Описание учебного проекта

JMemcached

Описание проекта:

1)Необходимо реализовать проект Jmemcached, который является упрощенной Java версией популярного на сегодняшний день NoSQL продукта хранение данных в оперативной памяти memcached (http://www.memcached.org/ https://ru.wikipedia.org/wiki/Memcached);
2)Jmemcached будет хранить все данные в виде хэш таблицы только в оперативной памяти;
3)Jmemcached будет использовать клиент серверную архитектуру.

Детали проекта:

1)Для реализации проекта **Jmemcached** необходимо разработать два компонента: Jmemcached-server и Jmemcached-client;
2)Jmemcached-server будет автономно работать как Windows сервис и ожидать клиентских подключений;
3)Компонент Jmemcached-client является клиентской библиотекой, которую необходимо будет подключить к отдельному Java проекту, которому необходимо использование Jmemcached-server.
4)Если провести аналогию с SQL, то Jmemcached-server – это сервер базы данных (например PostgreSQL сервер), а Jmemcached-client – это JDBC драйвер и JDBC API.

Jmemcached-client:

public interface Client extends AutoCloseable {

Status put(String key, Serializable object) throws IOException; - сохраняет объект **object** в хэш таблицу на сервере под ключом равным **key** на неограниченное время;

Status put(String key, Serializable object, Integer ttl, TimeUnit timeUnit) throws IOException; - сохраняет объект **object** в хэш таблицу на сервере под ключом равным **key** на время **ttl** в единицах измерения **timeUnit.** Если время жизни объекта истечет, то сервер должен удалить объект автоматически; **TimeUnit** - это перечисление, которое используется для задания единиц изменения времени, например секунды, минуты, часы и т.д.

<T extends Serializable> T get(String key) throws IOException; - возвращает объект по ключу **key** или **null**, если объекта не существует;

Status remove(String key) throws IOException; - удаляет объект по ключу **key;**

Status clear() throws IOException; - удаляет все объекты с хэш таблицы.

}

Jmemcached-client:

Статусы состояния, которые возвращает данный клиент следующие:

```
public enum Status {
   ADDED,   REPLACED,   NOT_FOUND,   REMOVED,   CLEARED;
}
```

Описание значений Status, которые возвращаются методами клиента:

Status put(String key, Serializable object) throws IOException; - возвращает **ADDED** если по указанному ключу не было объекта и **REPLACED** – если по данному ключу хранился на сервере некоторый объект;

Status remove(String key) throws IOException; - возвращает **REMOVED** если объект был успешно удален и **NOT_FOUND** если по заданному ключу нет объекта**;**

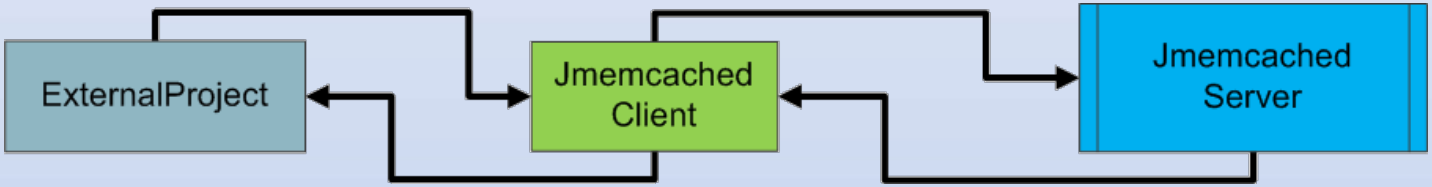
Status clear() throws IOException; - всегда возвращает **CLEARED**.

Пример использования:

```
public class ExternalProject{
    public static void main(String[] args) throws Exception {
        // Подключаемся к серверу Jmemcached, который работает на localhost на порту 9010
            try (Client client = JMemcachedClientFactory.buildNewClient("localhost", 9010)) {
            client.put("test", "Hello world"); //сохраняем текст под ключом test на сервере Jmemcached
            System.out.println(client.get("test")); //запрашиваем объект по ключу test

            client.remove("test"); //удаляем объект по ключу test
            System.out.println(client.get("test")); //должно отобразить null – объект был удален!

            client.put("test", "Hello world"); //сохраняем текст под ключом test
            client.put("test", new BusinessObject("TEST")); //выполняем замену объекта по ключу test

            System.out.println(client.get("test")); //Увидим BusinessObject объект
            client.clear(); //Очищаем все данные на сервере Jmemcached
            System.out.println(client.get("test")); //должно отобразить null – объект был удален!

            client.put("devstudy", "Devstudy JMemcached", 2, TimeUnit.SECONDS); //сохраняем текст под
ключом devstudy и указываем время жизни 2 секунды
            TimeUnit.SECONDS.sleep(3); //замораживаем текущий поток на 3 секунды
            System.out.println(client.get("devstudy")); //должно отобразить null – объект был удален
автоматически так как истекло время жизни объекта (2 секунды) на сервере
        }
//Благодаря конструкции try-with-resource после выполнения работы будет вызван метод close(),
который закроет соединение с сервером Jmemcached
    }
}
```

