

Міністерство освіти і науки України

Національний технічний університет України “Київський політехнічний інститут”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота № 7

з курсу

“ Технології розробки корпоративних застосувань”

Виконав: студент ІIІ курсу групи ІП – 42

Водотієць Денис Ігорович

Перевірив:

Подрубайло О.О.

Київ – 2016

**Тема**

Засвоєння технологій веб-сервісів JAX-WS/JAX-RS.

**Завдання**

Завдання виконується на базі лабораторної роботи №5.

У веб-застосуванні, яке було розроблене при виконанні лабораторної роботи №5 виконати наступне:

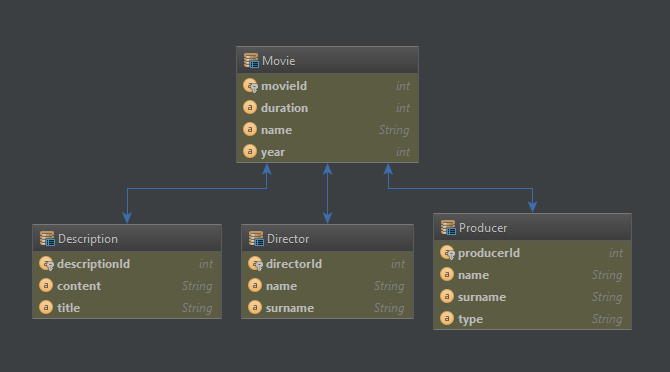
1) Реалізувати точку монтування веб-сервісу в вигляді EJB компоненту на базі технології JAX-WS. Здійснити виклик розробленого веб-сервісу із веб-застосування.

2) Реалізувати веб-сервіс на базі технології JAX-RS.

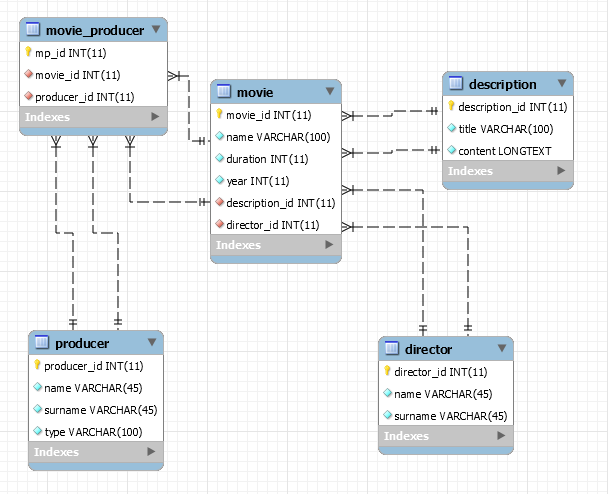
**Варіант 3**

«Фільмотека». В БД зберігається інформація про фільм (назва, режисер, тривалість, рік випуску, короткий опис тощо).

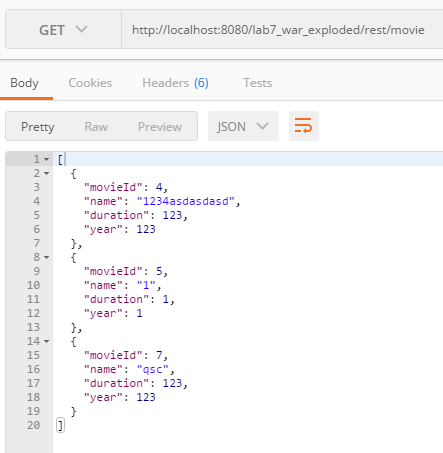
**ER-діаграма моделі даних**

****

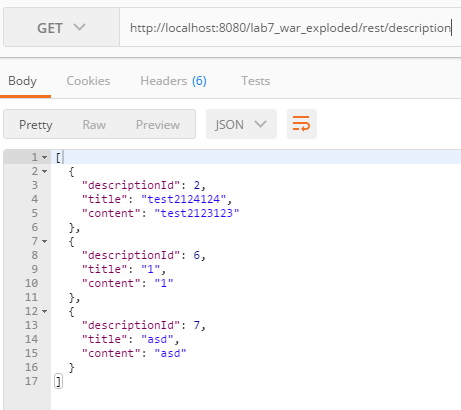
**Діаграма фізичної моделі даних**

****

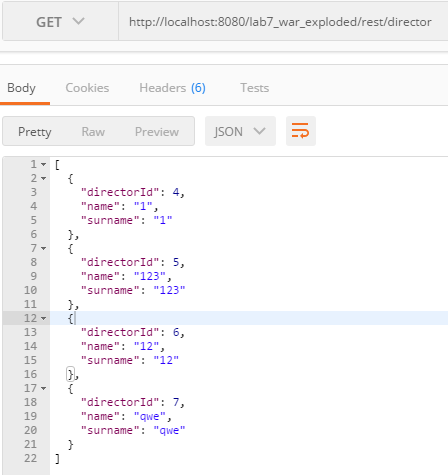
**Тестування**



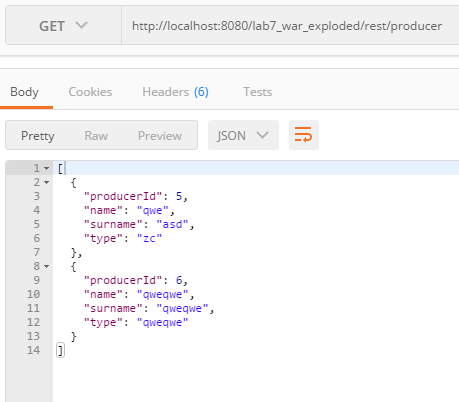
GET запит для отримання всіх фільмів



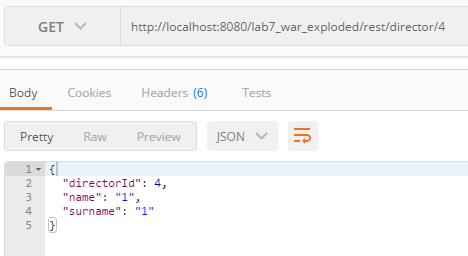
GET запит для отримання всіх описів



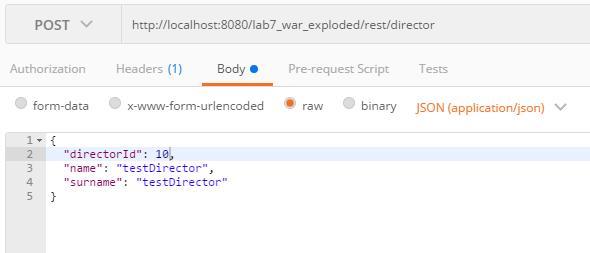
GET запит для отримання всіх режисерів



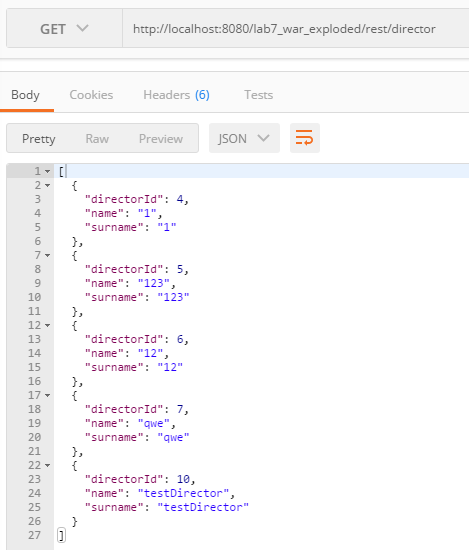
GET запит для отримання всіх продюсерів



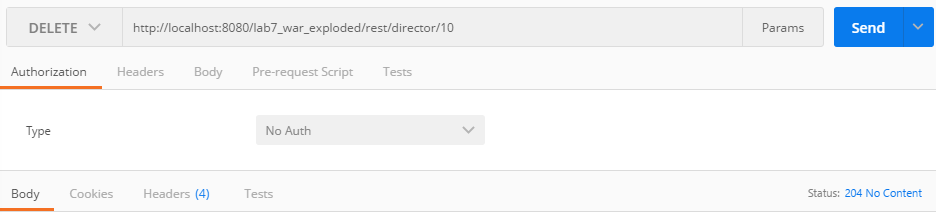
GET запит для отримання режисера по заданому id



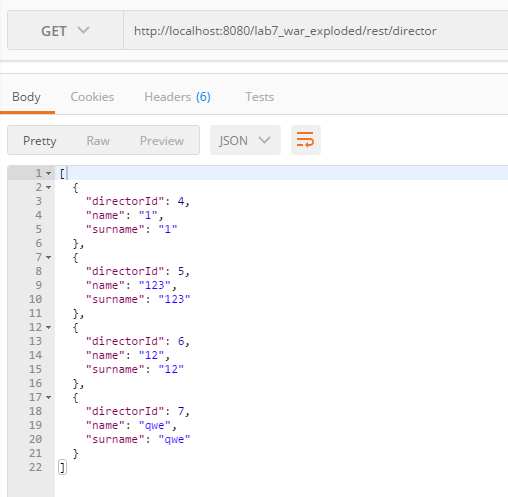
POST запит для додання режисера



GET запит для отримання всіх режисерів після додавання



DELETE запит для видалення режисера з заданим id



GET запит для отримання всіх режисерів після видалення

**Вихідні коди**

**AbstractDAO.java**

package dao;

import org.apache.log4j.Logger;

import javax.ejb.TransactionAttribute;

import javax.ejb.TransactionAttributeType;

import javax.persistence.EntityManager;

import javax.persistence.PersistenceContext;

import java.util.List;

/\*\*

\* Created by win10 on 29.10.2016.

\*/

public abstract class AbstractDAO<T> {

protected final static Logger logger = Logger.getLogger(AbstractDAO.class);

@PersistenceContext(unitName = "movies")

protected EntityManager entityManager;

@TransactionAttribute(TransactionAttributeType.NEVER)

public abstract List<T> getAll();

@TransactionAttribute(TransactionAttributeType.MANDATORY)

public abstract T getById(int id);

@TransactionAttribute(TransactionAttributeType.REQUIRED)

public void add(T entity) {

try {

entityManager.merge(entity);

} catch (Exception e) {

logger.error("An error occurred while adding an entity(" + entity + "). Error: " + e.getMessage());

throw e;

}

logger.info("An entity(" + entity + ") was added.");

}

@TransactionAttribute(TransactionAttributeType.REQUIRES\_NEW)

public void update(T entity) {

try {

entityManager.merge(entity);

} catch (Exception e) {

logger.error("An error occurred while updating an entity(" + entity + "). Error: " + e.getMessage());

throw e;

}

logger.info("Entity(" + entity + ") was updated.");

}

@TransactionAttribute(TransactionAttributeType.REQUIRED)

public void delete(T entity) {

entityManager.remove(entityManager.merge(entity));

logger.info("An entity(" + entity + ") was deleted.");

}

@TransactionAttribute(TransactionAttributeType.REQUIRED)

public void deleteById(int id) {

entityManager.remove(getById(id));

logger.info("An entity(id =" + id + ") was deleted.");

}

}

**DescriptionDAO.java**

package dao;

import entities.Description;

import javax.ejb.\*;

import javax.persistence.TypedQuery;

import java.util.List;

/\*\*

\* Created by win10 on 29.10.2016.

