



Модуль 08 - Piscine Java

Розетки

Резюме: Сегодня вы реализуете базовый механизм клиент-серверного приложения на основе Java-Sockets API

Содержание

I	Предислови е	2
II	Инструкции	3
III	Упражнение Регистрация 00 :	5
IV	Упражнение Обмен сообщениями 01 :	7
V	Упражнение Номера 02 :	9

Глава I

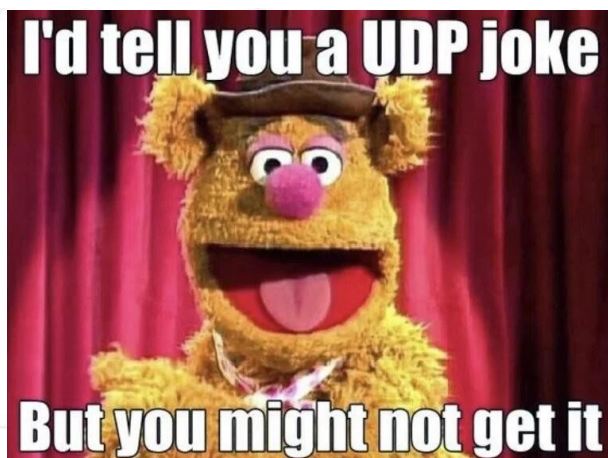
Предисловие

Взаимодействие клиент/сервер является основой современных систем. Сервер выполняет большой объем бизнес-логики и хранения информации. В результате нагрузка на клиентское приложение значительно снижается.

Разделение логики на серверные и клиентские компоненты позволяет гибко построить общую архитектуру системы, если серверные и клиентские реализации максимально независимы.

Клиент и сервер взаимодействуют через многочисленные протоколы, описанные в сетевой модели OSI как различные уровни:

Layer	Example
7. Application	HTTP
6. Representation	ASCII
5. Session	RPC
4. Transport	TCP, UDP
3. Network	IPv4
2. Channel	Ethernet, DSL
1. Physical	USB, "twisted pair"



Глава II


Инструкции

- Используйте эту страницу как единственную ссылку. Не слушайте никаких слухов и домыслов о том, как приготовить раствор.
- Теперь для вас существует только одна версия Java - 1.8. Убедитесь, что компилятор и интерпретатор этой версии установлены на вашей машине.
- Вы можете использовать IDE для написания и отладки исходного кода.
- Код чаще читают, чем пишут. Внимательно прочитайте [документ](#), в котором приведены правила форматирования кода. При выполнении каждой задачи убедитесь, что вы следуете общепринятым [стандартам Oracle](#)
- Комментарии не допускаются в исходном коде вашего решения. Они затрудняют чтение кода.
- Обратите внимание на разрешения ваших файлов и каталогов.
- Для оценки ваше решение должно находиться в вашем GIT-репозитории.
- Ваши решения будут оценивать ваши товарищи по аквариуму.
- Вы не должны оставлять в своем каталоге никаких других файлов, кроме тех, которые явно указаны в инструкциях к упражнению. Рекомендуется изменить свой .gitignore во избежание несчастных случаев.
- Когда вам нужно получить точный вывод в ваших программах, запрещено выводить предварительно рассчитанный вывод вместо правильного выполнения упражнения.
- У вас есть вопрос? Спросите своего соседа справа. В противном случае попробуйте поговорить с соседом слева.
- Ваше справочное пособие: товарищи / Интернет / Google. И еще кое-что. На любой ваш вопрос есть ответ на Stackoverflow. Узнайте, как правильно задавать вопросы.
- Внимательно прочитайте примеры. В них могут потребоваться вещи, которые не указаны в предмете.
- Используйте "System.out" для вывода

- И да пребудет с вами Сила!
- Никогда не оставляйте на завтра то, что вы можете сделать сегодня ;)

Глава III

Упражнение 00: Регистрация

	Упражнение 00
Регистрация	
Каталог для сдачи : ex00/	
Файлы для сдачи : Папка	
чата Разрешенные функции	
: Все	

Прежде чем приступить к созданию полномасштабного многопользовательского чата, необходимо реализовать основные функции и построить фундаментальную архитектуру системы.

Теперь необходимо создать два приложения: `socket-server` и `socket-client`. Сервер должен поддерживать подключение одного клиента и быть выполнен в виде отдельного проекта Maven. JAR-файл сервера запускается следующим образом:

```
$ java -jar target/socket-server.jar --port=8081
```

Клиент также является отдельным проектом:

```
$ java -jar target/socket-client.jar --server-port=8081
```

В этом задании необходимо реализовать функциональность регистрации. Пример работы клиента:

```
Привет из Сервера!  
> signUp  
Введите имя пользователя:  
> Марсел  
Введите пароль:  
> qwerty007  
Успешно!
```

Соединение должно быть закрыто после появления сообщения `Successful!`

Чтобы обеспечить безопасное хранение паролей, используйте механизм хэширования с помощью `PasswordEncoder` и `BCryptPasswordEncoder` (см. компоненты `Spring Security`). Бины для этого компонента должны быть описаны в классе конфигурации `SocketsApplicationConfig` и использоваться в

