НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Кафедра обчислювальної техніки

КУРСОВА РОБОТА

з дисципліни «Проектування програмного забезпечення»

на тему: «Сайт «Розклад руху поїздів». Модель»

Студента 2 курсу групи ІО-31

Напряму підготовки

6.050102 «Комп'ютерна інженерія»

Жижка Дмитра Сергійовича

Керівник

Болдак Андрій Олександрович

Національна шкала\_\_\_\_\_\_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис керівника)

Київ – 2015 р.

Оглавление

[РОЗДІЛ 1 4](#_Toc417250800)

[1.1 Введення 4](#_Toc417250801)

[1.1.1 Мета 4](#_Toc417250802)

[1.2 Короткий огляд продукту 4](#_Toc417250803)

[1.3 Детальний огляд продукту 5](#_Toc417250804)

[1.3.1 Призначення моделі 5](#_Toc417250805)

[1.3.2 Принцип використання 5](#_Toc417250806)

[1.3.3 Сценарій доступу до бази 6](#_Toc417250807)

[1.3.4 Сценарій оновлення бази 6](#_Toc417250808)

[1.4 Функціональність 6](#_Toc417250809)

[1.5 Практичність 6](#_Toc417250810)

[1.6 Надійність 7](#_Toc417250811)

[РОЗДІЛ 2 8](#_Toc417250812)

[2.1 Клас «Parser» 9](#_Toc417250813)

[2.2 Клас «Database» 9](#_Toc417250814)

[2.3 Клас «RestApplication» 10](#_Toc417250815)

[2.4 Клас «Search» 10](#_Toc417250816)

[ДОДАТКИ 12](#_Toc417250817)

[ДОДАТОК А 12](#_Toc417250818)

[ДОДАТОК Б 13](#_Toc417250819)

[ДОДАТОК В 54](#_Toc417250820)

# РОЗДІЛ 1

ЗАПИТИ ЗАЦІКАВЛЕНИХ ОСІБ

## 1.1 Введення

В цій роботі описується база даних, сервіс, який надає можливість користуватися її ресурсами, а також парсер, що заповнює базу.

### ***1.1.1 Мета***

Метою документа є визначення функціональності підсистеми та інтерфейсу взаємодіїї з базою.

## 1.2 Короткий огляд продукту

Модель складається з трьох основних частин: серверу MySQL, на якому знаходиться база даних, RESTful сервісу та парсера. База даних містить таблиці з розкладом стацій, маршрутів та їх вузлів. Сервіс забезпечує можливість доступу до бази за допомогою http протокола. Це дає можливість стороннім розробникам створювати на базі цієї моделі власні проекти. Для покращення взаємодіїї клієнта з сервісом було створено сукупність доступних запитів, які дозволяють передавити як окремі об’єкти, так і їх сукупності у вигляді списків. Парсер призначений для періодичного оновлення вмісту бази. Для цього він використовє сайт http://uz.gov.ua.

## 1.3 Детальний огляд продукту

### 1.3.1 **Призначення моделі**

Головна мета створення моделі – це відокремлення даних від реалізаціїї інтерфейсу та бізнес-логіки. Ці відношення описуються за допомогою патерна MVC. Цей шаблон поділяє систему на три частини: модель даних, вигляд даних та керування. Застосовується для відокремлення даних (модель) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача. Мета шаблону — гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми. Крім того використання цього шаблону у великих системах призводить до певної впорядкованості їх структури і робить їх зрозумілішими завдяки зменшенню складності.

### 1.3.2 **Принцип використання**

Спочатку треба створити базу даних на сервері MySQL. До неї потрібно додати 6 таблиць: таблиці Route, Station, Waypoint та три такі ж з додатком \_temp. Вони необхідні для прискорення оновлення бази. Спочатку парсер записує дані в тимчасові таблиці, а потім вони копіюються у постійні. Це дає можливість працювати з базою під час оновлення.

Далі потрібно інсталювати сервер-додатків, наприклад GlassFish або Tomcat. На ньому потрібно задиплоїти додаток timetable, що являє содою сервіс доступу до бази. Приклад запросу: http:/localhost/timetable/road/search/station\_by\_name/”Name”. Результат буде у вагладі списку станцій, які починаються та “Name”. Також на сервері ботрібно створити datasource з ім’ям jdbs/RoadSource.

### **1.3.3 Сценарій доступу до бази**

1. Користувач формує потрібний запит.
2. У заголовку вказує формат результату(xml/json).
3. Користувач надсилає запит.
4. Сервер відсилає результат.