\*/

@Stateless

@TransactionManagement(TransactionManagementType.CONTAINER)

public class DescriptionDAO extends AbstractDAO<Description>{

@TransactionAttribute(TransactionAttributeType.SUPPORTS)

@Override

public List<Description> getAll() {

TypedQuery<Description> namedQuery = entityManager.createNamedQuery("Description.getAll", Description.class);

List<Description> descriptions = namedQuery.getResultList();

logger.info("All entity from the table 'description' were received.");

return descriptions;

}

@TransactionAttribute(TransactionAttributeType.NOT\_SUPPORTED)

@Override

public Description getById(int id) {

logger.info("An entity(id =" + id + ") from the table 'description' was received.");

return entityManager.find(Description.class, id);

}

}

**DirectorDAO.java**

package dao;

import entities.Director;

import javax.ejb.Stateless;

import javax.ejb.TransactionManagement;

import javax.ejb.TransactionManagementType;

import javax.persistence.TypedQuery;

import java.util.List;

/\*\*

\* Created by win10 on 29.10.2016.

\*/

@Stateless

@TransactionManagement(TransactionManagementType.CONTAINER)

public class DirectorDAO extends AbstractDAO<Director> {

@Override

public List<Director> getAll() {

TypedQuery<Director> namedQuery = entityManager.createNamedQuery("Director.getAll", Director.class);

List<Director> directors = namedQuery.getResultList();

logger.info("All entity from the table 'director' were received.");

return directors;

}

@Override

public Director getById(int id) {

Director director = entityManager.find(Director.class, id);

logger.info("An entity(id = " + id + ") from the table 'director' was received.");

return director;

}

}

**MovieDAO.java**

package dao;  
  
import entities.Movie;  
  
import javax.ejb.Stateless;  
import javax.ejb.TransactionManagement;  
import javax.ejb.TransactionManagementType;  
import javax.persistence.TypedQuery;  
import java.util.List;  
  
*/\*\*  
 \* Created by win10 on 29.10.2016.  
 \*/*@Stateless  
@TransactionManagement(TransactionManagementType.*CONTAINER*)  
public class MovieDAO extends AbstractDAO<Movie>{  
  
 @Override  
 public List<Movie> getAll() {  
 TypedQuery<Movie> namedQuery = entityManager.createNamedQuery("Movie.getAll", Movie.class);  
 List<Movie> movies = namedQuery.getResultList();  
  
 *logger*.info("All entity from the table 'movie' were received.");  
  
 return movies;  
 }  
  
 @Override  
 public Movie getById(int id) {  
 Movie movie = entityManager.find(Movie.class, id);  
  
 *logger*.info("An entity(id = " + id + ") from the table 'movie' was received.");  
  
 return movie;  
 }  
  
}

**ProdeucerDAO.java**

package dao;  
  
import entities.Producer;  
import org.apache.log4j.Logger;  
  
import javax.annotation.Resource;  
import javax.ejb.Stateless;  
import javax.ejb.TransactionManagement;  
import javax.ejb.TransactionManagementType;  
import javax.persistence.EntityManager;  
import javax.persistence.PersistenceContext;  
import javax.persistence.TypedQuery;  
import javax.transaction.UserTransaction;  
import java.util.List;  
  
*/\*\*  
 \* Created by win10 on 29.10.2016.  
 \*/*@Stateless  
@TransactionManagement(TransactionManagementType.*BEAN*)  
public class ProducerDAO {  
  
 protected final static Logger *logger* = Logger.*getLogger*(ProducerDAO.class);  
  
 @Resource  
 private UserTransaction userTransaction;  
  
 @PersistenceContext(unitName = "movies")  
 protected EntityManager entityManager;  
  
 public void add(Producer entity) {  
  
 try {  
 userTransaction.begin();  
 entityManager.merge(entity);  
 userTransaction.commit();  
 } catch (Exception e) {  
 *logger*.error("An error occurred while adding a producer(" + entity + "). Error: " + e.getMessage());  
 return;  
 }  
 *logger*.info("A producer(" + entity + ") was added.");  
 }  
  
 public void update(Producer entity) {  
 try {  
 userTransaction.begin();  
 entityManager.merge(entity);  
 userTransaction.commit();  
 } catch (Exception e) {  
 *logger*.error("An error occurred while updating a producer(" + entity + "). Error: " + e.getMessage());  
 return;  
 }  
  
 *logger*.info("A producer(" + entity + ") was updated.");  
 }  
  
 public void delete(Producer entity) {  
 try {  
 userTransaction.begin();  
 entityManager.remove(entityManager.merge(entity));  
 userTransaction.commit();  
 } catch (Exception e) {  
 *logger*.error("An error occurred while deleting a producer(" + entity + "). Error: " + e.getMessage());  
 return;  
 }  
  
 *logger*.info("An entity(" + entity + ") was deleted.");  
 }  
  
 public void deleteById(int id) {  
  
 try {  
 userTransaction.begin();  
 entityManager.remove(getById(id));  
 userTransaction.commit();  
 } catch (Exception e) {  
 *logger*.error("An error occurred while deleting a producer(id = " + id + "). Error: " + e.getMessage());  
 return;  
 }  
  
 *logger*.info("An entity(id =" + id + ") was deleted.");  
 }  
  
 public List<Producer> getAll() {  
 TypedQuery<Producer> namedQuery = entityManager.createNamedQuery("Producer.getAll", Producer.class);  
 List<Producer> producers = namedQuery.getResultList();  
  
 *logger*.info("All entity from the table 'producer' were received.");  
  
 return producers;  
 }  
  
 public Producer getById(int id) {  
 Producer producer = entityManager.find(Producer.class, id);  
  
 *logger*.info("An entity(id = " + id + ") from the table 'producer' was received.");  
  
 return producer;  
 }  
  
}

**SearchDAO.java**

package dao;

import entities.Director;

import entities.Movie;

import org.apache.log4j.Logger;

import javax.ejb.\*;

import java.util.List;

import java.util.Random;

/\*\*

\* Created by win10 on 14.11.2016.

\*/

@Stateless

@TransactionManagement(TransactionManagementType.CONTAINER)

public class SearchDAO {

private final static Logger logger = Logger.getLogger(SearchDAO.class);

@EJB

private DirectorDAO directorDAO;

@TransactionAttribute(TransactionAttributeType.REQUIRED)

public void searchDirector(Movie movie) {

Director director = movie.getDirector();

try {

offer(director);

} catch (RuntimeException e) {

logger.error("SEARCH:: Transaction error! Director (" + director + ") did not agree.");

List<Director> directors = directorDAO.getAll();

for (Director director1 : directors) {

if (!director1.equals(director)) {

try {

offer(director1);

logger.info("Director (" + director1 + ") agreed");

} catch (RuntimeException e1) {

logger.error("SEARCH:: Transaction error! Director (" + director1 + ") did not agree.");

}

}

}

}

logger.info("SEARCH:: Transaction finished.");

}

@TransactionAttribute(TransactionAttributeType.REQUIRES\_NEW)

public boolean offer(Director director) {

Random random = new Random();

boolean result = random.nextInt(10) > 5;

if (!result) {

logger.error("OFFER:: Transaction error! Director (" + director + ") did not agree");

throw new RuntimeException("Director (" + director + ") did not agree");

}

logger.error("OFFER:: Director (" + director + ") agreed.");

return result;

}

}

**Description.java**

package entities;  
  
import org.codehaus.jackson.annotate.JsonManagedReference;  
  
import javax.persistence.\*;  
import javax.xml.bind.annotation.XmlRootElement;  
import javax.xml.bind.annotation.XmlTransient;  
import java.io.Serializable;  
  
*/\*\*  
 \* Created by win10 on 29.10.2016.  
 \*/*@Entity  
@XmlRootElement  
@Table(name = "description")  
@NamedQuery(name = "Description.getAll", query = "SELECT d FROM Description d")  
public class Description implements Serializable{  
  
 @Id  
 @Column(name = "description\_id", unique = true, nullable = false)  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private int descriptionId;  
  
 @Column(name = "title", unique = true, nullable = false)  
 private String title;  
  
 @Column(name = "content", nullable = false)  
 private String content;  
  
 @OneToOne(mappedBy = "description")  
 @XmlTransient  
 @JsonManagedReference  
 private Movie movie;  
  
 public Description() {}  
  
 public Description(String title, String content) {  
 this.title = title;  
 this.content = content;  
 }  
  
 public Description(String title, String content, Movie movie) {  
 this.title = title;  
 this.content = content;  
 this.movie = movie;  
 }  
  
 public int getDescriptionId() {  
 return descriptionId;  
 }  
  
 public void setDescriptionId(int descriptionId) {  
 this.descriptionId = descriptionId;  
 }  
  
 public String getTitle() {  
 return title;  
 }  
  
 public void setTitle(String title) {  
 this.title = title;  
 }  
  
 public String getContent() {  
 return content;  
 }  
  
 public void setContent(String content) {  
 this.content = content;  
 }  
  
 @XmlTransient  
 public Movie getMovie() {  
 return movie;  
 }  
  
 public void setMovie(Movie movie) {  
 this.movie = movie;  
  
 if (movie != null) {  
 movie.setDescription(this);  
 }  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
  
 Description that = (Description) o;  
  
 if (getDescriptionId() != that.getDescriptionId()) return false;  
 if (getTitle() != null ? !getTitle().equals(that.getTitle()) : that.getTitle() != null) return false;  
 return getContent() != null ? getContent().equals(that.getContent()) : that.getContent() == null;  
  
 }  
  
 @Override  
 public int hashCode() {  
 int result = getDescriptionId();  
 result = 31 \* result + (getTitle() != null ? getTitle().hashCode() : 0);  
 result = 31 \* result + (getContent() != null ? getContent().hashCode() : 0);  
 return result;  
 }  
  
 @Override  
 public String toString() {  
 StringBuilder stringBuilder = new StringBuilder("Description{title='").append(title)  
 .append("', content='").append(content).append("}");  
  
 return stringBuilder.toString();  
 }  
}

**Director.java**

package entities;  
  
import org.codehaus.jackson.annotate.JsonManagedReference;  
  
import javax.persistence.\*;  
import javax.xml.bind.annotation.XmlRootElement;  
import javax.xml.bind.annotation.XmlTransient;  
import java.io.Serializable;  
import java.util.HashSet;  
import java.util.Iterator;  
import java.util.Set;  
  
*/\*\*  
 \* Created by win10 on 29.10.2016.  
 \*/*@Entity  
@XmlRootElement  
@Table(name = "director")  
@NamedQuery(name = "Director.getAll", query = "SELECT d FROM Director d")  
public class Director implements Serializable {  
  
 @Id  
 @Column(name = "director\_id", unique = true, nullable = false)  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private int directorId;  
  
 @Column(name = "name", nullable = false)  
 private String name;  
  
 @Column(name = "surname", nullable = false)  
 private String surname;  
  
