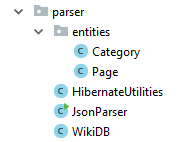
**Структура пакетов и файлов**



**Код парсера**

**public class** JsonParser {  
 **public static void** main (String[] args) **throws** IOException {  
 *parseDataToDatabase*();  
  
 *saveDataToFiles*();  
  
 System.***out***.println(**"FINISH"**);  
  
 **return**;  
 }  
  
 **public static void** saveDataToFiles() **throws** IOException {  
 System.***out***.println(**"Saving categories to file..."**);  
 WikiDB.*getInstance*().saveAllCategoriesToFile(**"D:\\BigData\\Categories.txt"**);  
 System.***out***.println(**"Categories are saved to file Categories.txt"**);  
  
 System.***out***.println(**"Saving pages to file..."**);  
 WikiDB.*getInstance*().saveAllPagesToFile(**"D:\\BigData\\Pages.txt"**);  
 System.***out***.println(**"Pages are saved to file Pages.txt"**);  
  
 System.***out***.println(**"Saving all data to file..."**);  
 WikiDB.*getInstance*().saveAllDataToFile(**"D:\\BigData\\AllData.txt"**);  
 System.***out***.println(**"All data is saved to file AllData.txt"**);  
 }  
  
 **public static void** parseDataToDatabase() **throws** IOException {  
 System.***out***.println(**"Parsing categories..."**);  
 *parseCategoriesToDatabase*();  
 System.***out***.println(**"Finished parsing categories!"**);  
  
 List<Category> categories = WikiDB.*getAllCategories*();  
  
 System.***out***.println(**"Parsing pages..."**);  
 *parsePagesToDatabase*(categories);  
 System.***out***.println(**"Finished parsing pages!"**);  
 }  
  
 **public static void** parseCategoriesToDatabase() **throws** IOException {  
 **int** counter=1;  
 **for** (**char** letter = **'A'**; letter <= **'Z'**; ++letter) {  
 String url = **"https://en.wikipedia.org/w/api.php?action=query&acprop=size&format=json&list=allcategories&aclimit=max&acprefix="** + letter;  
 JSONObject json = *readJsonFromUrl*(url);  
  
 JSONObject query = **new** JSONObject(json.get(**"query"**).toString());  
  
 JSONArray categories = (JSONArray) query.get(**"allcategories"**);  
 Iterator<Object> iterator = categories.iterator();  
  
 **while** (iterator.hasNext()) {  
 System.***out***.println(**"Category - "** + counter++ + **"/13000"**);  
 JSONObject categoryJSON = (JSONObject) iterator.next();  
  
 Category category = **new** Category(categoryJSON.get(**"\*"**).toString(), Integer.*parseInt*(categoryJSON.get(**"files"**).toString()), Integer.*parseInt*(categoryJSON.get(**"pages"**).toString()));  
 WikiDB.*getInstance*().addCategory(category);  
 }  
 }  
 }  
  
 **public static** JSONObject readJsonFromUrl(String url) **throws** IOException, JSONException {  
 InputStream is = **new** URL(url).openStream();  
 **try** {  
 BufferedReader rd = **new** BufferedReader(**new** InputStreamReader(is, Charset.*forName*(**"UTF-8"**)));  
  
 StringBuilder sb = **new** StringBuilder();  
 **int** cp;  
 **while** ((cp = rd.read()) != -1) {  
 sb.append((**char**) cp);  
 }  
  
 JSONObject json = **new** JSONObject(sb.toString());  
 **return** json;  
 } **finally** {  
 is.close();  
 }  
 }  
  
 **public static void** parsePagesToDatabase(List<Category> categories) **throws** IOException {  
 **int** counter = 1;  
  
 **for** (**int** k=0; k < categories.size() ; ++k) {  
 **try** {  
 System.***out***.println(**"Category - "** + counter++ + **"/13000"**);  
  