### **1.3.4 Сценарій оновлення бази**

1. Запуск парсера.
2. Парсер підключається до сайту укрзалізниці.
3. Парсер формує нову базу у вигляді об’єктів в java.
4. Парсер записує об’екти у тимчасові таблиці.
5. Сервер MySQL видаляє усі дані з основних таблиць.
6. Сервер MySQL копіює дані з тимчасових до основних таблиць.

## 1.4 Функціональність

Доступні такі функціїї:

* Відсилання запросів.
* Оновлення бази.
* Можливість вибрати формат резултуючих даних(xml/json, спиок або один об’єкт).

## 1.5 Практичність

Оскільки модель реалізована у вигляді RestFull сервіса, то доступ до даних можуть отримувати не тільки контролер та view, а й сторонні розробники. Також це дає можливість розробити різні клієнти для різних платформ (сайт, програма, додаток для android). Для автоматичного оновлення можна використовувати можливості операційної системи (планувальник завдань) або MySQL (треба створити UDF).

## 1.6 Надійність

Парсер працює в автономному режимі, тому помилки можуть виникнути тільки при зміні структури сайту укрзалізниці. Якщо користувач відправить неправильний запит, то сервер проінформує його у відповідному вікні, де буде вказано номер помилки. Якщо на запит з відповідним параметром у базі немає потрібних об’єктів, то буде надіслано порожній список. Оскільки змінювати базу може тільки парсер, то внесення неправильних даних неможливе.

# РОЗДІЛ 2

РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

При розробці програмного продукту була використана мова програмування Java та програмне середовище IntelliJ IDEA 14.1. Даний продукт є сервісом, основна задача якого полягає в зберіганні та організації доступу до розкладу поїздів на території України.

## 2.1 Клас «Parser»

package parser;

public class Parser {

основний клас парсера

public ArrayList<Station> getStations() {}

повертає список станцій

public ArrayList<Route> getRoutes() {}

повертає список маршрутів

public void parse(String name) {}

запускає парсер

public ArrayList<WayPoint> getWayPoints() {}

повертає список точок маршрутів

}

## 2.2 Клас «Database»

public class Database {

забезпечує доступ до бази даних

public static Integer save(Object obj)

зберігає об’єкт

public static void saveDataBase(ArrayList<Station> stations, ArrayList<Route> routes, ArrayList<WayPoint> waypoints)

оновлює базу

public static Object get(Class clas , Integer id)

повертає обєкт за номером

public static List getRoute(String name)

повертає список маршрутів

public static List getStation(String name)

повертає список станцій

public static List getWayPointByStation(Integer id)

повертає список точок маршрутів

## 2.3 Клас «RestApplication»

@ApplicationPath("road")

public class RestApplication extends Application

основний клас вебсервісу

## 2.4 Клас «Search»

@Path("/search")

public class Search {

відповідає на запити

@Produces({"application/json", "application/xml"})

@Path("stations\_by\_name/{symbol}")

@GET

public StationList getStation(@PathParam("symbol") String symb)

@Produces({"application/json", "application/xml"})

@Path("station\_by\_id/{symbol}")

@GET

public Station getStationById(@PathParam("symbol") String symb)

@Produces({"application/json", "application/xml"})

@Path("route\_by\_id/{symbol}")

@GET

public Route getRouteById(@PathParam("symbol") String symb)

@Produces({"application/json", "application/xml"})

@Path("routes\_by\_train/{symbol}")

@GET

public RouteList getRouteByTrain(@PathParam("symbol") String symb)

@Produces({"application/json", "application/xml"})

@Path("routes\_by\_name/{symbol}")

@GET

public RouteList getRouteByName(@PathParam("symbol") String symb)

@Produces({"application/json", "application/xml"})

@Path("waypoint\_by\_id/{symbol}")

@GET

public WayPoint getWayPointById(@PathParam("symbol") String symb)

@Produces({"application/json", "application/xml"})

@Path("waypoints\_by\_route\_id/{symbol}")