 @OneToMany(mappedBy = "director")  
 @JsonManagedReference  
 @XmlTransient  
 private Set<Movie> movies;  
  
 public Director() {  
 movies = new HashSet<>();  
 }  
  
 public Director(String name, String surname) {  
 this();  
 this.name = name;  
 this.surname = surname;  
 }  
  
 public Director(String name, String surname, Set<Movie> movies) {  
 this.name = name;  
 this.surname = surname;  
  
 this.movies = movies;  
 for (Movie movie : movies) {  
 movie.setDirector(this);  
 }  
 }  
  
 public void addMovie(Movie movie) {  
 movies.add(movie);  
 movie.setDirector(this);  
 }  
  
 public void removeMovie(int movieId) {  
 Iterator<Movie> movieIterator = movies.iterator();  
 while (movieIterator.hasNext()) {  
 if(movieIterator.next().getMovieId() == movieId) {  
 movieIterator.remove();  
 return;  
 }  
 }  
 }  
  
 public void removeMovie(Movie movie) {  
 movies.remove(movie);  
 }  
  
 public void setMovies(Set<Movie> movies) {  
 if (this.movies != null) {  
 Iterator<Movie> movieIterator = this.movies.iterator();  
  
 while (movieIterator.hasNext()) {  
 Movie movie = movieIterator.next();  
 movie.setDirector(null);  
 }  
 }  
  
 this.movies = movies;  
 for (Movie movie : movies) {  
 movie.setDirector(this);  
 }  
 }  
  
 public int getDirectorId() {  
 return directorId;  
 }  
  
 public void setDirectorId(int directorId) {  
 this.directorId = directorId;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getSurname() {  
 return surname;  
 }  
  
 public void setSurname(String surname) {  
 this.surname = surname;  
 }  
  
 @XmlTransient  
 public Set<Movie> getMovies() {  
 return movies;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
  
 Director director = (Director) o;  
  
 if (getDirectorId() != director.getDirectorId()) return false;  
 if (getName() != null ? !getName().equals(director.getName()) : director.getName() != null) return false;  
 return getSurname() != null ? getSurname().equals(director.getSurname()) : director.getSurname() == null;  
  
 }  
  
 @Override  
 public int hashCode() {  
 int result = getDirectorId();  
 result = 31 \* result + (getName() != null ? getName().hashCode() : 0);  
 result = 31 \* result + (getSurname() != null ? getSurname().hashCode() : 0);  
 return result;  
 }  
  
 @Override  
 public String toString() {  
 StringBuilder stringBuilder = new StringBuilder("Director{ ")  
 .append(name).append(" ").append(surname).append("}");  
  
 return stringBuilder.toString();  
 }  
}

**Movie.java**

package entities;  
  
import org.codehaus.jackson.annotate.JsonBackReference;  
  
import javax.persistence.\*;  
import javax.xml.bind.annotation.XmlRootElement;  
import java.io.Serializable;  
import java.util.ArrayList;  
import java.util.Iterator;  
import java.util.List;  
  
*/\*\*  
 \* Created by win10 on 29.10.2016.  
 \*/*@Entity  
@XmlRootElement  
@Table(name = "movie")  
@NamedQuery(name = "Movie.getAll", query = "SELECT m FROM Movie m")  
public class Movie implements Serializable {  
  
 @Id  
 @Column(name = "movie\_id", nullable = false)  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private int movieId;  
  
 @Column(name = "name", nullable = false)  
 private String name;  
  
 @Column(name = "duration", nullable = false)  
 private int duration;  
  
 @Column(name = "year", nullable = false)  
 private int year;  
  
 @OneToOne(cascade = CascadeType.*MERGE*)  
 @JoinColumn(name = "description\_id", unique = true, nullable = false)  
 @JsonBackReference(value = "movieDescription")  
 private Description description;  
  
 @ManyToOne(cascade = CascadeType.*MERGE*)  
 @JoinColumn(name = "director\_id", nullable = false)  
 @JsonBackReference(value = "movieDirector")  
 private Director director;  
  
 @ManyToMany(fetch = FetchType.*EAGER*)  
 @JoinTable(name = "movie\_producer",  
 joinColumns = {@JoinColumn(name = "movie\_id", referencedColumnName = "movie\_id")},  
 inverseJoinColumns = {@JoinColumn(name = "producer\_id", referencedColumnName = "producer\_id")})  
 @JsonBackReference(value = "movieAllProducers")  
 private List<Producer> producers;  
  
 public Movie() {  
 producers = new ArrayList<>();  
 }  
  
 public Movie(String name, int duration, int year, Description description, Director director) {  
 this();  
 this.name = name;  
 this.duration = duration;  
 this.year = year;  
 this.director = director;  
 this.description = description;  
 }  
  
 public Movie(String name, int duration, int year, Description description, Director director, List<Producer> producers) {  
 this.name = name;  
 this.duration = duration;  
 this.year = year;  
 this.description = description;  
 this.director = director;  
 this.producers = producers;  
 }  
  
 public void addProducer(Producer producer) {  
 if (producers.add(producer)) {  
 if (!producer.getMovies().contains(this)) {  
 producer.addMovie(this);  
 }  
 }  
 }  
  
 public void removeProducer(int producerId) {  
 for (Producer producer : producers) {  
 if(producer.getProducerId() == producerId) {  
 producers.remove(producer);  
 producer.removeMovie(this);  
 return;  
 }  
 }  
 }  
  
 public void removeProducer(Producer producer) {  
 if (producers.remove(producer)) {  
 producer.removeMovie(this);  
 }  
 }  
  
 public void setProducers(List<Producer> producers) {  
 if (this.producers != null) {  
 Iterator<Producer> producerIterator = this.producers.iterator();  
 while (producerIterator.hasNext()) {  
 Producer producer = producerIterator.next();  
 producer.removeMovie(this);  
 }  
 }  
  
 this.producers = producers;  
 for (Producer producer : producers) {  
 producer.addMovie(this);  
 }  
 }  
  
 public List<Producer> getProducers() {  
 return producers;  
 }  
  
 public int getMovieId() {  
 return movieId;  
 }  
  
 public void setMovieId(int movieId) {  
 this.movieId = movieId;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public int getDuration() {  
 return duration;  
 }  
  
 public void setDuration(int duration) {  
 this.duration = duration;  
 }  
  
 public int getYear() {  
 return year;  
 }  
  
 public void setYear(int year) {  
 this.year = year;  
 }  
  
 public Description getDescription() {  
 return description;  
 }  
  
 public void setDescription(Description description) {  
 if (this.description == null) {  
 this.description = description;  
 description.setMovie(this);  
 return;  
 }  
  
 if (this.description.equals(description)) {  
 return;  
 }  
  
 this.description.setMovie(null);  
 this.description = description;  
 description.setMovie(this);  
  
 }  
  
 public Director getDirector() {  
 return director;  
 }  
  
 public void setDirector(Director director) {  
 if (this.director != null) {  
 if (this.director.equals(director)) {  
 return;  
 }  
  
 this.director.removeMovie(this);  
 this.director = director;  
 director.addMovie(this);  
 } else {  
 this.director = director;  
 }  
  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
  
 Movie movie = (Movie) o;  
  
 if (getMovieId() != movie.getMovieId()) return false;  
 if (getDuration() != movie.getDuration()) return false;  
 if (getYear() != movie.getYear()) return false;  
 if (getName() != null ? !getName().equals(movie.getName()) : movie.getName() != null) return false;  
 if (getDescription() != null ? !getDescription().equals(movie.getDescription()) : movie.getDescription() != null)  
 return false;  
 return getDirector() != null ? getDirector().equals(movie.getDirector()) : movie.getDirector() == null;  
  
 }  
  
 @Override  
 public int hashCode() {  
 int result = getMovieId();  
 result = 31 \* result + (getName() != null ? getName().hashCode() : 0);  
 result = 31 \* result + getDuration();  
 result = 31 \* result + getYear();  
 result = 31 \* result + (getDescription() != null ? getDescription().hashCode() : 0);  
 result = 31 \* result + (getDirector() != null ? getDirector().hashCode() : 0);  
 return result;  
 }  
  
 @Override  
 public String toString() {  
 StringBuilder stringBuilder = new StringBuilder("Movie{name='").append(name).append("',duration=")  
 .append(duration).append("sec, year=").append(year).append(", description=[title=")  
 .append(description.getTitle()).append(",content=").append(description.getContent())  
 .append("], director=").append(director.getName()).append(" ").append(director.getSurname())  
 .append(", producers={");  
  
 if (producers != null && producers.size() > 0) {  
 for (Producer producer : producers) {  
 stringBuilder.append(producer.getName()).append(" ").append(producer.getSurname())  
 .append(", type=").append(producer.getType()).append(";");  
 }  
 } else {  
 stringBuilder.append("}");  
 }  
  
 return stringBuilder.append("}").toString();  
 }  
}

**Producer.java**

package entities;  
  
import org.codehaus.jackson.annotate.JsonManagedReference;  
  
import javax.persistence.\*;  
import javax.xml.bind.annotation.XmlRootElement;  
import javax.xml.bind.annotation.XmlTransient;  
import java.io.Serializable;  
import java.util.ArrayList;  
import java.util.Iterator;  
import java.util.List;  
  
*/\*\*  
 \* Created by win10 on 29.10.2016.  
 \*/*@Entity  
@XmlRootElement  
@Table(name = "producer")  
@NamedQuery(name = "Producer.getAll", query = "SELECT p FROM Producer p")  
public class Producer implements Serializable {  
  
 @Id  
 @Column(name = "producer\_id", unique = true, nullable = false)  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private int producerId;  
  
 @Column(name = "name", nullable = false)  
 private String name;  
  
 @Column(name = "surname", nullable = false)  
 private String surname;  
  
 @Column(name = "type", nullable = false)  
 private String type;  
  
 @ManyToMany(mappedBy = "producers", fetch = FetchType.*EAGER*)  
 @JsonManagedReference  
 @XmlTransient  
 private List<Movie> movies;  
  
 public Producer() {  
 movies = new ArrayList<>();  
 }  
  
 public Producer(String name, String surname, String type) {  
 this();  
 this.name = name;  
 this.surname = surname;  
 this.type = type;  
 }  
  
 public Producer(String name, String surname, String type, List<Movie> movies) {  
 this.name = name;  
 this.surname = surname;  
 this.type = type;  
 this.movies = movies;  
 }  
  
 public void addMovie(Movie movie) {  
 if (movies.add(movie)) {  
 if (!movie.getProducers().contains(this)) {  
 movie.addProducer(this);  
 }  
 }  
 }  
  
 public void removeMovie(int movieId) {  
 for (Movie movie : movies) {  
 if(movie.getMovieId() == movieId) {  
 movies.remove(movie);  
 movie.removeProducer(this);  
 return;  
 }  
 }  
 }  
  
 public void removeMovie(Movie movie) {  
 if (movies.remove(movie)) {  
 movie.removeProducer(this);  
 }  
 }  
  
 public void setMovies(List<Movie> movies) {  
 if (this.movies != null) {  
 Iterator<Movie> movieIterator = this.movies.iterator();  
 movieIterator.next();  
 while (movieIterator.hasNext()) {  
 Movie movie = movieIterator.next();  
 movie.removeProducer(this);  
 }  
 }  
  
 this.movies = movies;  
 for (Movie movie : movies) {  
 movie.addProducer(this);  
 }  
 }  
  
 public int getProducerId() {  
 return producerId;  
 }  
  
 public void setProducerId(int producerId) {  
 this.producerId = producerId;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
  
 this.name = name;  
  
 }  
  
 public String getSurname() {  
 return surname;  
 }  
  
 public void setSurname(String surname) {  
 this.surname = surname;  
 }  
  
 public String getType() {  
 return type;  
 }  
  
 public void setType(String type) {  
 this.type = type;  
 }  
  
 @XmlTransient  
 public List<Movie> getMovies() {  
 return movies;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
  
 Producer producer = (Producer) o;  
  
 if (getProducerId() != producer.getProducerId()) return false;  
 if (getName() != null ? !getName().equals(producer.getName()) : producer.getName() != null) return false;  
 if (getSurname() != null ? !getSurname().equals(producer.getSurname()) : producer.getSurname() != null)  
 return false;  
 return getType() != null ? getType().equals(producer.getType()) : producer.getType() == null;  
  