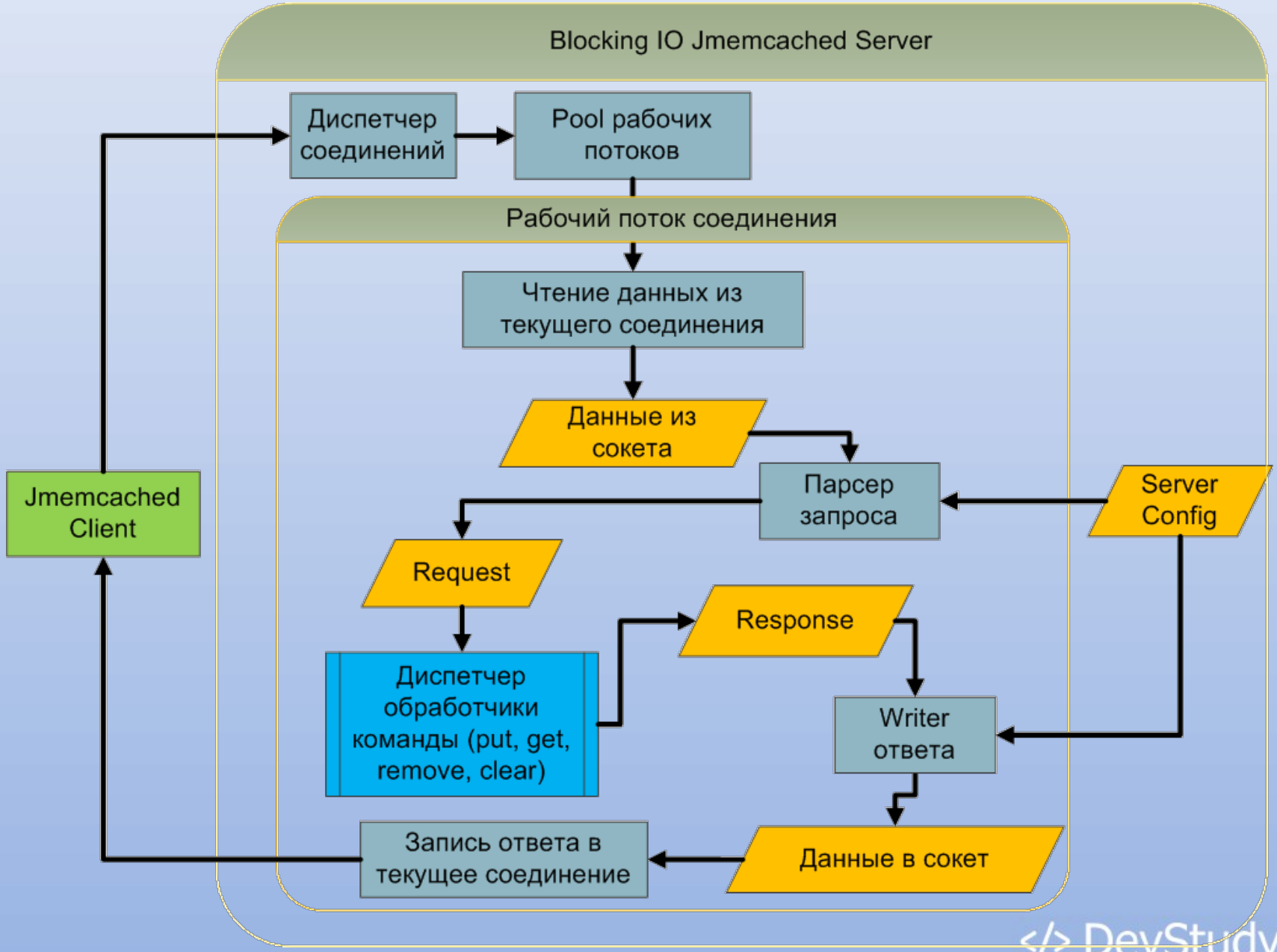
Схема взаимодействия компонентов:

Компоненты будут взаимодействовать по следующей схеме:



1. **Jmemcached server** будет запущен как Windows сервис и будет слушать порт 9010;
2. **ExternalProject** является примером отдельного проекта, который будет использовать Jmemcached. Для взаимодействия с **Jmemcached server ExternalProject** необходимо использовать **Jmemcached client**, который будет преобразовывать команды клиента (put, get, remove, clear) в формат понятный серверу, а сервер уже будет их выполнять.
3. Для взаимодействия **Jmemcached server** и **Jmemcached client** необходимо разработать протокол, в котором будут описаны правила передачи команд и данных с сервера и на сервер.
4. Обязанность протокола, который Вам необходимо разработать - преобразовать команды и данные из Java объектов в последовательность байтов (при использовании бинарного типа протокола) или в последовательность текстовых символов (при использовании текстового протокола). Например для взаимодействия браузера и web сервера используется текстовый протокол HTTP, который в текстовом виде передает данные и команды между этими приложениями по сети.
5. Для того, чтобы преобразовать Java объект в поток байт и наоборот, можно использовать встроенную в Java технологию сериализации.
6. Таким образом последовательность выполнения команды put следующая: **Jmemcached client** получив команду put с ключом и объектом преобразует эти данные в последовательность байтов, согласно правилам разработанного Вами протокола и передает на **Jmemcached server**; **Jmemcached server** получив поток байт, восстанавливает команду, ключ и данные и сохраняет данные по указанному ключу; после этого возвращает **Jmemcached client** результат операции в виде потока байтов; **Jmemcached client** восстанавливает статус из потока байтов и возвращает объект Status с результатом операции в проект **ExternalProject.**

Архитектура Jmemcached server:



Для разработки проекта Вам необходимо:

1)Разработать и реализовать протокол взаимодействия клиента и сервера;
2)Реализовать модули преобразования Java объектов в поток байт и наоборот на основе технологии сериализации;
3)Реализовать модуль хранения данных в виде хэш таблицы с автоматическим удалением устаревших записей из хэш таблицы;
4)Реализовать Jmemcached server согласно описанной архитектуре на предыдущем слайде;
5)Реализовать Jmemcached client, который будет взаимодействовать с сервером по Вашему протоколу;
6)Покрыть все проекты Unit тестами.