 String categoryName = categories.get(k).getCategoryName().replaceAll(**" "**, **"%20"**);  
 categoryName = categoryName.replaceAll(**"&"**, **"%26"**);  
 String url = **"https://en.wikipedia.org/w/api.php?action=query&list=categorymembers&format=json&cmlimit=max&cmtitle=Category:"** + categoryName;  
  
 JSONObject json = *readJsonFromUrl*(url);  
 JSONObject query = **new** JSONObject(json.get(**"query"**).toString());  
 JSONArray pages = (JSONArray) query.get(**"categorymembers"**);  
  
 Iterator<Object> iterator = pages.iterator();  
  
 **while** (iterator.hasNext()) {  
  
 JSONObject pageJSON = (JSONObject) iterator.next();  
 **if** (!pageJSON.get(**"title"**).toString().contains(**"Category:"**)) {  
 Page page = **new** Page(pageJSON.get(**"title"**).toString(), categories.get(k));  
 **try** {  
 WikiDB.*getInstance*().addPage(page);  
 } **catch** (HibernateException ex) {  
 **continue**;  
 }  
 }  
 }  
 } **catch** (UnknownHostException ex) {  
 **continue**;  
 }  
 }  
 }  
}

**WikiDB – класс для работы с базой данных**

**public class** WikiDB {  
 **private static** WikiDB *instance* = **null**;  
  
 **public static synchronized** WikiDB getInstance() {  
 **if** (*instance* == **null**) {  
 *instance* = **new** WikiDB();  
 }  
 **return** *instance*;  
 }  
  
 **private** WikiDB() {  
 }  
  
 **public synchronized void** addCategory(Category category) {  
  
 SessionFactory sessionFactory = HibernateUtilities.*getSessionFactory*();  
  
 **try** (Session session = sessionFactory.openSession()) {  
 Transaction tx;  
 tx = session.beginTransaction();  
  
 session.save(category);  
 tx.commit();  
 }  
 }  
  
 **public synchronized void** addPage(Page page) **throws** HibernateException {  
  
 SessionFactory sessionFactory = HibernateUtilities.*getSessionFactory*();  
  
 **try** (Session session = sessionFactory.openSession()) {  
 Transaction tx;  
 tx = session.beginTransaction();  
  
 session.save(page);  
 tx.commit();  
 } **catch** (HibernateException ex) {  
 **throw** ex;  
 }  
 }  
  
 **public synchronized void** saveAllCategoriesToFile(String filePath) **throws** IOException {  
 Path file = Paths.*get*(filePath);  
  
 SessionFactory sessionFactory = HibernateUtilities.*getSessionFactory*();  
  
 List<Category> categories = **new** ArrayList<>();  
  
 **try** (Session session = sessionFactory.openSession()) {  
 Transaction tx;  
 tx = session.beginTransaction();  
 Query query = session.createQuery(**"select "** + **"new parser.entities.Category(c.id, c.categoryName, c.numberOfFiles, c.numberOfPages) "** + **"from Category c"**);  
  
 categories = query.getResultList();  
 tx.commit();  
 }  
  
 List<String> text = **new** ArrayList<>();  
 **for** (Category category : categories) {  
 text.add(category.toString());  
 }  
 Files.*write*(file, text, Charset.*forName*(**"UTF-8"**));  
 }  
  
 **public synchronized void** saveAllPagesToFile(String filePath) **throws** IOException {  
 Path file = Paths.*get*(filePath);  
  
 SessionFactory sessionFactory = HibernateUtilities.*getSessionFactory*();  
  
 List<Page> pages = **new** ArrayList<>();  
  
 **try** (Session session = sessionFactory.openSession()) {  
 Transaction tx;  
 tx = session.beginTransaction();  
 Query query = session.createQuery(**"select "** + **"new parser.entities.Page(p.id, p.pageName, p.category) "** + **"from Page p"**);  
  
 pages = query.getResultList();  
 tx.commit();  
 }  
  
 List<String> text = **new** ArrayList<>();  
 **for** (Page page : pages) {  
 text.add(page.toString());  
 }  
 Files.*write*(file, text, Charset.*forName*(**"UTF-8"**));  
 }  
  