 }  
  
 @Override  
 public int hashCode() {  
 int result = getProducerId();  
 result = 31 \* result + (getName() != null ? getName().hashCode() : 0);  
 result = 31 \* result + (getSurname() != null ? getSurname().hashCode() : 0);  
 result = 31 \* result + (getType() != null ? getType().hashCode() : 0);  
 return result;  
 }  
  
 @Override  
 public String toString() {  
 StringBuilder stringBuilder = new StringBuilder("Producer{").append(name).append(" ").append(surname)  
 .append(", type=").append(type).append("}");  
  
 return stringBuilder.toString();  
 }  
}

**DescriptionValidator.java**

package managedBean.validators;  
  
import org.apache.log4j.Logger;  
import wsClient.Description;  
import wsClient.DescriptionWS;  
import wsClient.DescriptionWSImpImplementationService;  
  
import javax.annotation.PostConstruct;  
import javax.faces.application.FacesMessage;  
import javax.faces.bean.ManagedBean;  
import javax.faces.bean.RequestScoped;  
import javax.faces.component.UIComponent;  
import javax.faces.context.FacesContext;  
import javax.faces.validator.Validator;  
import javax.faces.validator.ValidatorException;  
import java.util.List;  
  
  
*/\*\*  
 \* Created by win10 on 08.11.2016.  
 \*/*@ManagedBean  
@RequestScoped  
public class DescriptionValidator implements Validator {  
  
 private final static Logger *LOGGER* = Logger.*getLogger*(DescriptionValidator.class);  
 private List<Description> descriptions;  
  
 private DescriptionWS descriptionDAO;  
  
 @PostConstruct  
 public void init() {  
 DescriptionWSImpImplementationService descriptionWSImpImplementationService  
 = new DescriptionWSImpImplementationService();  
 descriptionDAO = descriptionWSImpImplementationService.getDescriptionWSImpImplementationPort();  
 }  
  
 @Override  
 public void validate(FacesContext facesContext, UIComponent uiComponent, Object o) throws ValidatorException {  
  
 String operation = (String)uiComponent.getAttributes().get("parameter");  
  
 if (operation.equals("title")) {  
 String title = o.toString();  
  
 if (title.length() < 1 || title.length() > 100) {  
 *LOGGER*.warn("The title must be in the range [1;100]!");  
 throw new ValidatorException(new FacesMessage("The title must be in the range [1;100]!"));  
 }  
  
 descriptions = descriptionDAO.getAllDescriptions();  
 if (descriptions != null) {  
 if (uiComponent.getAttributes().get("update").equals("no")) {  
 for (Description description : descriptions) {  
 if (description.getTitle().equals(title)) {  
 *LOGGER*.warn("The title '" + title + "' exists. The title must be unique");  
 throw new ValidatorException(new FacesMessage("The title '"  
 + title + "' exists. The title must be unique"));  
 }  
 }  
 }  
 }  
  
 } else {  
 String content = o.toString();  
  
 if (content.length() < 1) {  
 *LOGGER*.warn("The content was missing!");  
 throw new ValidatorException(new FacesMessage("The content was missing!"));  
 }  
 }  
  
 }  
}

**DirectorValidator.java**

package managedBean.validators;  
  
import org.apache.log4j.Logger;  
  
import javax.faces.application.FacesMessage;  
import javax.faces.bean.ManagedBean;  
import javax.faces.bean.RequestScoped;  
import javax.faces.component.UIComponent;  
import javax.faces.context.FacesContext;  
import javax.faces.validator.Validator;  
import javax.faces.validator.ValidatorException;  
  
*/\*\*  
 \* Created by win10 on 08.11.2016.  
 \*/*@ManagedBean  
@RequestScoped  
public class DirectorValidator implements Validator {  
  
 private final static Logger LOGGER = Logger.getLogger(DirectorValidator.class);  
  
 @Override  
 public void validate(FacesContext facesContext, UIComponent uiComponent, Object o) throws ValidatorException {  
  
 String operation = (String) uiComponent.getAttributes().get("parameter");  
  
 String name = o.toString();  
  
 if (name.length() < 1 || name.length() > 45) {  
 if (operation.equals("name")) {  
 LOGGER.warn("The name must be in the range [1;45]!");  
 throw new ValidatorException(new FacesMessage("The name must be in the range [1;45]!"));  
 } else {  
 LOGGER.warn("The surname must be in the range [1;45]!");  
 throw new ValidatorException(new FacesMessage("The surname must be in the range [1;45]!"));  
 }  
 }  
  
 }  
  
}

**MovieValidator.java**

package managedBean.validators;  
  
import org.apache.log4j.Logger;  
import wsClient.\*;  
  
import javax.annotation.PostConstruct;  
import javax.faces.application.FacesMessage;  
import javax.faces.bean.ManagedBean;  
import javax.faces.bean.RequestScoped;  
import javax.faces.component.UIComponent;  
import javax.faces.context.FacesContext;  
import javax.faces.validator.Validator;  
import javax.faces.validator.ValidatorException;  
  
*/\*\*  
 \* Created by win10 on 08.11.2016.  
 \*/*@ManagedBean  
@RequestScoped  
public class MovieValidator implements Validator {  
  
 private final static Logger *LOGGER* = Logger.*getLogger*(MovieValidator.class);  
  
 private DescriptionWS descriptionDAO;  
  
 private DirectorWS directorDAO;  
  
 @PostConstruct  
 public void init() {  
 DescriptionWSImpImplementationService descriptionWSImpImplementationService  
 = new DescriptionWSImpImplementationService();  
 descriptionDAO = descriptionWSImpImplementationService.getDescriptionWSImpImplementationPort();  
  
 DirectorWSImpImplementationService directorWSImpImplementationService = new DirectorWSImpImplementationService();  
 directorDAO = directorWSImpImplementationService.getDirectorWSImpImplementationPort();  
 }  
  
 @Override  
 public void validate(FacesContext facesContext, UIComponent uiComponent, Object o) throws ValidatorException {  
  
 String parameter = (String)uiComponent.getAttributes().get("parameter");  
  
 if (parameter.equals("name")) {  
 String name = o.toString();  
  
 if (name.length() < 1 || name.length() > 100) {  
 *LOGGER*.warn("The name must be in the range [1;100]!");  
 throw new ValidatorException(new FacesMessage("The name must be in the range [1;100]!"));  
 }  
 } else if (parameter.equals("duration")) {  
 String durationStr = o.toString();  
 check(durationStr, "duration");  
 } else if (parameter.equals("year")) {  
 String yearStr = o.toString();  
 check(yearStr, "year");  
 } else if (parameter.equals("descriptionId")) {  
 String descriptionStr = o.toString();  
 int descriptionId = check(descriptionStr, "descriptionId");  
  
 Description description = descriptionDAO.getDescriptionById(descriptionId);  
 if (description == null) {  
 *LOGGER*.warn("The description with id =" + descriptionId + " does not exist.");  
 throw new ValidatorException(new FacesMessage("the description with id =" + descriptionId + " does not exist"));  
 }  
 } else if (parameter.equals("directorId")) {  
 String directorStr = o.toString();  
 int directorId = check(directorStr, "descriptionId");  
  
 Director director = directorDAO.getDirectorById(directorId);  
 if (director == null) {  
 *LOGGER*.warn("The director with id =" + directorId + " does not exist.");  
 throw new ValidatorException(new FacesMessage("the director with id =" + directorId + " does not exist"));  
 }  
 }  
  
 }  
  
 private int check(String parameter, String name) {  
 int number;  
 try {  
 number = Integer.*parseInt*(parameter);  
 } catch (Exception e) {  
 *LOGGER*.warn("The wrong value of " + name + "!", e);  
 throw new ValidatorException(new FacesMessage("The wrong value of " + name + "!"));  
 }  
 if (number < 1) {  
 *LOGGER*.warn("The wrong value of " + name + "!");  
 throw new ValidatorException(new FacesMessage("The wrong value of " + name + "!"));  
 }  
  
 return number;  
 }  
}

**MPValidator.java**

package managedBean.validators;  
  
import org.apache.log4j.Logger;  
import wsClient.\*;  
  
import javax.annotation.PostConstruct;  
import javax.faces.application.FacesMessage;  
import javax.faces.bean.ManagedBean;  
import javax.faces.bean.RequestScoped;  
import javax.faces.component.UIComponent;  
import javax.faces.context.FacesContext;  
import javax.faces.validator.Validator;  
import javax.faces.validator.ValidatorException;  
  
*/\*\*  
 \* Created by win10 on 08.11.2016.  
 \*/*@ManagedBean  
@RequestScoped  
public class MPValidator implements Validator {  
 private final static Logger LOGGER = Logger.getLogger(MPValidator.class);  
  
 private MovieWS movieDAO;  
 private ProducerWS producerDAO;  
  
 @PostConstruct  
 public void init() {  
 MovieWSWSImpImplementationService movieWSWSImpImplementationService = new MovieWSWSImpImplementationService();  
 movieDAO = movieWSWSImpImplementationService.getMovieWSWSImpImplementationPort();  
  
 ProducerWSImpImplementationService producerWSImpImplementationService = new ProducerWSImpImplementationService();  
 producerDAO = producerWSImpImplementationService.getProducerWSImpImplementationPort();  
 }  
  
 public void validate(FacesContext facesContext, UIComponent uiComponent, Object o) throws ValidatorException {  
 if (uiComponent.getAttributes().get("parameter").equals("movieId")) {  
 String movieIdStr = o.toString();  
 int movieId = check(movieIdStr, "movieId");  
  
 Movie movie = movieDAO.getMovieById(movieId);  
 if (movie == null) {  
 LOGGER.warn("The movie with id =" + movieId + " does not exist.");  
 throw new ValidatorException(new FacesMessage("the movie with id =" + movieId + " does not exist"));  
 }  
 } else {  
 String producerIdStr = o.toString();  
 int producerId = check(producerIdStr, "producerId");  
  
 Producer producer = producerDAO.getProducerById(producerId);  
 if (producer == null) {  
 *LOGGER*.warn("The producer with id =" + producerId + " does not exist.");  
 throw new ValidatorException(new FacesMessage("the producer with id =" + producerId + " does not exist"));  
 }  
 }  
 }  
  
 private int check(String parameter, String name) {  
 int number;  
 try {  
 number = Integer.parseInt(parameter);  
 } catch (Exception e) {  
 LOGGER.warn("The wrong value of " + name + "!", e);  
 throw new ValidatorException(new FacesMessage("The wrong value of " + name + "!"));  
 }  
 if (number < 1) {  
 LOGGER.warn("The wrong value of " + name + "!");  
 throw new ValidatorException(new FacesMessage("The wrong value of " + name + "!"));  
 }  
  
 return number;  
 }  
}

**ProdeucerValidator.java**

package managedBean.validators;  
  
import org.apache.log4j.Logger;  
  
import javax.faces.application.FacesMessage;  
import javax.faces.bean.ManagedBean;  
import javax.faces.bean.RequestScoped;  
import javax.faces.component.UIComponent;  
import javax.faces.context.FacesContext;  
import javax.faces.validator.Validator;  
import javax.faces.validator.ValidatorException;  
  
*/\*\*  
 \* Created by win10 on 08.11.2016.  
 \*/*@ManagedBean  
@RequestScoped  
public class ProducerValidator implements Validator {  
  
 private final static Logger *LOGGER* = Logger.*getLogger*(ProducerValidator.class);  
  
