotacją @Id jest generowane automatycznie w momenncie zapisu do bazy danych. |
| **@Table** | * Służy do zniamy nazwy tablicy * Parametr (name=”...”) jest używany do określenia nowej nazwy   @Entity  @Table(name=”cats”) class Cat{  @Id  private Long id;  private String name;  /\* getters & setter \*/ } |
| **@Column** | * Domyślnie każda kolumna ma taką samą nazwę jak pole klasy * Do zmiany tej nazwy w bazie danych służy adnotacja @Column, której użycie spowoduje mapowanie kolumny o nazwie innej niż podana w polu   @Entity  @Table(name=”cats”) class Cat{  @Id  private Long id;  @Column(name=”cat\_name”, nullable=false)  private String name;  /\* getters & setter \*/ }  // powyższy przykład posiada parametr nullable, który dla wartości false określa że pole poniżej nie może mieć wartości null |
| [org.hibernate.annotations.Entity](http://docs.jboss.org/hibernate/orm/3.5/api/index.html?org/hibernate/annotations/Entity.html) | * Dziedziczy po javax.persistence.Entity oraz ją rozszerza * Jeśli nie jest potrzebna zaawansowana logika implementacja javax.persistence |
| EntityManager | * Standardowy sposób wykonywania operacji w standardzie JPA * To “most” między bazą danych a aplikacją |
| **Session** | * Spełnia częściowo funkcję ale nie zapewnia standardu JPA * Ma nieco większe możliwości niż EntityManager |
| **Konfiguracja EntityManager w XML** | |
| **<tx:annotation-driven />**  **<bean**  **id="entityManagerFactory"**  **class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">**  **<property**  **name="dataSource"**  **ref="dataSource" />**  **<property**  **name="packagesToScan"**  **value="pl.project.model" />**  **<property**  **name="jpaVendorAdapter">**  **<bean**  **class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter">**  **<property**  **name="showSql"**  **value="false" />**  **<property**  **name="databasePlatform"**  **value="org.hibernate.dialect.PostgreSQLDialect" />**  **</bean>**  **</property>**  **<property name="jpaProperties">**  **<props>**  **<prop key="hibernate.hbm2ddl.auto">update</prop>**  **</props>**  **</property>**  **</bean>**  **<bean**  **class="org.springframework.orm.jpa.support.PersistenceAnnotationBeanPostProcessor" />**  **<bean**  **class="org.springframework.dao.annotation.PersistenceExceptionTranslationPostProcessor" />**  **<bean**  **id="transactionManager"**  **class="org.springframework.orm.jpa.JpaTransactionManager"> <property**  **name="entityManagerFactory"**  **ref="entityManagerFactory" />**  **</bean>**   * Pierwszy bean zapewnia zachowanie EntityManager * Pozostałe są wsierające:  - zwracają sensowne wyjątki - zarządzają tworzeniem i zamykaniem transakcji   Jest to implementacja wzorca fabryki | |
| **dataSource** | * Odniesienie do beana definiujące bazę danych |
| **packageToScan** | * Ścieżka do paczki zawierającego encje |
| **jpaVendorAdapter** | * Bean * Można go zdefiniować za pomocą atrybutu ref * Dodatkowe atrybuty: showSql - steruje wyświetlaniem zapytań w logach. databasePlatform - wskazuje nazwę klasy wskazującej rodzaj bazy danych (npL MySQL, PostgreSQL, Oracle). |
| **jpaProperties** | * Pozwala na zdefiniowanie wszystkich niestandardowych ustawień hibernate.hbm2ddl. - określa jak ma się zachowywać hibernate przy uruchomieniu, możliwe opcje:  |  |  | | --- | --- | | auto | Automatyczne | | validate | * Tylko weryfikuje * Powinno być wstawieniem dla działającej aplikacji * W ramach niespójności za nie pozwala uruchomić aplikacji | | update | * Jeśli Hibernate napotka niespójność próbuje ją usunąć * Modyfikuje schemat * Naraża program na utratę danych * Nadaje się w etapie testowania i rozwoju | | create | * Tworzy schemat i strukturę | | create-drop | * Tworzy schemat i strukturę * Usuwa istniejące strukture * Przy zamknięciu aplikacji usuwa zawartość bazy danych | |
| **@PersistanceContext** | * Zapewnia korzystanie z EntityManagera   class CatDao{  @PersistanceContext  private EntityManager entityManager;  @Transactional  Public Cat findCat(){  /\*  cat persistance logic based on entityManager  \*/  }  }   * Adnotacja @PersistanceContext jest wszystkim co potrzebne aby zainicjalizować EntityManager * Pozwala określić Hibernate’owi zachowanie na początku i na koniec transakcji |
| **EntityManager** | [dokumentacją EntityManagera i jego metod](https://docs.oracle.com/javaee/6/api/javax/persistence/EntityManager.html" \t "/home/emil/Documents\\x/_blank)  Najpopularniejsze metody:   |  |  | | --- | --- | | find | * Argumenty:  1. poszukiwany typ  2. identyfikator * Zwraca obiekt zgodny z typem podanym w 1. argumencie   entityManager.find(Cat.class, 1L); | | merge | * Używana do wstawiania obiektu do bazy danych oraz do aktualizowania stanu obiektu w bazie   Cat cat = new Cat();  entityManager.merge(cat); | |
|  |  |
| **Link z materiałami** | [EntityManager vs Session](https://www.theserverside.com/news/2240186700/The-JPA-20-EntityManager-vs-the-Hibernate-Session-Which-one-to-use)  [Lista klas związanych ze standardem JPA](https://docs.oracle.com/javaee/5/api/javax/persistence/package-summary.html" \t "/home/emil/Documents\\x/_blank) (EN)  [Oficjalny tutorial Java Persistence API](https://docs.oracle.com/javaee/5/tutorial/doc/bnbpy.html" \t "/home/emil/Documents\\x/_blank) (EN)  [Tutorial Spring + JPA (polecam po 2 części lekcji)](https://spring.io/guides/gs/accessing-data-jpa/" \t "/home/emil/Documents\\x/_blank) (EN)  [Lista klas związanych ze standardem JPA](https://docs.oracle.com/javaee/5/api/javax/persistence/package-summary.html" \t "/home/emil/Documents\\x/_blank) (EN)  [Oficjalny tutorial Java Persistence API](https://docs.oracle.com/javaee/5/tutorial/doc/bnbpy.html" \t "/home/emil/Documents\\x/_blank) (EN)  [Tutorial Spring + JPA](https://spring.io/guides/gs/accessing-data-jpa/" \t "/home/emil/Documents\\x/_blank) (EN)  [Tutorial HQL (JPQL)](http://www.mkyong.com/hibernate/hibernate-query-examples-hql/" \t "/home/emil/Documents\\x/_blank) (EN)  [Dokumentacja JPQL](https://docs.oracle.com/cd/E17904_01/apirefs.1111/e13946/ejb3_langref.html" \t "/home/emil/Documents\\x/_blank) (EN)  [Inny tutorial HQL](https://www.tutorialspoint.com/hibernate/hibernate_query_language.htm" \t "/home/emil/Documents\\x/_blank) (EN)  [Strona wikipedii na temat JPQL](https://en.wikipedia.org/wiki/Java_Persistence_Query_Language" \t "/home/emil/Documents\\x/_blank) (EN) |