@GET

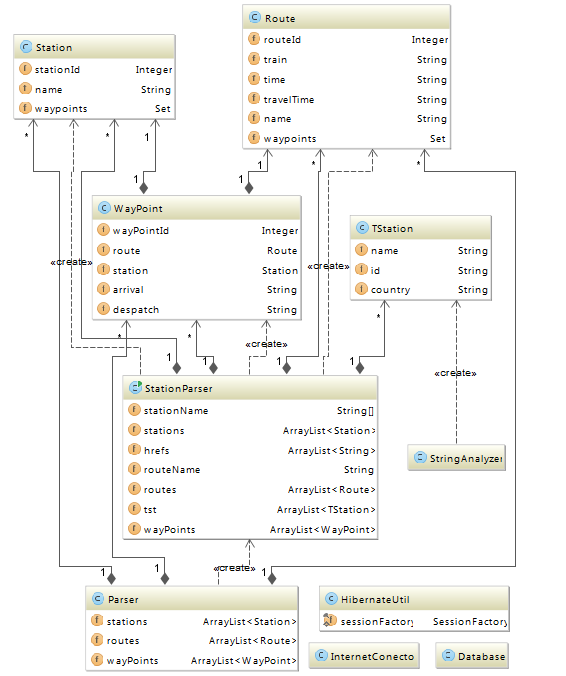
public WayPointList getWayPointByRouteId(@PathParam("symbol") String symb

}

# ДОДАТКИ

# ДОДАТОК А

Діаграма класів



# ДОДАТОК Б

package parser;

import orm.Route;

import orm.Station;

import orm.WayPoint;

import java.util.ArrayList;

public class Parser {

private ArrayList<Station> stations;

private ArrayList<Route> routes;

private ArrayList<WayPoint> wayPoints;

public ArrayList<Station> getStations() {

return stations;

}

public void setStations(ArrayList<Station> stations) {

this.stations = stations;

}

public ArrayList<Route> getRoutes() {

return routes;

}

public void setRoutes(ArrayList<Route> routes) {

this.routes = routes;

}

public void parse(String name) {

StationParser parser = new StationParser();

try {

parser.parse(name);

parser.createObjects();

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

stations = parser.getStations\_in\_list();

routes = parser.getRoutes\_in\_list();

wayPoints = parser.getWayPoints\_in\_list();

}

public ArrayList<WayPoint> getWayPoints() {

return wayPoints;

}

public void setWayPoints(ArrayList<WayPoint> wayPoints) {

this.wayPoints = wayPoints;

}

}

package ORMroad;

import org.hibernate.Hibernate;

import org.hibernate.Session;

import org.hibernate.Transaction;

import org.hibernate.criterion.Restrictions;

import java.util.\*;

public class Database {

public static Integer save(Object obj) {

Session ses = HibernateUtil.getSessionFactory().openSession();

ses.beginTransaction();

Integer id = (Integer) ses.save(obj);

ses.getTransaction().commit();

ses.close();

return id;

}

public static void saveDataBase(ArrayList<Station> stations, ArrayList<Route> routes,

ArrayList<WayPoint> waypoints) {

deleteDataBase();

Iterator<Route> it1 = routes.iterator();

while(it1.hasNext()) {

Route route = it1.next();

save(route);

}

Iterator<Station> it2 = stations.iterator();

while(it2.hasNext()) {

Station station = it2.next();

save(station);

}

Iterator<WayPoint> it3 = waypoints.iterator();

while(it3.hasNext()) {

WayPoint way = it3.next();

save(way);

}

}

public static void deleteDataBase() {

Session session = HibernateUtil.getSessionFactory().openSession();

Transaction tx = session.beginTransaction();

String hqlDelete = "delete WayPoint";

session.createQuery(hqlDelete).executeUpdate();

hqlDelete = "delete Route";

session.createQuery(hqlDelete).executeUpdate();

hqlDelete = "delete Station";

session.createQuery(hqlDelete).executeUpdate();

tx.commit();

session.close();

}

public static Object get(Class clas , Integer id) {

Session session = HibernateUtil.getSessionFactory().getCurrentSession();

Transaction tx = session.beginTransaction();

Object result = session.get(clas, id);

tx.commit();

//session.close();

return result;

}

public static Station initializeStation(Station entity) {

Session session = HibernateUtil.getSessionFactory().openSession();

entity = (Station) session.merge(entity);

Set<WayPoint> ways = entity.getWaypoints();

Iterator<WayPoint> it = ways.iterator();

Set<WayPoint> newWays = new HashSet<WayPoint>();

while(it.hasNext()) {

WayPoint point = it.next();

Hibernate.initialize(point);

newWays.add(point);

}

session.close();

entity.setWaypoints(newWays);

return entity;

}

public static Route initializeRoute(Route entity) {

Session session = HibernateUtil.getSessionFactory().openSession();

entity = (Route) session.merge(entity);

Set<WayPoint> ways = entity.getWaypoints();

Iterator<WayPoint> it = ways.iterator();

Set<WayPoint> newWays = new HashSet<WayPoint>();

while(it.hasNext()) {

WayPoint point = it.next();

Hibernate.initialize(point);

newWays.add(point);

}

session.close();

entity.setWaypoints(newWays);

return entity;