 @Override  
 public void validate(FacesContext facesContext, UIComponent uiComponent, Object o) throws ValidatorException {  
  
 String operation = (String) uiComponent.getAttributes().get("parameter");  
  
 String name = o.toString();  
  
 if (name.length() < 1 || name.length() > 45) {  
 if (operation.equals("name")) {  
 *LOGGER*.warn("The name must be in the range [1;45]!");  
 throw new ValidatorException(new FacesMessage("The name must be in the range [1;45]!"));  
 } else if (operation.equals("surname")){  
 *LOGGER*.warn("The surname must be in the range [1;45]!");  
 throw new ValidatorException(new FacesMessage("The surname must be in the range [1;45]!"));  
 } else {  
 *LOGGER*.warn("The type must be in the range [1;45]!");  
 throw new ValidatorException(new FacesMessage("The type must be in the range [1;45]!"));  
 }  
 }  
  
 }  
  
}

**DescriptionController.java**

package managedBean;  
  
import org.apache.log4j.Logger;  
import wsClient.Description;  
import wsClient.DescriptionWS;  
import wsClient.DescriptionWSImpImplementationService;  
  
import javax.annotation.PostConstruct;  
import javax.faces.bean.ManagedBean;  
import javax.faces.bean.SessionScoped;  
import java.util.List;  
  
*/\*\*  
 \* Created by win10 on 07.11.2016.  
 \*/*@ManagedBean(name = "descriptionController", eager = true)  
@SessionScoped  
public class DescriptionController {  
  
 private final static Logger *logger* = Logger.*getLogger*(DescriptionController.class);  
  
 private DescriptionWS dao;  
 private Description description;  
  
 @PostConstruct  
 public void init() {  
 DescriptionWSImpImplementationService descriptionWSImpImplementationService  
 = new DescriptionWSImpImplementationService();  
 dao = descriptionWSImpImplementationService.getDescriptionWSImpImplementationPort();  
 }  
  
 public List<Description> getAll() {  
 return dao.getAllDescriptions();  
 }  
  
 public String update() {  
 dao.updateDescription(description);  
 return "/index?faces-redirect=true";  
 }  
  
 public String goToAdd() {  
 description = new Description();  
 return "JSF/add/addDescription";  
 }  
  
 public String add() {  
 dao.addDescription(description);  
 return "/index?faces-redirect=true";  
 }  
  
 public String delete(Description description) {  
 dao.deleteDescription(description);  
 return "/index?faces-redirect=true";  
 }  
  
 public String goToUpdate(Description description) {  
 this.description = description;  
 return "/JSF/update/updateDescription";  
 }  
  
 public Description getDescription() {  
 return description;  
 }  
  
 public void setDescription(Description description) {  
 this.description = description;  
 }  
  
}

**DirectorController.java**

package managedBean;  
  
import org.apache.log4j.Logger;  
import wsClient.Director;  
import wsClient.DirectorWS;  
import wsClient.DirectorWSImpImplementationService;  
  
import javax.annotation.PostConstruct;  
import javax.faces.bean.ManagedBean;  
import javax.faces.bean.SessionScoped;  
import java.util.List;  
  
*/\*\*  
 \* Created by win10 on 08.11.2016.  
 \*/*@ManagedBean(name = "directorController", eager = true)  
@SessionScoped  
public class DirectorController {  
  
 private final static Logger *logger* = Logger.getLogger(DirectorController.class);  
  
 private DirectorWS dao;  
 private Director director;  
  
 @PostConstruct  
 public void init() {  
 DirectorWSImpImplementationService directorWSImpImplementationService  
 = new DirectorWSImpImplementationService();  
 dao = directorWSImpImplementationService.getDirectorWSImpImplementationPort();  
 }  
  
 public List<Director> getAll() {  
 return dao.getAllDirectors();  
 }  
  
 public String update() {  
 dao.updateDirector(director);  
 return "/index?faces-redirect=true";  
 }  
  
 public String goToAdd() {  
 director = new Director();  
 return "JSF/add/addDirector";  
 }  
  
 public String add() {  
 dao.addDirector(director);  
 return "/index?faces-redirect=true";  
 }  
  
 public String delete(Director director) {  
 dao.deleteDirector(director);  
 return "/index?faces-redirect=true";  
 }  
  
 public String goToUpdate(Director director) {  
 this.director = director;  
 return "JSF/update/updateDirector";  
 }  
  
 public Director getDirector() {  
 return director;  
 }  
  
 public void setDirector(Director director) {  
 this.director = director;  
 }  
  
}

**MovieController.java**

package managedBean;  
  
import org.apache.log4j.Logger;  
import wsClient.\*;  
  
import javax.annotation.PostConstruct;  
import javax.faces.bean.ManagedBean;  
import javax.faces.bean.SessionScoped;  
import java.util.List;  
  
*/\*\*  
 \* Created by win10 on 07.11.2016.  
 \*/*@ManagedBean(name = "movieController", eager = true)  
@SessionScoped  
public class MovieController {  
  
 private final static Logger logger = Logger.getLogger(MovieController.class);  
  
 private MovieWS dao;  
  
 private DescriptionWS descriptionDAO;  
  
 private DirectorWS directorDAO;  
  
 @PostConstruct  
 public void init() {  
 DescriptionWSImpImplementationService descriptionWSImpImplementationService  
 = new DescriptionWSImpImplementationService();  
 descriptionDAO = descriptionWSImpImplementationService.getDescriptionWSImpImplementationPort();  
  
 DirectorWSImpImplementationService directorWSImpImplementationService  
 = new DirectorWSImpImplementationService();  
 directorDAO = directorWSImpImplementationService.getDirectorWSImpImplementationPort();  
  
 MovieWSWSImpImplementationService movieWSWSImpImplementationService  
 = new MovieWSWSImpImplementationService();  
 dao = movieWSWSImpImplementationService.getMovieWSWSImpImplementationPort();  
 }  
  
 private Movie movie;  
  
 private String durationStr;  
 private String yearStr;  
  
 private String descriptionIdStr;  
 private String directorIdStr;  
  
 public List<Movie> getAll() {  
 return dao.getAllMovies();  
 }  
  
 public String update() {  
 movie.setYear(Integer.parseInt(yearStr));  
 movie.setDuration(Integer.parseInt(durationStr));  
  
 movie.setDescription(descriptionDAO.getDescriptionById(Integer.parseInt(descriptionIdStr)));  
 movie.setDirector(directorDAO.getDirectorById(Integer.parseInt(directorIdStr)));  
 dao.updateMovie(movie);  
 return "/index?faces-redirect=true";  
 }  
  
 public String goToAdd() {  
 movie = new Movie();  
 yearStr = durationStr = descriptionIdStr = directorIdStr = null;  
 return "JSF/add/addMovie";  
 }  
  
 public String add() {  
 movie.setYear(Integer.parseInt(yearStr));  
 movie.setDuration(Integer.parseInt(durationStr));  
  
 movie.setDescription(descriptionDAO.getDescriptionById(Integer.parseInt(descriptionIdStr)));  
 movie.setDirector(directorDAO.getDirectorById(Integer.parseInt(directorIdStr)));  
 dao.addMovie(movie);  
 return "/index?faces-redirect=true";  
 }  
  
 public String delete(Movie movie) {  
 dao.deleteMovie(movie);  
 return "/index?faces-redirect=true";  
 }  
  
 public String goToUpdate(Movie movie) {  
 yearStr = "" + movie.getYear();  
 durationStr = "" + movie.getDuration();  
 descriptionIdStr = "" + movie.getDescription().getDescriptionId();  
 directorIdStr = "" + movie.getDirector().getDirectorId();  
 this.movie = movie;  
 return "JSF/update/updateMovie";  
 }  
  
 public Movie getMovie() {  
 return movie;  
 }  
  
 public void setMovie(Movie movie) {  
 this.movie = movie;  
 }  
  
 public String getDescriptionIdStr() {  
 return descriptionIdStr;  
 }  
  
 public void setDescriptionIdStr(String descriptionIdStr) {  
 this.descriptionIdStr = descriptionIdStr;  
 }  
  
 public String getDirectorIdStr() {  
 return directorIdStr;  
 }  
  
 public void setDirectorIdStr(String directorIdStr) {  
 this.directorIdStr = directorIdStr;  
 }  
  
 public String getDurationStr() {  
 return durationStr;  
 }  
  
 public void setDurationStr(String durationStr) {  
 this.durationStr = durationStr;  
 }  
  
 public String getYearStr() {  
 return yearStr;  
 }  
  
 public void setYearStr(String yearStr) {  
 this.yearStr = yearStr;  
 }  
}

**MPController.java**

package managedBean;  
  
import dao.MovieDAO;  
import dao.ProducerDAO;  
import entities.Movie;  
import entities.Producer;  
import org.apache.log4j.Logger;  
  
import javax.ejb.EJB;  
import javax.faces.bean.ManagedBean;  
import javax.faces.bean.SessionScoped;  
  
*/\*\*  
 \* Created by win10 on 08.11.2016.  
 \*/*@ManagedBean(name = "MPController", eager = true)  
@SessionScoped  
public class MPController {  
 protected final static Logger *LOGGER* = Logger.*getLogger*(MPController.class);  
  
 @EJB  
 private MovieDAO movieDAO;  
  
 @EJB  
 private ProducerDAO producerDAO;  
  
 private String movieIdStr;  
 private String producerIdStr;  
  
 public String goToAdd() {  
 movieIdStr = producerIdStr = null;  
 return "JSF/add/addMP";  
 }  
  
 public String delete(Movie movie, Producer producer){  
 movie.removeProducer(producer);  
 movieDAO.update(movie);  
 return "/index?faces-redirect=true";  
 }  
  
 public String add() {  
 Movie movie = movieDAO.getById(Integer.*parseInt*(movieIdStr));  
 Producer producer = producerDAO.getById(Integer.*parseInt*(producerIdStr));  
 movie.addProducer(producer);  
 movieDAO.update(movie);  
 return "/index?faces-redirect=true";  
 }  
  
 public String getMovieIdStr() {  
 return movieIdStr;  
 }  
  
 public void setMovieIdStr(String movieIdStr) {  
 this.movieIdStr = movieIdStr;  
 }  
  
 public String getProducerIdStr() {  
 return producerIdStr;  
 }  
  
 public void setProducerIdStr(String producerIdStr) {  
 this.producerIdStr = producerIdStr;  
 }  
}

**ProducerController.java**

package managedBean;  
  
import org.apache.log4j.Logger;  
import wsClient.Producer;  
import wsClient.ProducerWS;  
import wsClient.ProducerWSImpImplementationService;  
  
import javax.annotation.PostConstruct;  
import javax.faces.bean.ManagedBean;  
import javax.faces.bean.SessionScoped;  
import java.util.List;  
  
*/\*\*  
 \* Created by win10 on 08.11.2016.  
 \*/*@ManagedBean(name = "producerController", eager = true)  
@SessionScoped  
public class ProducerController {  
  
 private final static Logger logger = Logger.getLogger(ProducerController.class);  
  
 private ProducerWS dao;  
  