 **public synchronized void** saveAllDataToFile(String filePath) **throws** IOException {  
 Path file = Paths.*get*(filePath);  
 List<String> text = **new** ArrayList<>();  
  
 List<Category> categories = **new** ArrayList<>();  
  
 **for** (**long** i = 1; i <= 13000 ; i+=500) {  
  
 SessionFactory sessionFactory = HibernateUtilities.*getSessionFactory*();  
  
 **try** (Session session = sessionFactory.openSession()) {  
 Transaction tx;  
 tx = session.beginTransaction();  
  
 Query query = session.createQuery(**"select "** + **"new parser.entities.Category(c.id, c.categoryName, c.numberOfFiles, c.numberOfPages) "** + **"from Category c "** + **"where c.id>=:min and c.id<=:max"**);  
  
 query.setParameter(**"min"**, i);  
 query.setParameter(**"max"**, i+499);  
  
 categories = query.getResultList();  
 tx.commit();  
 }  
  
 **for** (Category category : categories) {  
 System.***out***.println(**"Category - "** + category.getId() + **"/13000"**);  
  
 text.add(category.toString());  
  
 SessionFactory sessionFactory2 = HibernateUtilities.*getSessionFactory*();  
  
 List<Page> pages = **new** ArrayList<>();  
  
 **try** (Session session2 = sessionFactory2.openSession()) {  
 Transaction tx2;  
 tx2 = session2.beginTransaction();  
 Query query2 = session2.createQuery(**"select "** + **"new parser.entities.Page(p.id, p.pageName, p.category) "** + **"from Page p where p.category.id=:id"**);  
 query2.setParameter(**"id"**, category.getId());  
  
 pages = query2.getResultList();  
 tx2.commit();  
 }  
  
 **for** (Page page : pages) {  
 text.add(**"\t->\t"** + page.toStringWithoutCategory());  
 }  
 }  
 }  
 Files.*write*(file, text, Charset.*forName*(**"UTF-8"**));  
 }  
  
 **public static** List<Category> getAllCategories(){  
 SessionFactory sessionFactory = HibernateUtilities.*getSessionFactory*();  
  
 List<Category> categories = **new** ArrayList<>();  
  
 **try** (Session session = sessionFactory.openSession()) {  
 Transaction tx;  
 tx = session.beginTransaction();  
 Query query = session.createQuery(**"select "** + **"new parser.entities.Category(c.id, c.categoryName, c.numberOfFiles, c.numberOfPages) "** + **"from Category c"**);  
  
 categories = query.getResultList();  
 tx.commit();  
 }  
 **return** categories;  
 }  
}

**HibernateUtilities**

**public class** HibernateUtilities {  
  
 **private static** SessionFactory *sessionFactory* = **null**;  
  
 **private** HibernateUtilities() {  
 }  
  
 **public static** SessionFactory getSessionFactory() {  
 **if** (*sessionFactory* == **null**) {  
 *sessionFactory* = **new** Configuration()  
 .configure()  
 .buildSessionFactory();  
 }  
 **return** *sessionFactory*;  
 }  
}

**Entities**

Category

@Entity  
@Table(name = **"categories"**)  
**public class** Category {  
 @Id  
 @GeneratedValue(strategy = GenerationType.***IDENTITY***)  
 **private long id**;  
  
 @Column(name = **"categoryName"**)  
 **private** String **categoryName**;  
  
 @Column(name = **"numberOfFiles"**)  
 **private long numberOfFiles**;  
  
 @Column(name = **"numberOfPages"**)  
 **private long numberOfPages**;  
  