}

public static WayPoint initializeWayPoint(WayPoint entity) {

Session session = HibernateUtil.getSessionFactory().openSession();

entity = (WayPoint) session.merge(entity);

Station station = entity.getStation();

Hibernate.initialize(station);

station = (Station) session.merge(station);

//Hibernate.initialize(station);

//station.getName();

Route route = entity.getRoute();

Hibernate.initialize(route);

route = (Route) session.merge(route);

entity.setStation(station);

entity.setRoute(route);

session.close();

return entity;

}

public static List getStation(String name) {

Session session = HibernateUtil.getSessionFactory().openSession();

try {

return session.createCriteria(Station.class).add(

Restrictions.like("name", name + "%")).list();

} finally { session.close();}

}

public static List getRoute(String name) {

Session session = HibernateUtil.getSessionFactory().openSession();

try {

return session.createCriteria(Route.class).add(

Restrictions.like("name", name + "%")).list();

} finally { session.close();}

}

public static List getRouteByTrain(String train) {

Session session = HibernateUtil.getSessionFactory().openSession();

try {

return session.createCriteria(Route.class).add(

Restrictions.like("train", train)).list();

} finally { session.close();}

}

public static List getWayPointByStation(Integer id) {

Session session = HibernateUtil.getSessionFactory().openSession();

try {

Station station = (Station) Database.get(Station.class, id);

return session.createCriteria(WayPoint.class).add(

Restrictions.like("station", station)).list();

} finally { session.close();}

}

public static List getWayPointByRoute(Integer id) {

Session session = HibernateUtil.getSessionFactory().openSession();

try {

Route route = (Route) Database.get(Route.class, id);

return session.createCriteria(WayPoint.class).add(

Restrictions.like("route", route)).list();

} finally { session.close();}

}

}

import ORMroad.Database;

import ORMroad.Route;

import ORMroad.Station;

import ORMroad.WayPoint;

import collection.RouteList;

import collection.StationList;

import collection.WayPointList;

import javax.ws.rs.GET;

import javax.ws.rs.Path;

import javax.ws.rs.PathParam;

import javax.ws.rs.Produces;

import java.util.ArrayList;

import java.util.Iterator;

import java.util.List;

/\*\*

\* Created by Admin on 25.03.2015.

\*/

@Path("/search")

public class Search {

@Produces({"application/json", "application/xml"})

@Path("stations\_by\_name/{symbol}")

@GET

public StationList getStation(@PathParam("symbol") String symb) {

List<Station> list = Database.getStation(symb);

Iterator<Station> it = list.iterator();

ArrayList<Station> result = new ArrayList<Station>();

while(it.hasNext()) {

Station st = it.next();

st.setWaypoints(null);

result.add(st);

}

StationList stl = new StationList();

stl.setStations(result);

return stl;

}

@Produces({"application/json", "application/xml"})

@Path("station\_by\_id/{symbol}")

@GET

public Station getStationById(@PathParam("symbol") String symb) {

Integer id = new Integer(symb);

Station station = (Station) Database.get(Station.class, id);

station.setWaypoints(null);

return station;

}

@Produces({"application/json", "application/xml"})

@Path("route\_by\_id/{symbol}")

@GET

public Route getRouteById(@PathParam("symbol") String symb) {

Integer id = new Integer(symb);

Route route = (Route) Database.get(Route.class, id);

route.setWaypoints(null);

route.splitTrainName();

return route;

}

@Produces({"application/json", "application/xml"})

@Path("routes\_by\_train/{symbol}")

@GET

public RouteList getRouteByTrain(@PathParam("symbol") String symb) {

List<Route> routes = Database.getRouteByTrain(symb);

Iterator<Route> it = routes.iterator();

while (it.hasNext()) {

Route route = it.next();

route.splitTrainName();

route.setWaypoints(null);

}

RouteList list = new RouteList();

list.setRoutes(routes);

return list;

}

@Produces({"application/json", "application/xml"})

@Path("routes\_by\_name/{symbol}")

@GET

public RouteList getRouteByName(@PathParam("symbol") String symb) {

List<Route> routes = Database.getRoute(symb);

Iterator<Route> it = routes.iterator();

while (it.hasNext()) {

Route route = it.next();

route.splitTrainName();

route.setWaypoints(null);

}

RouteList list = new RouteList();

list.setRoutes(routes);

return list;

}

@Produces({"application/json", "application/xml"})

@Path("waypoint\_by\_id/{symbol}")

@GET

public WayPoint getWayPointById(@PathParam("symbol") String symb) {

Integer id = new Integer(symb);