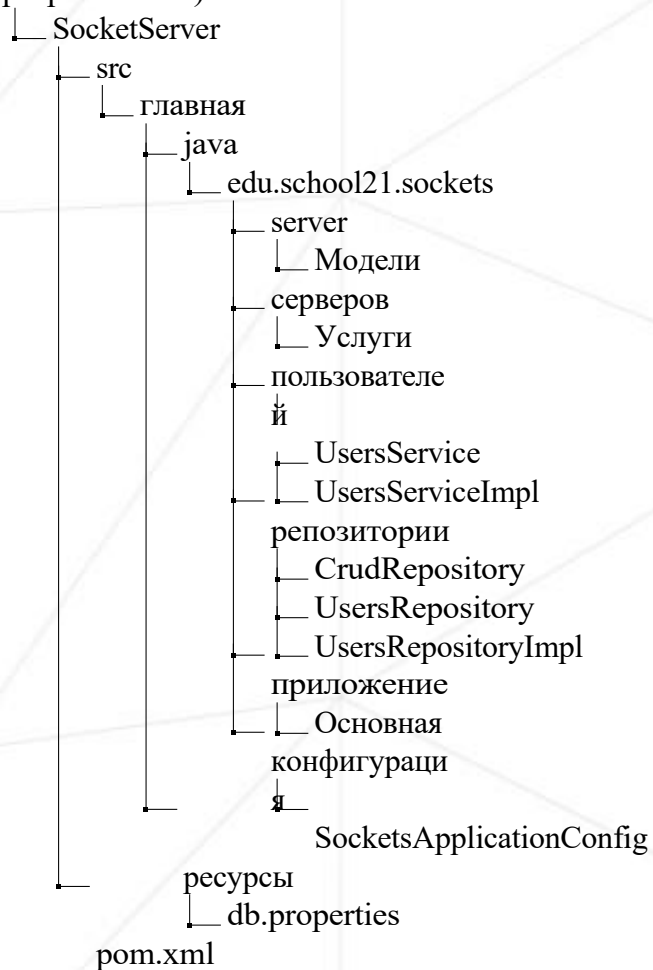
UserService.

Ключевая логика взаимодействия клиента и сервера и использование UsersService через Spring Context должны быть реализованы в классе Server.

Дополнительные требования:


- Использование одного источника данных - HikariCP
- Работа репозитория должна быть реализована через JdbcTemplate
- Сервисы, хранилища, утилитарные классы должны быть контекстными корзинами.

Архитектура серверного приложения (клиентское приложение - на усмотрение разработчика): Char



Глава IV

Упражнение 01: Обмен сообщениями

	Упражнение 01
Обмен	
Каталог для сдачи : ex01/	
Файлы для сдачи : Chat-	сообщениями
folder Разрешенные	
функции : Все	

После реализации магистрали приложений необходимо обеспечить многопользовательский обмен сообщениями.

Вам нужно изменить приложение так, чтобы оно поддерживало следующий жизненный цикл пользователя чата:

1. Регистрация
2. Войти в систему (если пользователь не обнаружен, закрыть соединение)
3. Отправка сообщений (каждый пользователь, подключенный к серверу, должен получить сообщение)
4. Выход из системы

Пример работы приложения на стороне клиента:

```
Привет из Сервера!  
> signIn  
Введите имя пользователя:  
> Марсел  
Введите пароль:  
> qwerty007  
Начать обмен  
сообщениями  
> Привет!  
Марсель:  
Здравствуйте!  
NotMarsel: Пока!
```

```
> Выход  
Вы покинули чат.
```

Каждое сообщение должно быть сохранено в базе данных и содержать следующую информацию: Отправитель

- Текст сообщения


- Время отправки

Примечание:

- Для всестороннего тестирования необходимо запустить несколько jar-файлов клиентского приложения.

Глава V

Упражнение 02 : Комнаты

	Упражнение 02
Каталог для сдачи : ex02/	Номера
Файлы для сдачи : Chat-	
folder Разрешенные	
функции : Все	

Чтобы сделать наше приложение полнофункциональным, давайте добавим в него концепцию "чатов". Каждый чат может иметь определенный набор пользователей. Чат содержит набор сообщений от участвующих пользователей.

Каждый пользователь может:

1. Создайте чат
2. Выберите чат
3. Отправить сообщение в чат
4. Покинуть чат

Когда пользователь снова входит в приложение, 30 последних сообщений отображаются в комнате, которую пользователь посещал ранее.

Пример работы приложения на стороне клиента:

```
Привет с сервера! Вход
в систему
Регистраци
я Выход
> 1
Введите имя пользователя:
> Марсел
Введите пароль:
> qwerty007
Создать
комнату
Выбрать
комнату
Выход
> 2
Комнаты:
Первая
комната
Простая
комната
```



```
JavaRoom
Выход
> 3
Java Room ---
JavaMan: Привет!
> Привет!
Марсель:
Здравствуйте!
> Выход
```

Вы покинули чат.

Использование формата JSON для обмена сообщениями станет для вас специальной задачей. Таким образом, каждая команда пользователя или сообщение должны быть переданы на сервер (и получены с сервера) в виде строки JSON.

Например, команда для отправки сообщения может выглядеть следующим образом (конкретное содержание сообщений - на усмотрение разработчика):

```
{
  "message" : "Hello!",
  "fromId" : 4,
  "roomId" : 10
}
```