 **public** Category(String categoryName, **long** numberOfFiles, **long** numberOfPages) {  
 **this**.**categoryName** = categoryName;  
 **this**.**numberOfFiles** = numberOfFiles;  
 **this**.**numberOfPages** = numberOfPages;  
 }  
  
 **public** Category() {  
 **this**.**id** = 0;  
 **this**.**categoryName** = **""**;  
 **this**.**numberOfFiles** = 0;  
 **this**.**numberOfPages** = 0;  
 }  
  
 **public** Category(**long** id, String categoryName, **long** numberOfFiles, **long** numberOfPages) {  
 **this**.**id** = id;  
 **this**.**categoryName** = categoryName;  
 **this**.**numberOfFiles** = numberOfFiles;  
 **this**.**numberOfPages** = numberOfPages;  
 }  
  
 **public long** getId() {  
 **return id**;  
 }  
  
 **public void** setId(**long** id) {  
 **this**.**id** = id;  
 }  
  
 **public** String getCategoryName() {  
 **return categoryName**;  
 }  
  
 **public void** setCategoryName(String categoryName) {  
 **this**.**categoryName** = categoryName;  
 }  
  
 **public long** getNumberOfFiles() {  
 **return numberOfFiles**;  
 }  
  
 **public void** setNumberOfFiles(**long** numberOfFiles) {  
 **this**.**numberOfFiles** = numberOfFiles;  
 }  
  
 **public long** getNumberOfPages() {  
 **return numberOfPages**;  
 }  
  
 **public void** setNumberOfPages(**long** numberOfPages) {  
 **this**.**numberOfPages** = numberOfPages;  
 }  
  
 @Override  
 **public** String toString() {  
 **return "Category{"** +  
 **"id="** + **id** +  
 **",\t\tcategoryName='"** + **categoryName** + **'\''** +  
 **",\t\t\t\tnumberOfFiles="** + **numberOfFiles** +  
 **",\t\tnumberOfPages="** + **numberOfPages** +  
 **'}'**;  
 }  
}

Page

@Entity  
@Table(name = **"pages"**)  
**public class** Page {  
 @Id  
 @GeneratedValue(strategy = GenerationType.***IDENTITY***)  
 **private long id**;  
  
 @Column(name = **"pageName"**)  
 **private** String **pageName**;  
  
 @ManyToOne(fetch = FetchType.***EAGER***, cascade = CascadeType.***ALL***)  
 @JoinColumn(name = **"category\_id"**)  
 **private** Category **category**;  
  
 **public** Page(String pageName, Category category) {  
 **this**.**pageName** = pageName;  
 **this**.**category** = category;  
 }  
  
 **public** Page(**long** id, String pageName, Category category) {  
 **this**.**id** = id;  
 **this**.**pageName** = pageName;  
 **this**.**category** = category;  
 }  
  
 **public** Page() {  
 **this**.**id** = 0;  
 **this**.**pageName** = **""**;  
 **this**.**category** = **null**;  
 }  
  
 **public long** getId() {  
 **return id**;  
 }  
  
 **public void** setId(**long** id) {  
 **this**.**id** = id;  
 }  
  
 **public** String getPageName() {  
 **return pageName**;  
 }  
  
 **public void** setPageName(String pageName) {  
 **this**.**pageName** = pageName;  
 }  
  
 **public** Category getCategory() {  
 **return category**;  
 }  
  
 **public void** setCategory(Category category) {  
 **this**.**category** = category;  
 }  
  
 @Override  
 **public** String toString() {  
 **return "Page{"** +  
 **"id="** + **id** +  
 **",\tpageName='"** + **pageName** + **'\''** +  
 **",\t\t\tcategory="** + **category** +  
 **'}'**;  
 }  
  
 **public** String toStringWithoutCategory() {  
 **return "Page{"** +  
 **"id="** + **id** +  
 **",\tpageName='"** + **pageName** + **'\''** +  
 **'}'**;  
 }  
}