WayPoint point = (WayPoint) Database.get(WayPoint.class, id);

point = Database.initializeWayPoint(point);

Station station = point.getStation();

station.setWaypoints(null);

Route route = point.getRoute();

route.setWaypoints(null);

point.setStation(station);

point.setRoute(route);

return point;

}

@Produces({"application/json", "application/xml"})

@Path("waypoints\_by\_station\_id/{symbol}")

@GET

public WayPointList getWayPointByStationId(@PathParam("symbol") String symb) {

Integer id = new Integer(symb);

List<WayPoint> list = Database.getWayPointByStation(id);

Iterator<WayPoint> it = list.iterator();

ArrayList<WayPoint> result = new ArrayList<WayPoint>();

while (it.hasNext()) {

WayPoint point = it.next();

point = Database.initializeWayPoint(point);

Station station = point.getStation();

station.setWaypoints(null);

Route route = point.getRoute();

route.setWaypoints(null);

point.setStation(station);

point.setRoute(route);

result.add(point);

}

WayPointList re = new WayPointList();

re.setWayPoints(result);

return re;

}

@Produces({"application/json", "application/xml"})

@Path("waypoints\_by\_route\_id/{symbol}")

@GET

public WayPointList getWayPointByRouteId(@PathParam("symbol") String symb) {

Integer id = new Integer(symb);

List<WayPoint> list = Database.getWayPointByRoute(id);

Iterator<WayPoint> it = list.iterator();

ArrayList<WayPoint> result = new ArrayList<WayPoint>();

while (it.hasNext()) {

WayPoint point = it.next();

point = Database.initializeWayPoint(point);

Station station = point.getStation();

station.setWaypoints(null);

Route route = point.getRoute();

route.setWaypoints(null);

point.setStation(station);

point.setRoute(route);

result.add(point);

}

WayPointList re = new WayPointList();

re.setWayPoints(result);

return re;

}

}

import javax.ws.rs.ApplicationPath;

import javax.ws.rs.core.Application;

/\*\*

\* Created by Admin on 25.03.2015.

\*/

@ApplicationPath("road")

public class RestApplication extends Application {

}

package Parser;

import java.io.IOException;

import java.io.InputStream;

import java.io.InputStreamReader;

import java.io.OutputStreamWriter;

import java.net.MalformedURLException;

import java.net.URL;

import java.net.URLConnection;

import java.util.ArrayList;

import java.util.Iterator;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import ORMroad.Route;

import ORMroad.Station;

import ORMroad.WayPoint;

public class StationParser {

private String[] stationName;

private ArrayList<Station> stations;

private ArrayList<String> hrefs;

private String routeName;

private ArrayList<Route> routes;

private ArrayList<TStation> tst;

private ArrayList<WayPoint> wayPoints;

public ArrayList<WayPoint> getWayPoints\_in\_list() {

return wayPoints;

}

public ArrayList<Station> getStations\_in\_list() {

return stations;

}

public ArrayList<Route> getRoutes\_in\_list() {

return routes;

}

public StationParser() {

hrefs = new ArrayList<String>();

stationName = new String[2];

stationName[0] = "http://uz.gov.ua/passengers/timetables/?station=";

stationName[1] = "&by\_station=РџРѕС€СѓРє";

stations = new ArrayList<Station>();

routeName = "http://uz.gov.ua/passengers/timetables/";

routes = new ArrayList<Route>();

wayPoints = new ArrayList<WayPoint>();

}

public void parse(String name) throws Exception {

String spat1 = "<table>.\*</tbody></table>";

Pattern pat1 = Pattern.compile(spat1, Pattern.DOTALL);

String spat2 = "<tbody>.\*</tbody>";

Pattern pat2 = Pattern.compile(spat2, Pattern.DOTALL);

String spat3 = "<tr>.\*?</tr>";

Pattern pat3 = Pattern.compile(spat3, Pattern.DOTALL);

String spat4 = "<td>.\*?</td>";

Pattern pat4 = Pattern.compile(spat4, Pattern.DOTALL);

String spat5 = "\"[^.]+?\"";

Pattern pat5 = Pattern.compile(spat5, Pattern.DOTALL);

tst = InternetConector.viewStation(name);

//System.out.println("tst=" + tst.size());

Iterator<TStation> it = tst.iterator();

//int n = 0;

while(it.hasNext()) {

//System.out.println(++n);

TStation tstation = it.next();

//Station station = new Station();

//station.setName(tstation.getName());

//stations.add(station);

String s = InternetConector.conection(this.getStationURL(tstation.getId()));