 @PostConstruct  
 public void init() {  
 ProducerWSImpImplementationService producerWSImpImplementationService = new ProducerWSImpImplementationService();  
 dao = producerWSImpImplementationService.getProducerWSImpImplementationPort();  
 }  
  
 private Producer producer;  
  
 public List<Producer> getAll() {  
 return dao.getAllProducers();  
 }  
  
 public String update() {  
 dao.updateProducer(producer);  
 return "/index?faces-redirect=true";  
 }  
  
 public String goToAdd() {  
 producer = new Producer();  
 return "JSF/add/addProducer";  
 }  
  
 public String add() {  
 dao.addProducer(producer);  
 return "/index?faces-redirect=true";  
 }  
  
 public String delete(Producer producer) {  
 dao.deleteProducer(producer);  
 return "/index?faces-redirect=true";  
 }  
  
 public String goToUpdate(Producer producer) {  
 this.producer = producer;  
 return "JSF/update/updateProducer";  
 }  
  
 public Producer getProducer() {  
 return producer;  
 }  
  
 public void setProducer(Producer producer) {  
 this.producer = producer;  
 }  
  
}

**SequelController.java**

package managedBean;  
  
import dao.SearchDAO;  
import entities.Movie;  
  
import javax.ejb.EJB;  
import javax.faces.bean.ManagedBean;  
import javax.faces.bean.SessionScoped;  
  
*/\*\*  
 \* Created by win10 on 14.11.2016.  
 \*/*@ManagedBean(name = "sequelController", eager = true)  
@SessionScoped  
public class SequelController {  
  
 @EJB  
 private SearchDAO searchDAO;  
  
 public void searchDirector(Movie movie) {  
 searchDAO.searchDirector(movie);  
 }  
  
}

**DescriptionService.java**

package rsServices;  
  
import dao.DescriptionDAO;  
import entities.Description;  
  
import javax.ejb.EJB;  
import javax.ws.rs.\*;  
import javax.ws.rs.core.MediaType;  
import java.util.List;  
  
*/\*\*  
 \* Created by win10 on 21.12.2016.  
 \*/*@Path("/description")  
public class DescriptionService {  
  
 @EJB  
 private DescriptionDAO descriptionDAO;  
  
 @GET  
 @Produces(MediaType.*APPLICATION\_JSON*)  
 public List<Description> getDescriptions() {  
 return descriptionDAO.getAll();  
 }  
  
 @GET  
 @Path("/{id}")  
 @Produces(MediaType.*APPLICATION\_JSON*)  
 public Description getDescription(@PathParam("id") int id) {  
 return descriptionDAO.getById(id);  
 }  
  
 @POST  
 @Consumes(MediaType.*APPLICATION\_JSON*)  
 public void updateDescription(Description description) {  
 descriptionDAO.update(description);  
 }  
  
 @DELETE  
 @Path("/{id}")  
 @Produces(MediaType.*APPLICATION\_JSON*)  
 public void deleteDescription(@PathParam("id") int id) {  
 descriptionDAO.deleteById(id);  
 }  
  
}

**DirectorService.java**

package rsServices;  
  
import dao.DirectorDAO;  
import entities.Director;  
  
import javax.ejb.EJB;  
import javax.ws.rs.\*;  
import javax.ws.rs.core.MediaType;  
import java.util.List;  
  
*/\*\*  
 \* Created by win10 on 21.12.2016.  
 \*/*@Path("/director")  
public class DirectorService {  
  
 @EJB  
 private DirectorDAO directorDAO;  
  
 @GET  
 @Produces(MediaType.*APPLICATION\_JSON*)  
 public List<Director> getDirectors() {  
 return directorDAO.getAll();  
 }  
  
 @GET  
 @Path("/{id}")  
 @Produces(MediaType.*APPLICATION\_JSON*)  
 public Director getDirector(@PathParam("id") int id) {  
 return directorDAO.getById(id);  
 }  
  
 @POST  
 @Consumes(MediaType.*APPLICATION\_JSON*)  
 public void updateDirector(Director director) {  
 directorDAO.update(director);  
 }  
  
 @DELETE  
 @Path("/{id}")  
 @Produces(MediaType.*APPLICATION\_JSON*)  
 public void deleteDirector(@PathParam("id") int id) {  
 directorDAO.deleteById(id);  
 }  
  
}

**MovieService.java**

package rsServices;  
  
import dao.MovieDAO;  
import entities.Movie;  
  
import javax.ejb.EJB;  
import javax.ws.rs.\*;  
import javax.ws.rs.core.MediaType;  
import java.util.List;  
  
*/\*\*  
 \* Created by win10 on 21.12.2016.  
 \*/*@Path("/movie")  
public class MovieService {  
  
 @EJB  
 private MovieDAO movieDAO;  
  
 @GET  
 @Produces(MediaType.*APPLICATION\_JSON*)  
 public List<Movie> getMovies() {  
 return movieDAO.getAll();  
 }  
  
 @GET  
 @Path("/{id}")  
 @Produces(MediaType.*APPLICATION\_JSON*)  
 public Movie getMovie(@PathParam("id") int id) {  
 return movieDAO.getById(id);  
 }  
  
 @POST  
 @Consumes(MediaType.*APPLICATION\_JSON*)  
 public void updateMovie(Movie movie) {  
 movieDAO.update(movie);  
 }  
  
 @DELETE  
 @Path("/{id}")  
 @Produces(MediaType.*APPLICATION\_JSON*)  
 public void deleteMovie(@PathParam("id") int id) {  
 movieDAO.deleteById(id);  
 }  
  
}

**ProducerService.java**

package rsServices;  
  
import dao.ProducerDAO;  
import entities.Producer;  
  
import javax.ejb.EJB;  
import javax.ws.rs.\*;  
import javax.ws.rs.core.MediaType;  
import java.util.List;  
  
*/\*\*  
 \* Created by win10 on 21.12.2016.  
 \*/*@Path("/producer")  
public class ProducerService {  
  
 @EJB  
 private ProducerDAO producerDAO;  
  
 @GET  
 @Produces(MediaType.APPLICATION\_JSON)  
 public List<Producer> getProducers() {  
 return producerDAO.getAll();  
 }  
  
 @GET  
 @Path("/{id}")  
 @Produces(MediaType.APPLICATION\_JSON)  
 public Producer getProducer(@PathParam("id") int id) {  
 return producerDAO.getById(id);  
 }  
  
 @POST  
 @Consumes(MediaType.APPLICATION\_JSON)  
 public void updateProducer(Producer producer) {  
 producerDAO.update(producer);  
 }  
  
 @DELETE  
 @Path("/{id}")  
 @Produces(MediaType.APPLICATION\_JSON)  
 public void deleteProducer(@PathParam("id") int id) {  
 producerDAO.deleteById(id);  
 }  
  
}

**DescriptionWS.java**

package wsServices;  
  
import entities.Description;  
  
import javax.jws.WebMethod;  
import javax.jws.WebService;  
import java.util.List;  
  
  
@WebService  
public interface DescriptionWS {  
  
 @WebMethod  
 Description getDescriptionById(int id);  
  
 @WebMethod  
 List<Description> getAllDescriptions();  
  
 @WebMethod  
 void addDescription(Description description);  
  
 @WebMethod  
 void updateDescription(Description description);  
  
 @WebMethod  
 void deleteDescription(Description description);  
  
 @WebMethod  
 void deleteDescriptionById(int id);  
}

**DirectorWS.java**

package wsServices;  
  
import entities.Director;  
  
import javax.jws.WebMethod;  
import javax.jws.WebService;  
import java.util.List;  
  
*/\*\*  
 \* Created by win10 on 20.12.2016.  
 \*/*@WebService  
public interface DirectorWS {  
  
 @WebMethod  
 Director getDirectorById(int id);  
  
 @WebMethod  
 List<Director> getAllDirectors();  
  
 @WebMethod  
 void addDirector(Director director);  
  
 @WebMethod  
 void updateDirector(Director director);  
  
 @WebMethod  
 void deleteDirector(Director director);  
  
 @WebMethod  
 void deleteDirectorById(int id);  
  
}

**MovieWS.java**

package wsServices;  
  
import entities.Movie;  
  
import javax.jws.WebMethod;  
import javax.jws.WebService;  
import java.util.List;  
  
*/\*\*  
 \* Created by win10 on 20.12.2016.  
 \*/*@WebService  
public interface MovieWS {  
  
 @WebMethod  
 Movie getMovieById(int id);  
  
 @WebMethod  
 List<Movie> getAllMovies();  
  
 @WebMethod  
 void addMovie(Movie movie);  
  
 @WebMethod  
 void updateMovie(Movie movie);  
  
 @WebMethod  
 void deleteMovie(Movie movie);  
  
 @WebMethod  
 void deleteMovieById(int id);  
  
}

**ProducerWS.java**

package wsServices;  
  
import entities.Producer;  
  
import javax.jws.WebMethod;  
import javax.jws.WebService;  
import java.util.List;  
  
*/\*\*  
 \* Created by win10 on 20.12.2016.  
 \*/*@WebService  
public interface ProducerWS {  
  
 @WebMethod  
 Producer getProducerById(int id);  
  
 @WebMethod  
 List<Producer> getAllProducers();  
  
 @WebMethod  
 void addProducer(Producer producer);  
  
 @WebMethod  
 void updateProducer(Producer producer);  
  
 @WebMethod  
 void deleteProducer(Producer producer);  
  
 @WebMethod  
 void deleteProducerById(int id);  
  
}

**DescriptionWSImpImplementation.java**

package wsServices;  
  
import dao.DescriptionDAO;  
import entities.Description;  
  
import javax.ejb.EJB;  
import javax.ejb.Stateless;  
import javax.jws.WebService;  
import java.util.List;  
  
@Stateless  
@WebService(endpointInterface = "wsServices.DescriptionWS")  
public class DescriptionWSImpImplementation implements DescriptionWS {  
  
 @EJB private DescriptionDAO descriptionDAO;  
  
 @Override  
 public Description getDescriptionById(int id) {  
 return descriptionDAO.getById(id);  
 }  
  
 @Override  
 public List<Description> getAllDescriptions() {  
 return descriptionDAO.getAll();  
 }  
  
 @Override  
 public void addDescription(Description description) {  
 descriptionDAO.add(description);  
 }  
  
 @Override  
 public void updateDescription(Description description) {  
 descriptionDAO.update(description);  
 }  
  
 @Override  
 public void deleteDescription(Description description) {  
 descriptionDAO.delete(description);  
 }  
  
 @Override  
 public void deleteDescriptionById(int id) {  
 descriptionDAO.deleteById(id);  
 }  
}

**DirectorWSImpImplementation.java**

package wsServices;  
  
import dao.DirectorDAO;  
import entities.Director;  
  
import javax.ejb.EJB;  
import javax.ejb.Stateless;  
import javax.jws.WebService;  
import java.util.List;  
  
*/\*\*  
 \* Created by win10 on 20.12.2016.  
 \*/*@Stateless  
@WebService(endpointInterface = "wsServices.DirectorWS")  
public class DirectorWSImpImplementation implements DirectorWS{  
  
 @EJB  
 private DirectorDAO directorDAO;  
  
 @Override  
 public Director getDirectorById(int id) {  
 return directorDAO.getById(id);  
 }  
  
 @Override  
 public List<Director> getAllDirectors() {  
 return directorDAO.getAll();  
 }  
  
 @Override  
 public void addDirector(Director director) {  
 directorDAO.add(director);  
 }  
  
 @Override  
 public void updateDirector(Director director) {  
 directorDAO.update(director);  
 }  
  
 @Override  
 public void deleteDirector(Director director) {  
 directorDAO.delete(director);  
 }  
  
 @Override  
 public void deleteDirectorById(int id) {  
 directorDAO.deleteById(id);  
 }  
  
