**WEB SERVICE**

**<Liste de films&service Movie Web>**

Préparé par

|  |  |
| --- | --- |
| Groupe#:15 | Projet:7 |
| Groupe Email:1478200283@qq.com | |
| Membres du Groupe: | |
| *<Etudiant1, ID ,Role>* | Léonard,1191024017,Membre |
| *<Etudiant2, ID, Role>* | Thibault,1191024033,Membre |
| *<Etudiant2, ID, Role>* | Quentin,1191024032,Membre |

Supervisé par

Dr. Fatiha&Dr. Ninghua

2022-2023 Semestre 1

**SOMMAIRE**

**[WEB SERVICE 1](#_Toc15030)**

**[1、Présentation du projet 1](#_Toc23648)**

**[1.1 Titre du projet 1](#_Toc9600)**

**[1.2 Exigences du projet 1](#_Toc9590)**

**[2、Ajoute des films 1](#_Toc8833)**

**[3、Ajoute des acteurs 2](#_Toc18382)**

**[4、Créez un client 3](#_Toc2314)**

**[5、Mise en œuvre de la fonction 3](#_Toc24220)**

**[6、Autres implémentations fonctionnelles 4](#_Toc19460)**

**[7、Annexe du code 5](#_Toc19361)**

**[7.1 Annexe I：movie.java 5](#_Toc32445)**

**[7.2 Annexe II：actor.java 11](#_Toc29498)**

**[7.3 Annexe III：HistoryEndpoint.java 12](#_Toc16706)**

**[7.4 Annexe IV：server.java 13](#_Toc5598)**

**[7.5 Annexe V：addclient.java 15](#_Toc22621)**

**[7.6 Annexe VI：DemoWSClient.java 16](#_Toc6809)**

**1、Présentation du projet**

**1.1 Titre du projet**

《Liste de films + service Movie Web》

**1.2 Exigences du projet**

● Créer un service qui

○ Ajoute des films

○ Ajoute des acteurs

○ Relie les films aux acteurs

○ Récupère les informations d'un titre de film,

○ Récupère les informations données à un nom d'acteur.

● Créez un client qui

○ Consomme les services susmentionnés

○ Interagit avec le service API OMDB (www.omdbapi.com) pour fournir le classement du film.

**2、Ajoute des films**

Nouvelle fonction nommée "movie". Cette fonction implémentera la fonction d'ajout de film. Un film contient des informations telles que" released, runtime, genre, actors, plot;, imdbRating, poster, title, id, year". Grâce à la fonction "movie", nous pouvons ajouter des informations sur le film.

|  |
| --- |
|  |

De plus, en créant la fonction "getHttpRequestData", nous implémenterons" Interagit avec le service API OMDB (www.omdbapi.com) pour fournir le classement du film. "

|  |
| --- |
|  |

**3、Ajoute des acteurs**

Nouvelle fonction nommée "actor". Les informations contenues dans un actor sont le nom, l'âge et le nom du film associé. Nous utilisons movieList pour associer l'acteur à son film correspondant.

Avec la fonction "actor", nous pouvons implémenter la fonction d'ajout d'acteur.

|  |
| --- |
|  |

**4、Créez un client**

Nouvelle fonction nommée "addclient", définissez le nom d'utilisateur, etc. et implémentez" Consommer les services susmentionnés".

|  |
| --- |
|  |

Nous prenons "star wars" comme exemple, et nous avons utilisé le film comme sujet de test pour le débogage et la démonstration du programme suivant.

|  |
| --- |
|  |

**5、Mise en œuvre de la fonction**

Nouvelle fonction nommée "DemoWSClient", effectuez la configuration du programme et recherchez toutes les informations sur les films et les acteurs. Grâce à la recherche, vous pouvez obtenir des informations pertinentes pour remplir les principales exigences de ce projet.

|  |
| --- |
|  |

Les résultats sont les suivants:

|  |
| --- |
| ae6b59918e8160bdce1052e3e92ab32 |

**6、Autres implémentations fonctionnelles**

Une nouvelle fonction, nommée "HistoryEndpoint", permet d'afficher certaines informations pendant la phase opérationnelle du projet, telles que:" Start Web Service Endpoint...", "Press any key to exit...", et la fonction affiche également l'emplacement du point de terminaison.