Matcher mat = pat1.matcher(s);

//System.out.println(s);

if(mat.find()) {

int start = mat.start();

int end = mat.end();

String tab2 = s.substring(start, end);

//System.out.println(tab2);

mat = pat2.matcher(tab2);

mat.find();

start = mat.start();

end = mat.end();

tab2 = tab2.substring(start, end);

//System.out.println(tab2);

mat = pat3.matcher(tab2);

//mat.find();

while(mat.find()) {

String tab3 = tab2.substring(mat.start(), mat.end());

//System.out.println(tab3);

Matcher mat4 = pat4.matcher(tab3);

mat4.find();

Matcher mat5 = pat5.matcher(tab3);

mat5.find();

String href = tab3.substring(mat5.start() + 1, mat5.end() - 1);

boolean isConteinsH = false;

Iterator<String> iter = hrefs.iterator();

while(iter.hasNext() && !isConteinsH) {

String temp = iter.next();

isConteinsH = temp.equals(href);

}

if(!isConteinsH) hrefs.add(href);

//System.out.println(href);

//Route route = new Route();

}

}

}

//System.out.println("hrefs=" + hrefs.size());

//Iterator<String> ite = hrefs.iterator();

//while(ite.hasNext()) {

//System.out.println(ite.next());

//}

}

public void createObjects() throws Exception {

String spat1 = "<table>.\*</tbody></table>";

Pattern pat1 = Pattern.compile(spat1, Pattern.DOTALL);

String spat2 = "<tbody>.\*</tbody>";

Pattern pat2 = Pattern.compile(spat2, Pattern.DOTALL);

String spat3 = "<tr>[^.]+?</tr>";

Pattern pat3 = Pattern.compile(spat3, Pattern.DOTALL);

String spat4 = "<td>.\*?</td>";

Pattern pat4 = Pattern.compile(spat4, Pattern.DOTALL);

String spat5 = "\"[^.]+?\"";

Pattern pat5 = Pattern.compile(spat5, Pattern.DOTALL);

String spat6 = "<tbody>[^.]+?</tbody>";

Pattern pat6 = Pattern.compile(spat6, Pattern.DOTALL);

String spat7 = ";.\*&";

Pattern pat7 = Pattern.compile(spat7);

Iterator<String> it = hrefs.iterator();

//int n = 0;

while(it.hasNext()) {

//System.out.println(++n);

String s = InternetConector.conection(

routeName + it.next());

//String s = InternetConector.conection(

// routeName + "?ntrain=52519&by\_id=1");

//System.out.println(s);

Matcher mat1 = pat1.matcher(s);

mat1.find();

//mat7.find();

String tab = s.substring(mat1.start(), mat1.end());

//System.out.println(tab);

Matcher mat2 = pat2.matcher(tab);

mat2.find();

tab = tab.substring(mat2.start(), mat2.end());

//System.out.println(tab);

Matcher mat4 = pat4.matcher(tab);

mat4.find();

String name = tab.substring(mat4.start() + 4, mat4.end() - 5).trim();

//System.out.println(name);

mat4.find();

String train = tab.substring(mat4.start() + 4, mat4.end() - 5).trim();

//System.out.println(train);

mat4.find();

String time = tab.substring(mat4.start() + 4, mat4.end() - 5).trim();

//System.out.println(time);

mat4.find();

String travelTime = tab.substring(mat4.start() + 4, mat4.end() - 5).trim();

//System.out.println(travelTime);

Route route = new Route(train, time, travelTime, name);

routes.add(route);

while(mat4.find()) {

String st\_name = tab.substring(mat4.start(), mat4.end());

Matcher mat7 = pat7.matcher(st\_name);

mat7.find();

st\_name = st\_name.substring(mat7.start() + 1, mat7.end() - 1).trim();

//System.out.println(st\_name);

mat4.find();

String arrival = tab.substring(mat4.start() + 4, mat4.end() - 5).trim();

//System.out.println(arrival);

mat4.find();

String despatch = tab.substring(mat4.start() + 4, mat4.end() - 5).trim();

//System.out.println(despatch);

Iterator<Station> it\_st = stations.iterator();

Station add = null;

while(add == null && it\_st.hasNext()) {

Station st\_temp = it\_st.next();

String name\_temp = st\_temp.getName();

if(name\_temp.equals(st\_name)) add = st\_temp;

}

if(add == null) {

add = new Station(st\_name);

stations.add(add);

}

WayPoint way = new WayPoint(route, add, arrival, despatch);

//route.addWayPoint(way);

way.setRoute(route);

way.setStation(add);

wayPoints.add(way);