}

**MovieWSWSImpImplementation.java**

package wsServices;  
  
import dao.MovieDAO;  
import entities.Movie;  
  
import javax.ejb.EJB;  
import javax.ejb.Stateless;  
import javax.jws.WebService;  
import java.util.List;  
  
*/\*\*  
 \* Created by win10 on 20.12.2016.  
 \*/*@Stateless  
@WebService(endpointInterface = "wsServices.MovieWS")  
public class MovieWSWSImpImplementation implements MovieWS{  
  
 @EJB  
 private MovieDAO movieDAO;  
  
 @Override  
 public Movie getMovieById(int id) {  
 return movieDAO.getById(id);  
 }  
  
 @Override  
 public List<Movie> getAllMovies() {  
 return movieDAO.getAll();  
 }  
  
 @Override  
 public void addMovie(Movie movie) {  
 movieDAO.add(movie);  
 }  
  
 @Override  
 public void updateMovie(Movie movie) {  
 movieDAO.update(movie);  
 }  
  
 @Override  
 public void deleteMovie(Movie movie) {  
 movieDAO.delete(movie);  
 }  
  
 @Override  
 public void deleteMovieById(int id) {  
 movieDAO.deleteById(id);  
 }  
  
}

**ProducerWSImpImplementation.java**

package wsServices;  
  
import dao.ProducerDAO;  
import entities.Producer;  
  
import javax.ejb.EJB;  
import javax.ejb.Stateless;  
import javax.jws.WebService;  
import java.util.List;  
  
*/\*\*  
 \* Created by win10 on 20.12.2016.  
 \*/*@Stateless  
@WebService(endpointInterface = "wsServices.ProducerWS")  
public class ProducerWSImpImplementation implements ProducerWS{  
  
 @EJB  
 private ProducerDAO producerDAO;  
  
 @Override  
 public Producer getProducerById(int id) {  
 return producerDAO.getById(id);  
 }  
  
 @Override  
 public List<Producer> getAllProducers() {  
 return producerDAO.getAll();  
 }  
  
 @Override  
 public void addProducer(Producer producer) {  
 producerDAO.add(producer);  
 }  
  
 @Override  
 public void updateProducer(Producer producer) {  
 producerDAO.update(producer);  
 }  
  
 @Override  
 public void deleteProducer(Producer producer) {  
 producerDAO.delete(producer);  
 }  
  
 @Override  
 public void deleteProducerById(int id) {  
 producerDAO.deleteById(id);  
 }  
  
}

**DescriptionWSImpImplementationService.java**

package wsClient;  
  
import javax.xml.namespace.QName;  
import javax.xml.ws.\*;  
import java.net.MalformedURLException;  
import java.net.URL;  
  
  
*/\*\*  
 \* This class was generated by the JAX-WS RI.  
 \* JAX-WS RI 2.2.9-b130926.1035  
 \* Generated source version: 2.2  
 \*   
 \*/*@WebServiceClient(name = "DescriptionWSImpImplementationService", targetNamespace = "http://wsServices/", wsdlLocation = "http://localhost:8080/lab7\_war\_exploded/DescriptionWSImpImplementation?wsdl")  
public class DescriptionWSImpImplementationService  
 extends Service  
{  
  
 private final static URL *DESCRIPTIONWSIMPIMPLEMENTATIONSERVICE\_WSDL\_LOCATION*;  
 private final static WebServiceException *DESCRIPTIONWSIMPIMPLEMENTATIONSERVICE\_EXCEPTION*;  
 private final static QName *DESCRIPTIONWSIMPIMPLEMENTATIONSERVICE\_QNAME* = new QName("http://wsServices/", "DescriptionWSImpImplementationService");  
  
 static {  
 URL url = null;  
 WebServiceException e = null;  
 try {  
 url = new URL("http://localhost:8080/lab7\_war\_exploded/DescriptionWSImpImplementation?wsdl");  
 } catch (MalformedURLException ex) {  
 e = new WebServiceException(ex);  
 }  
 *DESCRIPTIONWSIMPIMPLEMENTATIONSERVICE\_WSDL\_LOCATION* = url;  
 *DESCRIPTIONWSIMPIMPLEMENTATIONSERVICE\_EXCEPTION* = e;  
 }  
  
 public DescriptionWSImpImplementationService() {  
 super(*\_\_getWsdlLocation*(), *DESCRIPTIONWSIMPIMPLEMENTATIONSERVICE\_QNAME*);  
 }  
  
 public DescriptionWSImpImplementationService(WebServiceFeature... features) {  
 super(*\_\_getWsdlLocation*(), *DESCRIPTIONWSIMPIMPLEMENTATIONSERVICE\_QNAME*, features);  
 }  
  
 public DescriptionWSImpImplementationService(URL wsdlLocation) {  
 super(wsdlLocation, *DESCRIPTIONWSIMPIMPLEMENTATIONSERVICE\_QNAME*);  
 }  
  
 public DescriptionWSImpImplementationService(URL wsdlLocation, WebServiceFeature... features) {  
 super(wsdlLocation, *DESCRIPTIONWSIMPIMPLEMENTATIONSERVICE\_QNAME*, features);  
 }  
  
 public DescriptionWSImpImplementationService(URL wsdlLocation, QName serviceName) {  
 super(wsdlLocation, serviceName);  
 }  
  
 public DescriptionWSImpImplementationService(URL wsdlLocation, QName serviceName, WebServiceFeature... features) {  
 super(wsdlLocation, serviceName, features);  
 }  
  
 */\*\*  
 \*   
 \** ***@return*** *\* returns DescriptionWS  
 \*/* @WebEndpoint(name = "DescriptionWSImpImplementationPort")  
 public DescriptionWS getDescriptionWSImpImplementationPort() {  
 return super.getPort(new QName("http://wsServices/", "DescriptionWSImpImplementationPort"), DescriptionWS.class);  
 }  
  
 */\*\*  
 \*   
 \** ***@param*** *features  
 \* A list of {****@link*** *javax.xml.ws.WebServiceFeature} to configure on the proxy. Supported features not in the <code>features</code> parameter will have their default values.  
 \** ***@return*** *\* returns DescriptionWS  
 \*/* @WebEndpoint(name = "DescriptionWSImpImplementationPort")  
 public DescriptionWS getDescriptionWSImpImplementationPort(WebServiceFeature... features) {  
 return super.getPort(new QName("http://wsServices/", "DescriptionWSImpImplementationPort"), DescriptionWS.class, features);  
 }  
  
 private static URL \_\_getWsdlLocation() {  
 if (*DESCRIPTIONWSIMPIMPLEMENTATIONSERVICE\_EXCEPTION*!= null) {  
 throw *DESCRIPTIONWSIMPIMPLEMENTATIONSERVICE\_EXCEPTION*;  
 }  
 return *DESCRIPTIONWSIMPIMPLEMENTATIONSERVICE\_WSDL\_LOCATION*;  
 }  
  
}

**DirectorWSImpImplementationService.java**

package wsClient;  
  
import javax.xml.namespace.QName;  
import javax.xml.ws.\*;  
import java.net.MalformedURLException;  
import java.net.URL;  
  
  
*/\*\*  
 \* This class was generated by the JAX-WS RI.  
 \* JAX-WS RI 2.2.9-b130926.1035  
 \* Generated source version: 2.2  
 \*   
 \*/*@WebServiceClient(name = "DirectorWSImpImplementationService", targetNamespace = "http://wsServices/", wsdlLocation = "http://localhost:8080/lab7\_war\_exploded/DirectorWSImpImplementation?wsdl")  
public class DirectorWSImpImplementationService  
 extends Service  
{  
  
 private final static URL *DIRECTORWSIMPIMPLEMENTATIONSERVICE\_WSDL\_LOCATION*;  
 private final static WebServiceException *DIRECTORWSIMPIMPLEMENTATIONSERVICE\_EXCEPTION*;  
 private final static QName *DIRECTORWSIMPIMPLEMENTATIONSERVICE\_QNAME* = new QName("http://wsServices/", "DirectorWSImpImplementationService");  
  
 static {  
 URL url = null;  
 WebServiceException e = null;  
 try {  
 url = new URL("http://localhost:8080/lab7\_war\_exploded/DirectorWSImpImplementation?wsdl");  
 } catch (MalformedURLException ex) {  
 e = new WebServiceException(ex);  
 }  
 *DIRECTORWSIMPIMPLEMENTATIONSERVICE\_WSDL\_LOCATION* = url;  
 *DIRECTORWSIMPIMPLEMENTATIONSERVICE\_EXCEPTION* = e;  
 }  
  
 public DirectorWSImpImplementationService() {  
 super(*\_\_getWsdlLocation*(), *DIRECTORWSIMPIMPLEMENTATIONSERVICE\_QNAME*);  
 }  
  
 public DirectorWSImpImplementationService(WebServiceFeature... features) {  
 super(*\_\_getWsdlLocation*(), *DIRECTORWSIMPIMPLEMENTATIONSERVICE\_QNAME*, features);  
 }  
  
 public DirectorWSImpImplementationService(URL wsdlLocation) {  
 super(wsdlLocation, *DIRECTORWSIMPIMPLEMENTATIONSERVICE\_QNAME*);  
 }  
  
 public DirectorWSImpImplementationService(URL wsdlLocation, WebServiceFeature... features) {  
 super(wsdlLocation, *DIRECTORWSIMPIMPLEMENTATIONSERVICE\_QNAME*, features);  
 }  
  
 public DirectorWSImpImplementationService(URL wsdlLocation, QName serviceName) {  
 super(wsdlLocation, serviceName);  
 }  
  
 public DirectorWSImpImplementationService(URL wsdlLocation, QName serviceName, WebServiceFeature... features) {  
 super(wsdlLocation, serviceName, features);  
 }  
  
 */\*\*  
 \*   
 \** ***@return*** *\* returns DirectorWS  
 \*/* @WebEndpoint(name = "DirectorWSImpImplementationPort")  
 public DirectorWS getDirectorWSImpImplementationPort() {  
 return super.getPort(new QName("http://wsServices/", "DirectorWSImpImplementationPort"), DirectorWS.class);  
 }  
  
 */\*\*  
 \*   
 \** ***@param*** *features  
 \* A list of {****@link*** *javax.xml.ws.WebServiceFeature} to configure on the proxy. Supported features not in the <code>features</code> parameter will have their default values.  
 \** ***@return*** *\* returns DirectorWS  
 \*/* @WebEndpoint(name = "DirectorWSImpImplementationPort")  
 public DirectorWS getDirectorWSImpImplementationPort(WebServiceFeature... features) {  
 return super.getPort(new QName("http://wsServices/", "DirectorWSImpImplementationPort"), DirectorWS.class, features);  
 }  
  
 private static URL \_\_getWsdlLocation() {  
 if (*DIRECTORWSIMPIMPLEMENTATIONSERVICE\_EXCEPTION*!= null) {  
 throw *DIRECTORWSIMPIMPLEMENTATIONSERVICE\_EXCEPTION*;  
 }  
 return *DIRECTORWSIMPIMPLEMENTATIONSERVICE\_WSDL\_LOCATION*;  
 }  
  
}

**MovieWSWSImpImplementationService.java**

package wsClient;  
  
import javax.xml.namespace.QName;  
import javax.xml.ws.\*;  
import java.net.MalformedURLException;  
import java.net.URL;  
  