|  |
| --- |
|  |

**7、Annexe du code**

**7.1 Annexe I：movie.java**

|  |  |
| --- | --- |
| Nom du fichier | movie.java |
| package code;  import com.google.gson.Gson;  import com.google.gson.JsonIOException;  import com.google.gson.JsonSyntaxException;  import com.google.gson.reflect.TypeToken;  import org.apache.hc.client5.http.classic.HttpClient;  import org.apache.hc.client5.http.classic.methods.HttpGet;  import org.apache.hc.client5.http.impl.classic.CloseableHttpClient;  import org.apache.hc.client5.http.impl.classic.HttpClientBuilder;  import org.apache.hc.core5.http.HttpResponse;  import javax.xml.bind.annotation.XmlAccessType;  import javax.xml.bind.annotation.XmlAccessorType;  import javax.xml.bind.annotation.XmlRootElement;  import javax.xml.bind.annotation.XmlType;  import java.io.BufferedReader;  import java.io.IOException;  import java.io.InputStream;  import java.io.InputStreamReader;  import java.net.HttpURLConnection;  import java.net.URL;  import java.net.URLEncoder;  import java.util.ArrayList;  import java.util.HashMap;  import java.util.Map;  @XmlRootElement(name = "movie")  @XmlType(propOrder = { "title","year", "ActorList","id","released","runtime","genre","actors","plot","poster","imdbRating"}, namespace="abc")  @XmlAccessorType(XmlAccessType.FIELD)  public class movie {  private String released;  private String runtime;  private String genre;  private String actors;  private String plot;  private String imdbRating;  private String poster;  private String title;  private String id;  private String year;  public ArrayList<String> getGetActor() {  return ActorList;  }  public void setGetActor(ArrayList<String> getActor) {  this.ActorList = getActor;  }  public ArrayList<String> ActorList;  public movie(String title, String id) {  this.title = title;  this.id = id;  ActorList = new ArrayList<>();  }  public movie() {  }  ;  public movie(String title) {  this.title = title;  }  public String getTitle() {  return title;  }  public void setTitle(String title) {  this.title = title;  }  public String getId() {  return id;  }  public void setId(String id) {  this.id = id;  }  public String getYear() {  return year;  }  public void setYear(String year) {  this.year = year;  }  public movie(String released, String runtime, String genre, String actors, String plot, String imdbRating, String poster, String title, String id, String year) {  this.released = released;  this.runtime = runtime;  this.genre = genre;  this.actors = actors;  this.plot = plot;  this.imdbRating = imdbRating;  this.poster = poster;  this.title = title;  this.id = id;  this.year = year;  }  @Override  public String toString() {  return  ", title='" + title + '\'' +  ", id='" + id + '\'' +  "released='" + released + '\'' +  ", runtime='" + runtime + '\'' +  ", genre='" + genre + '\'' +  ", actors='" + actors + '\'' +  ", plot='" + plot + '\'' +  ", imdbRating='" + imdbRating + '\'' +  ", poster='" + poster + '\'' +  ", year='" + year + '\'' +  ", ActorList=" + ActorList  ;  }  public static movie serch(String data) {  String icarus = movie.getHttpRequestData(data);  Gson gson = new Gson();  HashMap<String, String> stringStringHashMap = new HashMap<>();  HashMap<String,String> map = gson.fromJson(icarus, stringStringHashMap.getClass());  String title = map.get("Title");  String year = map.get("Year");  String released = map.get("Released");  String runtime = map.get("Runtime");  String genre = map.get("Genre");  String actors = map.get("Actors");  String plot = map.get("Plot");  String imdbRating = map.get("imdbRating");  String poster = map.get("Poster");  String imdbID = map.get("imdbID");  movie movie = new movie(released, runtime, genre, actors, plot, imdbRating, poster, title, imdbID, year);  return movie;  }  public String getGenre() {  return genre;  }  public static String getHttpRequestData(String data) {  String returnString = "1";  try {  StringBuilder sb = new StringBuilder();  String urlP = "http://www.omdbapi.com/?apikey=dcbad6ad&t=";  sb.append(urlP);  data = URLEncoder.encode(data, "UTF-8");  sb.append(data);  URL url = new URL(sb.toString());  HttpURLConnection connect = (HttpURLConnection) url.openConnection();  connect.setRequestMethod("GET");  connect.connect();  InputStream isString = connect.getInputStream();  BufferedReader isRead = new BufferedReader(new InputStreamReader(isString));  String str = "";  while ((str = isRead.readLine()) != null) {  str = new String(str.getBytes(), "UTF-8");  returnString = str;  }  isString.close();  connect.disconnect();  } catch (Exception e) {  e.printStackTrace();  }  return returnString;  }  } | |

**7.2 Annexe II：actor.java**

|  |  |
| --- | --- |
| Nom du fichier | actor.java |
| package code;  import javax.xml.bind.annotation.XmlAccessorType;  import javax.xml.bind.annotation.XmlAccessType;  import javax.xml.bind.annotation.XmlRootElement;  import javax.xml.bind.annotation.XmlType;  import java.util.ArrayList;  @XmlRootElement(name = "actor")  @XmlType(propOrder = { "name","age", "movieList"}, namespace="abc")  @XmlAccessorType(XmlAccessType.FIELD)  public class actor {  name;  age;  public ArrayList<String> getMovieList() {  return movieList;  }  public void setMovieList(ArrayList<String> movieList) {  this.movieList = movieList;  }  public ArrayList<String> movieList;  public actor() {}  public actor(String name, String age) {  this.name = name;  this.age = age;  movieList=new ArrayList<>();  }  public actor(String name) {  this.name = name;  }  public String getName() {  return name;  }  @Override  public String toString() {  return  "name='" + name + '\'' +  ", age='" + age + '\''  ;  }  public void setName(String name) {  this.name = name;  }  } | |