//add.addRoute(route);

}

//String temp = tab.substring(mat4.start(), mat4.end());

//System.out.println(temp);

}

//System.out.println("st=" + stations.size());

//System.out.println("route=" + routes.size());

//Iterator<Route> itera = routes.iterator();

//while(itera.hasNext()) {

//Route r = itera.next();

//System.out.println(r.getName() + " " + r.getTrain());

//}

}

private String getStationURL(String id) throws MalformedURLException {

return stationName[0] + id + stationName[1];

}

public static void main(String[] args) throws Exception {

StationParser parser = new StationParser();

//parser.parse("Р’Р°СЂС€Р°РІР°");

//parser.createObjects();

//parser.less();

//parser.getHrefs();

//parser.getStations();

QueryString q = new QueryString()

.add("station\_id\_from","2200001")

.add("station\_id\_till", "2200600")

.add("station\_from","РљРёС—РІ")

.add("station\_till","Р§РµСЂРЅС–РіС–РІ")

.add("date\_dep", "12.03.2015")

.add("time\_dep", "00:00")

.add("time\_dep\_till", "")

.add("another\_ec", "0")

.add("search", "");

parser.postExample("http://booking.uz.gov.ua/", q);

}

private void less() {

Iterator<Station> it = stations.iterator();

while(it.hasNext()) {

Station st = it.next();

String name = st.getName();

Iterator<TStation> it2 = tst.iterator();

TStation temp = null;

while(temp == null && it2.hasNext()) {

TStation t2 = it2.next();

if(t2.getName().equals(name)) temp = t2;

}

if(temp != null) {

tst.remove(temp);

}

}

System.out.println();

System.out.println("less=" + tst.size());

Iterator<TStation> t3 = tst.iterator();

while(t3.hasNext()) {

TStation tr = t3.next();

System.out.println(tr.getName() + " . " + tr.getId());

}

}

private void getHrefs() {

Iterator<String> it = hrefs.iterator();

while(it.hasNext()) System.out.println(it.next());

}

private void getStations() {

System.out.println();

Iterator<Station> it = stations.iterator();

while(it.hasNext()) System.out.println(it.next().getName());

}

private ArrayList<int[]> parseData(String text) {

return null;

}

public void postExample(String url, QueryString query) throws IOException {

URLConnection conn = new URL(url).openConnection();

conn.setDoOutput(true);

//conn.setRequestProperty("Request URI", "purchase/search/");

OutputStreamWriter out = new OutputStreamWriter(conn.getOutputStream(), "ASCII");

//out.write("Request URI: /purchase/search/");

out.write(query.toString());

out.write("\r\n");

out.flush();

out.close();

String html = readStreamToString(conn.getInputStream(), "UTF-8");

System.out.println("URL:" + url);

System.out.println("Html:\n" + html);

}

private String readStreamToString(InputStream in, String encoding)

throws IOException {

StringBuffer b = new StringBuffer();

InputStreamReader r = new InputStreamReader(in, encoding);

int c;

while ((c = r.read()) != -1) {

b.append((char)c);

}

return b.toString();

}

}

package Parser;

import java.util.ArrayList;

public class StringAnalyzer {

public static ArrayList<TStation> getStationList(String s) {

ArrayList<TStation> res = new ArrayList<TStation>();

String name = null;

String id = null;

String country = null;

int i = 0;

while(i < s.length()) {

while(i < s.length() && s.charAt(i) != '"') i++;

i++;

int j = i + 1;

while(j < s.length() && s.charAt(j) != '(') j++;

j--;

name = s.substring(i, j);

i = j + 2;

j = j + 3;

while(j < s.length() && s.charAt(j) != ')') j++;

country = s.substring(i, j);

i = j + 2;

j = j + 3;

while(j < s.length() && s.charAt(j) != '"') j++;

id = s.substring(i, j);

i = j + 3;

res.add(new TStation(name,id ,country));

}

return res;

}

}

package ORMroad;

import org.hibernate.SessionFactory;

import org.hibernate.cfg.Configuration;

public class HibernateUtil {

private static final SessionFactory sessionFactory;

static {

try {

sessionFactory = new Configuration().configure().buildSessionFactory();

} catch (Throwable ex) {

System.err.println("Initial SessionFactory creation failed." + ex);

throw new ExceptionInInitializerError(ex);

}

}

public static SessionFactory getSessionFactory() {

return sessionFactory;

}

}

package ORMroad;