  
*/\*\*  
 \* This class was generated by the JAX-WS RI.  
 \* JAX-WS RI 2.2.9-b130926.1035  
 \* Generated source version: 2.2  
 \*   
 \*/*@WebServiceClient(name = "MovieWSWSImpImplementationService", targetNamespace = "http://wsServices/", wsdlLocation = "http://localhost:8080/lab7\_war\_exploded/MovieWSWSImpImplementation?wsdl")  
public class MovieWSWSImpImplementationService  
 extends Service  
{  
  
 private final static URL *MOVIEWSWSIMPIMPLEMENTATIONSERVICE\_WSDL\_LOCATION*;  
 private final static WebServiceException *MOVIEWSWSIMPIMPLEMENTATIONSERVICE\_EXCEPTION*;  
 private final static QName *MOVIEWSWSIMPIMPLEMENTATIONSERVICE\_QNAME* = new QName("http://wsServices/", "MovieWSWSImpImplementationService");  
  
 static {  
 URL url = null;  
 WebServiceException e = null;  
 try {  
 url = new URL("http://localhost:8080/lab7\_war\_exploded/MovieWSWSImpImplementation?wsdl");  
 } catch (MalformedURLException ex) {  
 e = new WebServiceException(ex);  
 }  
 *MOVIEWSWSIMPIMPLEMENTATIONSERVICE\_WSDL\_LOCATION* = url;  
 *MOVIEWSWSIMPIMPLEMENTATIONSERVICE\_EXCEPTION* = e;  
 }  
  
 public MovieWSWSImpImplementationService() {  
 super(*\_\_getWsdlLocation*(), *MOVIEWSWSIMPIMPLEMENTATIONSERVICE\_QNAME*);  
 }  
  
 public MovieWSWSImpImplementationService(WebServiceFeature... features) {  
 super(*\_\_getWsdlLocation*(), *MOVIEWSWSIMPIMPLEMENTATIONSERVICE\_QNAME*, features);  
 }  
  
 public MovieWSWSImpImplementationService(URL wsdlLocation) {  
 super(wsdlLocation, *MOVIEWSWSIMPIMPLEMENTATIONSERVICE\_QNAME*);  
 }  
  
 public MovieWSWSImpImplementationService(URL wsdlLocation, WebServiceFeature... features) {  
 super(wsdlLocation, *MOVIEWSWSIMPIMPLEMENTATIONSERVICE\_QNAME*, features);  
 }  
  
 public MovieWSWSImpImplementationService(URL wsdlLocation, QName serviceName) {  
 super(wsdlLocation, serviceName);  
 }  
  
 public MovieWSWSImpImplementationService(URL wsdlLocation, QName serviceName, WebServiceFeature... features) {  
 super(wsdlLocation, serviceName, features);  
 }  
  
 */\*\*  
 \*   
 \** ***@return*** *\* returns MovieWS  
 \*/* @WebEndpoint(name = "MovieWSWSImpImplementationPort")  
 public MovieWS getMovieWSWSImpImplementationPort() {  
 return super.getPort(new QName("http://wsServices/", "MovieWSWSImpImplementationPort"), MovieWS.class);  
 }  
  
 */\*\*  
 \*   
 \** ***@param*** *features  
 \* A list of {****@link*** *javax.xml.ws.WebServiceFeature} to configure on the proxy. Supported features not in the <code>features</code> parameter will have their default values.  
 \** ***@return*** *\* returns MovieWS  
 \*/* @WebEndpoint(name = "MovieWSWSImpImplementationPort")  
 public MovieWS getMovieWSWSImpImplementationPort(WebServiceFeature... features) {  
 return super.getPort(new QName("http://wsServices/", "MovieWSWSImpImplementationPort"), MovieWS.class, features);  
 }  
  
 private static URL \_\_getWsdlLocation() {  
 if (*MOVIEWSWSIMPIMPLEMENTATIONSERVICE\_EXCEPTION*!= null) {  
 throw *MOVIEWSWSIMPIMPLEMENTATIONSERVICE\_EXCEPTION*;  
 }  
 return *MOVIEWSWSIMPIMPLEMENTATIONSERVICE\_WSDL\_LOCATION*;  
 }  
  
}

**ProducerWSImpImplementationService.java**

package wsClient;  
  
import javax.xml.namespace.QName;  
import javax.xml.ws.\*;  
import java.net.MalformedURLException;  
import java.net.URL;  
  
  
*/\*\*  
 \* This class was generated by the JAX-WS RI.  
 \* JAX-WS RI 2.2.9-b130926.1035  
 \* Generated source version: 2.2  
 \*   
 \*/*@WebServiceClient(name = "ProducerWSImpImplementationService", targetNamespace = "http://wsServices/", wsdlLocation = "http://localhost:8080/lab7\_war\_exploded/ProducerWSImpImplementation?wsdl")  
public class ProducerWSImpImplementationService  
 extends Service  
{  
  
 private final static URL *PRODUCERWSIMPIMPLEMENTATIONSERVICE\_WSDL\_LOCATION*;  
 private final static WebServiceException *PRODUCERWSIMPIMPLEMENTATIONSERVICE\_EXCEPTION*;  
 private final static QName *PRODUCERWSIMPIMPLEMENTATIONSERVICE\_QNAME* = new QName("http://wsServices/", "ProducerWSImpImplementationService");  
  
 static {  
 URL url = null;  
 WebServiceException e = null;  
 try {  
 url = new URL("http://localhost:8080/lab7\_war\_exploded/ProducerWSImpImplementation?wsdl");  
 } catch (MalformedURLException ex) {  
 e = new WebServiceException(ex);  
 }  
 *PRODUCERWSIMPIMPLEMENTATIONSERVICE\_WSDL\_LOCATION* = url;  
 *PRODUCERWSIMPIMPLEMENTATIONSERVICE\_EXCEPTION* = e;  
 }  
  
 public ProducerWSImpImplementationService() {  
 super(*\_\_getWsdlLocation*(), *PRODUCERWSIMPIMPLEMENTATIONSERVICE\_QNAME*);  
 }  
  
 public ProducerWSImpImplementationService(WebServiceFeature... features) {  
 super(*\_\_getWsdlLocation*(), *PRODUCERWSIMPIMPLEMENTATIONSERVICE\_QNAME*, features);  
 }  
  
 public ProducerWSImpImplementationService(URL wsdlLocation) {  
 super(wsdlLocation, *PRODUCERWSIMPIMPLEMENTATIONSERVICE\_QNAME*);  
 }  
  
 public ProducerWSImpImplementationService(URL wsdlLocation, WebServiceFeature... features) {  
 super(wsdlLocation, *PRODUCERWSIMPIMPLEMENTATIONSERVICE\_QNAME*, features);  
 }  
  
 public ProducerWSImpImplementationService(URL wsdlLocation, QName serviceName) {  
 super(wsdlLocation, serviceName);  
 }  
  
 public ProducerWSImpImplementationService(URL wsdlLocation, QName serviceName, WebServiceFeature... features) {  
 super(wsdlLocation, serviceName, features);  
 }  
  
 */\*\*  
 \*   
 \** ***@return*** *\* returns ProducerWS  
 \*/* @WebEndpoint(name = "ProducerWSImpImplementationPort")  
 public ProducerWS getProducerWSImpImplementationPort() {  
 return super.getPort(new QName("http://wsServices/", "ProducerWSImpImplementationPort"), ProducerWS.class);  
 }  
  
 */\*\*  
 \*   
 \** ***@param*** *features  
 \* A list of {****@link*** *javax.xml.ws.WebServiceFeature} to configure on the proxy. Supported features not in the <code>features</code> parameter will have their default values.  
 \** ***@return*** *\* returns ProducerWS  
 \*/* @WebEndpoint(name = "ProducerWSImpImplementationPort")  
 public ProducerWS getProducerWSImpImplementationPort(WebServiceFeature... features) {  
 return super.getPort(new QName("http://wsServices/", "ProducerWSImpImplementationPort"), ProducerWS.class, features);  
 }  
  
 private static URL \_\_getWsdlLocation() {  
 if (*PRODUCERWSIMPIMPLEMENTATIONSERVICE\_EXCEPTION*!= null) {  
 throw *PRODUCERWSIMPIMPLEMENTATIONSERVICE\_EXCEPTION*;  
 }  
 return *PRODUCERWSIMPIMPLEMENTATIONSERVICE\_WSDL\_LOCATION*;  
 }  
  
}

**persistence.xml**

<?xml version="1.0" encoding="UTF-8"?>  
<persistence xmlns="http://java.sun.com/xml/ns/persistence" version="2.0">  
  
 <persistence-unit name="movies" transaction-type="JTA">  
 <provider>org.hibernate.ejb.HibernatePersistence</provider>  
 <jta-data-source>java:jboss/datasources/MySQLDataSource</jta-data-source>  
  
 <class>entities.Movie</class>  
 <class>entities.Producer</class>  
 <class>entities.Description</class>  
 <class>entities.Director</class>  
  
 <exclude-unlisted-classes>false</exclude-unlisted-classes>  
  
 <properties>  
 <property name="hibernate.transaction.jta.platform"  
 value="org.hibernate.service.jta.platform.internal.JBossAppServerJtaPlatform"/>  
 <property name="hibernate.dialect" value="org.hibernate.dialect.MySQLDialect"/>  
 <property name="hibernate.hbm2ddl.auto" value="update"/>  
 </properties>  
 </persistence-unit>  
</persistence>

**log4j.properties**

log4j.rootLogger=INFO, file, stdout  
  
log4j.appender.file=org.apache.log4j.RollingFileAppender  
log4j.appender.file.File=E:\\EE\\lab7\\src\\main\\resources\\lab7.log  
log4j.appender.file.MaxFileSize=10MB  
log4j.appender.file.MaxBackupIndex=10  
log4j.appender.file.layout=org.apache.log4j.PatternLayout  
log4j.appender.file.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L - %m%n  
  
log4j.appender.stdout=org.apache.log4j.ConsoleAppender  
log4j.appender.stdout.Target=System.out  
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout  
log4j.appender.stdout.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L - %m%n

**web.xml**

<?xml version="1.0" encoding="UTF-8"?>  
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd"  
 version="3.1">  
  
 <servlet>  
 <servlet-name>Faces Servlet</servlet-name>  
 <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>  
 </servlet>  
  
 <welcome-file-list>  
 <welcome-file>/JSF/index.xhtml</welcome-file>  
 </welcome-file-list>  
  
  
 <servlet-mapping>  
 <servlet-name>Faces Servlet</servlet-name>  
 <url-pattern>\*.managedBean</url-pattern>  
 </servlet-mapping>  
  
 <servlet-mapping>  
 <servlet-name>Faces Servlet</servlet-name>  
 <url-pattern>\*.xhtml</url-pattern>  
 </servlet-mapping>  
  
 <servlet>  
 <servlet-name>jersey-serlvet</servlet-name>  
 <servlet-class>com.sun.jersey.spi.container.servlet.ServletContainer</servlet-class>  
 <init-param>  
 <param-name>com.sun.jersey.config.property.packages</param-name>  
 <param-value>rsServices</param-value>  
 </init-param>  
 <init-param>  
 <param-name>com.sun.jersey.api.json.POJOMappingFeature</param-name>  
 <param-value>true</param-value>  
 </init-param>  
  
 <load-on-startup>1</load-on-startup>  
 </servlet>  
 <servlet-mapping>  
 <servlet-name>jersey-serlvet</servlet-name>  
 <url-pattern>/rest/\*</url-pattern>  
 </servlet-mapping>  
  
</web-app>

**Висновок**

Програма вирішує поставлене завдання.

Програма працює правильно.