

Клиент/сервер своими руками

Урок 5



План курса

1

Лекция 1. Лямбды и Stream API

2

Семинар 1. Лямбды и Stream API

3

Лекция 2. Reflection API.

4

Семинар 2. Reflection API.

5

Лекция 3. Сериализация

6

Семинар 3. Сериализация

7

Лекция 4. Базы данных и инструменты взаимодействия с ними

8

Семинар 4. Базы данных и инструменты взаимодействия с ними

9


Лекция 5. Клиент/Сервер своими руками

10

Семинар 5. Клиент/Сервер своими руками



На прошлом уроке мы изучили:

-  JDBC
-  ORM
-  Hibernate
-  JPA



На этом уроке нас ждет:

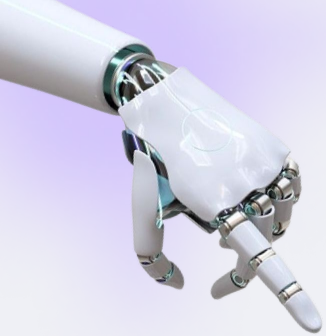


Работа с сетью

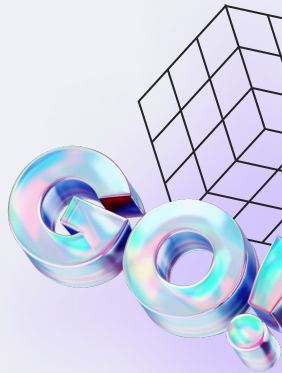


Библиотеки языка Java

Работа с сетью — это фактически обязательная составляющая работы с компьютером. Даже когда мы просто пишем текст или, тем более, рисуем в графическом редакторе. Мы ждём авто проверку, доступ к графическим ресурсам, примерам, онлайн помощь и многое другое. А значит, практически любая программа должна уметь работать с сетью. Инфраструктура сети уже устоялась и изменений в ней не предвидится, а в Java имеется пакет `java.net` специально для работы с сетью!



Socket является ключевым понятием в Java, связанным с сетевым программированием. Он представляет собой двусторонний канал связи между двумя процессами, выполняющимися на разных машинах, и служит для обмена данными между ними. Основная цель использования Socket в Java - это создание клиент-серверных приложений, где сервер принимает запросы от клиентов, обрабатывает их и отправляет обратно ответ. Это может включать в себя различные сетевые приложения, такие как чаты, онлайн-игры, системы сообщений и многое другое. Кроме того, Socket позволяет взаимодействовать с различными протоколами, такими как HTTP, FTP и другими, обеспечивая гибкость и универсальность в разработке Java-приложений.





Порт- это идентификатор, который используется для связи между клиентом и сервером. Он определяет протокол, по которому будет происходить обмен данными, и адрес сервера, к которому нужно подключиться.



Сервер должен уметь:

1. Слушать заданный порт, в ожидании запроса на подключение
2. После получения запроса, после начала формирования связи с клиентом, сервер должен продолжить слушать порт и быть готовым к новым подключениям
3. Идентифицировать участника чата по имени
4. Формировать оповещение о подключенных участниках чата
5. При подключении участника уметь оповестить оставшихся участников об этом
6. Уметь передавать сообщения всем участникам, кроме него самого

Клиент должен уметь:

1. **Подключение к серверу:** клиент должен иметь функционал для подключения к чат-серверу. Это включает в себя установку соединения с сервером, указание адреса и порта для подключения.
2. **Параллельная обработка сообщений:** клиент должен быть способен принимать и отправлять сообщения параллельно. Это означает, что клиент может одновременно отправлять свои сообщения и обрабатывать входящие сообщения от других пользователей чата.
3. **Обработка множественных сообщений:** клиент должен быть готов к ситуации, когда в процессе ввода своего сообщения могут приходить сообщения от других пользователей. Это предотвращает заикливание работы клиента на вводе или передаче сообщений.
4. **Корректное закрытие ресурсов:** клиент должен иметь механизм корректного закрытия всех используемых ресурсов при завершении работы. Это включает в себя закрытие сокета и других потоков.
5. **Простота использования:** клиент должен быть разработан так, чтобы им было легко пользоваться. Он должен предоставлять удобный интерфейс для ввода сообщений, подключения к серверу и отображения полученных сообщений.



Спасибо за внимание