// Generated 18.03.2015 1:22:07 by Hibernate Tools 3.4.0.CR1

import javax.xml.bind.annotation.XmlAccessType;

import javax.xml.bind.annotation.XmlAccessorType;

import javax.xml.bind.annotation.XmlRootElement;

import java.util.HashSet;

import java.util.Set;

/\*\*

\* Route generated by hbm2java

\*/

@XmlRootElement(name = "route")

@XmlAccessorType(XmlAccessType.FIELD)

public class Route implements java.io.Serializable {

private Integer routeId;

private String train;

private String time;

private String travelTime;

private String name;

private Set waypoints = new HashSet(0);

public Route() {

}

public Route(String train, String time, String travelTime, String name) {

this.train = train;

this.time = time;

this.travelTime = travelTime;

this.name = name;

}

public Route(String train, String time, String travelTime, String name,

Set waypoints) {

this.train = train;

this.time = time;

this.travelTime = travelTime;

this.name = name;

this.waypoints = waypoints;

}

public Integer getRouteId() {

return this.routeId;

}

public void setRouteId(Integer routeId) {

this.routeId = routeId;

}

public String getTrain() {

return this.train;

}

public void setTrain(String train) {

this.train = train;

}

public String getTime() {

return this.time;

}

public void setTime(String time) {

this.time = time;

}

public String getTravelTime() {

return this.travelTime;

}

public void setTravelTime(String travelTime) {

this.travelTime = travelTime;

}

public String getName() {

return this.name;

}

public void setName(String name) {

this.name = name;

}

public Set getWaypoints() {

return this.waypoints;

}

public void setWaypoints(Set waypoints) {

this.waypoints = waypoints;

}

public void splitTrainName() {

String[] trains = train.split(" ");

this.setTrain(trains[0]);

}

}

package ORMroad;

// Generated 18.03.2015 1:22:07 by Hibernate Tools 3.4.0.CR1

import javax.xml.bind.annotation.XmlAccessType;

import javax.xml.bind.annotation.XmlAccessorType;

import javax.xml.bind.annotation.XmlRootElement;

import java.util.HashSet;

import java.util.Set;

/\*\*

\* Station generated by hbm2java

\*/

@XmlRootElement(name = "station")

@XmlAccessorType(XmlAccessType.FIELD)

public class Station implements java.io.Serializable {

private Integer stationId;

private String name;

private Set waypoints = new HashSet(0);

public Station() {

}

public Station(String name) {

this.name = name;

}

public Station(String name, Set waypoints) {

this.name = name;

this.waypoints = waypoints;

}

public String getName() {

return this.name;

}

public void setName(String name) {

this.name = name;

}

public Set getWaypoints() {

return this.waypoints;

}

public void setWaypoints(Set waypoints) {

this.waypoints = waypoints;

}

}

package ORMroad;

// Generated 18.03.2015 1:22:07 by Hibernate Tools 3.4.0.CR1

import javax.xml.bind.annotation.XmlAccessType;

import javax.xml.bind.annotation.XmlAccessorType;

import javax.xml.bind.annotation.XmlElement;

import javax.xml.bind.annotation.XmlRootElement;

/\*\*

\* Waypoint generated by hbm2java

\*/

@XmlRootElement(name = "wayPoint")

@XmlAccessorType(XmlAccessType.FIELD)

public class WayPoint implements java.io.Serializable {

private Integer wayPointId;

@XmlElement(name = "route")

private Route route;

@XmlElement(name = "station")

private Station station;

private String arrival;

private String despatch;

public WayPoint() {

}

public WayPoint(Route route, Station station, String arrival,

String despatch) {

this.route = route;

this.station = station;

this.arrival = arrival;

this.despatch = despatch;

}

public Integer getWayPointId() {

return this.wayPointId;

}

public void setWayPointId(Integer wayPointId) {

this.wayPointId = wayPointId;

}

public Route getRoute() {

return this.route;

}

public void setRoute(Route route) {

this.route = route;

}

public Station getStation() {

return this.station;

}

public void setStation(Station station) {

this.station = station;

}

public String getArrival() {

return this.arrival;

}

public void setArrival(String arrival) {

this.arrival = arrival;

}

public String getDespatch() {

return this.despatch;

}

public void setDespatch(String despatch) {

this.despatch = despatch;

}

# ДОДАТОК В

**Висновок**

Виконання цієї роботи дало мені можливість закріпити знання Java та MySQL, а також поліпшило мої вміння працювати в команді. Я використав такі технології, як Hibernate для зв’язку з базою, Maven для збирання проекту, GitHub для покращення спільного використання ресурсів та сервер GlasFish.

Результатом моєї роботи є модель, що є частиною одного великого проекту «Сайт «Розклад руху поїздів».