

# Базы данных

Урок 4



## 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



## Что будет на уроке сегодня мы поговорим о:

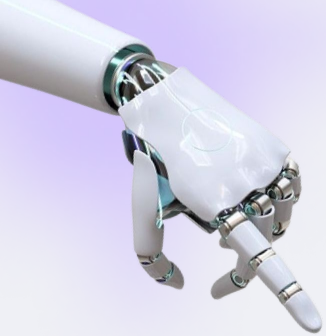
-  JDBC
-  ORM
-  Hibernate
-  JPA

**JDBC (Java Database Connectivity)** — это программный интерфейс, предоставляющий набор классов и методов для взаимодействия с базами данных из языка программирования Java. JDBC обеспечивает стандартизированный способ подключения к различным системам управления базами данных (СУБД), выполнения SQL-запросов, получения и обновления данных в базе. Он предоставляет абстракцию, позволяющую разработчикам писать приложения, которые могут взаимодействовать с базами данных, независимо от конкретной используемой СУБД. JDBC используется для создания портативных и эффективных приложений, работающих с данными в базах данных.



## Шаги по установке MySQL и поиску драйвера JDBC:

1. Выберем СУБД: Перейдем на официальный сайт MySQL по ссылке <https://www.mysql.com/>. Нажмем на вкладку "DOWNLOADS". Внизу страницы находим ссылку "MySQL Community (GPL) Downloads" и перейдем по ней.
2. Загрузим MySQL: На странице загрузки выбираем необходимую версию и загрузим MySQL Installer for Windows.
3. Установим MySQL: Запускаем загруженный установщик и следуем инструкциям по установке MySQL.
4. Найдем JDBC-драйвер: В поисковике введите "mysql driver java maven". Перейдите на первую ссылку, которая обычно ведет на maven repository. На странице выбора версий драйвера выберите нужную версию и перейдите на страницу с текстом.



**ORM** расшифровывается как **Object-Relational Mapping** (Отображение объектов на отношения). Это программная технология, которая связывает базы данных с системами, основанными на объектно-ориентированном программировании, путем создания виртуальной объектной базы данных. Она позволяет работать с данными в виде объектов в программном коде, а не в виде SQL-запросов к базе данных.

Основная идея ORM заключается в том, чтобы устранить несоответствие между объектами в приложении и таблицами в базе данных, предоставляя механизмы автоматического преобразования данных между ними. ORM-фреймворки, такие как Hibernate, предоставляют абстракцию над базой данных, упрощая и ускоряя разработку приложений.



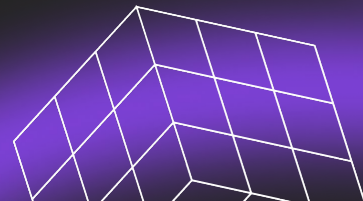


**Hibernate** - это фреймворк для языка Java, предназначенный для упрощения взаимодействия с реляционными базами данных. Он является одной из самых популярных реализаций технологии ORM (Object-Relational Mapping) в Java.





**Java Persistence API (JPA)** представляет собой стандартный интерфейс в языке Java для работы с системами управления реляционными базами данных. Этот стандарт был создан для облегчения и унификации разработки приложений, использующих базы данных, и предоставляет объектно-ориентированный способ взаимодействия с реляционными данными.





# Важно запомнить:

1. **JDBC (Java Database Connectivity):** Основой взаимодействия Java-приложений с базами данных является JDBC. Важно помнить, что JDBC предоставляет прямой доступ к базе данных с использованием SQL-запросов, требует подключения соответствующего драйвера и обеспечивает простоту и непосредственность работы с базой данных.
2. **ORM (Object-Relational Mapping):** Понимание принципов ORM, представленных фреймворком Hibernate, важно. Hibernate служит посредником между объектами в Java-коде и таблицами в базе данных, автоматизируя рутинные задачи, такие как создание, чтение, обновление и удаление данных. ORM поддерживает объектно-ориентированный подход к работе с данными.
3. **Hibernate и JPA (Java Persistence API):** Hibernate является одним из популярных фреймворков ORM, а JPA - стандартным интерфейсом Java для ORM. Важно знать, что JPA стандартизирует работу с ORM-фреймворками, предоставляя унифицированный набор аннотаций и методов для работы с данными.
4. **Различия между JDBC и Hibernate:** Необходимо помнить, что в то время как JDBC предоставляет более низкоуровневый доступ к базе данных с использованием SQL, Hibernate предлагает более высокоуровневый, объектно-ориентированный подход. Выбор между ними зависит от требований проекта и предпочтений разработчика.



**Спасибо за внимание**