**7.3 Annexe III：HistoryEndpoint.java**

|  |  |
| --- | --- |
| Nom du fichier | HistoryEndpoint.java |
| package code;  import netscape.javascript.JSObject;  import java.io.IOException;  import javax.xml.ws.Endpoint;  public class HistoryEndpoint {  public static void main(String[] args) throws IOException {  String url = "http://localhost:8080/custom/Service";  System.out.println("Starting Web Service Endpoint...");  Endpoint endpoint = Endpoint.publish(url, new server());  System.out.println("Endpoint available at " + url);  System.out.println("Press any key to exit...");  System.in.read();  endpoint.stop();  }  } | |

**7.4 Annexe IV：server.java**

|  |  |
| --- | --- |
| Nom du fichier | server.java |
| package code;  import java.util.\*;  import javax.jws.WebResult;  import javax.jws.WebService;  import javax.jws.WebMethod;  import javax.jws.WebParam;  @WebService(targetNamespace = "abc")  public class server {  HashMap<String, movie> movieMap = new HashMap<>();  HashMap<String, actor> actorMap = new HashMap<>();  public server() {  }  @WebMethod  @WebResult(name="movies")  public HashMap<String, movie> getAllmovie() {  return movieMap;  }  @WebMethod  @WebResult(name="actors")  public HashMap<String, actor> getAllactor() {  return actorMap;  }  @WebMethod  @WebResult(name="movies")  public HashMap<String, movie> addmovie(@WebParam(name = "id") String id, @WebParam(name = "title")String title) {  movie movie = new movie(title, id);  movieMap.put(title,movie);  return movieMap;  }  @WebMethod  @WebResult(name="actor")  public HashMap<String, actor> addactor(@WebParam(name = "name") String name, @WebParam(name = "age")String age) {  actor actor = new actor(name,age);  actorMap.put(name,actor);  return actorMap;  }  @WebMethod  @WebResult(name="connect")  public void connect(@WebParam(name = "name") String name, @WebParam(name = "title")String title) {  actor actor = actorMap.get(name);  movie movie = movieMap.get(title);  actor.movieList.add(title);  movie.ActorList.add(name);  actorMap.replace(name,actor);  movieMap.replace(title,movie);  }  } | |

**7.5 Annexe V：addclient.java**

|  |  |
| --- | --- |
| Nom du fichier | addclient.java |
| package code;  import javax.xml.namespace.QName;  import javax.xml.ws.Service;  import java.net.MalformedURLException;  import java.net.URL;  import java.util.HashMap;  public class addclient {  private static final String wsdlLocation = "http://localhost:8080/custom/Service?wsdl";  public static void main(String[] args) throws MalformedURLException {  URL location = new URL(wsdlLocation);  String namespace = "abc";  QName serviceName = new QName(namespace, "serverService");  Service service = Service.create(location, serviceName);  QName portName = new QName(namespace, "serverPort");  server proxy = service.getPort(portName, server.class);  // add movie  HashMap<String, movie> movie = proxy.addmovie("tt0076759", "star wars");  // add actor  HashMap<String, actor> actor = proxy.addactor("Mark Hamill", "20");  }  } | |

**7.6 Annexe VI：DemoWSClient.java**

|  |  |
| --- | --- |
| Nom du fichier | DemoWSClient.java |
| package code;  import java.net.MalformedURLException;  import java.net.URL;  import java.util.HashMap;  import java.util.List;  import javax.xml.namespace.QName;  import javax.xml.ws.Service;  public class DemoWSClient {  private static final String wsdlLocation = "http://localhost:8080/custom/Service?wsdl";  public static void main(String[] args) throws MalformedURLException {  URL location = new URL(wsdlLocation);  String namespace = "abc";  QName serviceName = new QName(namespace, "serverService");  Service service = Service.create(location, serviceName);  QName portName = new QName(namespace, "serverPort");  server proxy = service.getPort(portName, server.class);  // add movie  // HashMap<String, movie> movie = proxy.addmovie("tt0076759", "star wars");  // add actor  // HashMap<String, actor> actor = proxy.addactor("Mark Hamill", "20");  HashMap<String, actor> allactor = proxy.getAllactor();  HashMap<String, movie> allmovie = proxy.getAllmovie();  movie star = allmovie.get("star wars");  actor mark = allactor.get("Mark Hamill");  System.out.println("Search movie actor information:");  System.out.println("star wars' deatail:");  System.out.println(star);  System.out.println("Mark's detail:");  System.out.println(mark);  System.out.println("\n");  proxy.connect("Mark Hamill","star wars");  System.out.println("after connect:");  System.out.println("star wars' actorlist:");  System.out.println(star.ActorList);  System.out.println("Mark's movielist:");  System.out.println(mark.movieList);  System.out.println("\n");  System.out.println("star wars details from omdb:");  System.out.println(code.movie.serch("star wars"));  }} | |