K29 - Σχεδίαση & Χρήση Βάσεων Δεδομένων

Java, JDBC & Spring — From Architecture to ORM

Κορρές Μιχαήλ

2026

Modern Applications Follow a 3-Tier Architecture

- **Web Client (Tier 1):** HTML, CSS, JavaScript

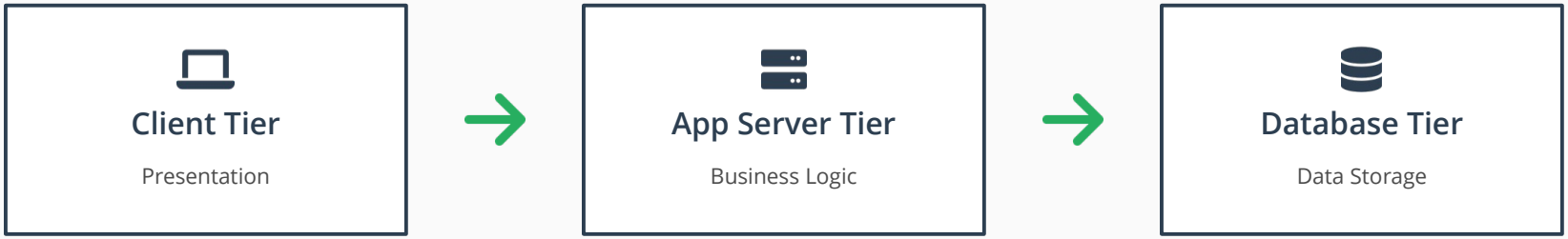
  What the user sees and interacts with

- **Application Server (Tier 2):** Java, Python, Go

  Business logic lives here

- **Database Server (Tier 3):** MySQL, PostgreSQL

  Persistent data storage



Your Application Server Acts as a Database Client

- The Application Server sends SQL queries to the DB Server

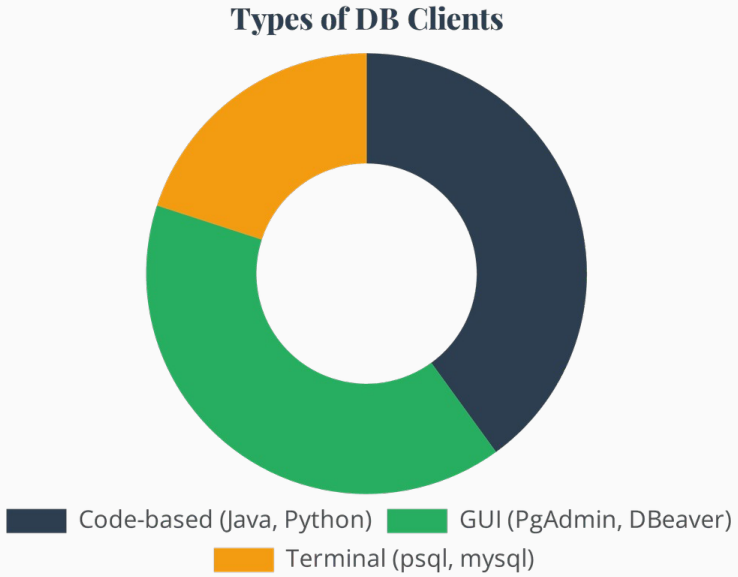
  This makes it a **DB Client**

- Other well-known DB clients include:

  PgAdmin — GUI client for PostgreSQL
  DBeaver — Universal DB GUI client
  DataGrip — JetBrains IDE for databases

- All clients connect to the DB server using the same protocol

  TCP/IP + DB driver



Maven Gives Every Java Project a Standard Structure

- Maven is the standard build tool for Java

  It manages dependencies and project structure

- Use IntelliJ IDEA as the recommended IDE

  First-class Maven support out of the box

- The pom.xml file is the heart of every Maven project

  Declares project identity and all dependencies

```
my-app/
├── pom.xml              ← Project info + dependencies
└── src/
    └── main/
        └── java/
            └── com/
                └── example/
                    └── App.java
```

JDBC is the Standard Java API for Database Connectivity

- **JDBC (Java Database Connectivity)** is the official Java API for connecting to relational databases
- Add the MySQL JDBC driver to `pom.xml`

  Maven downloads the driver automatically — no manual JAR management
- JDBC provides a **uniform API** regardless of the underlying database vendor

  Only the connection URL and driver JAR change when switching databases

POM.XML (DEPENDENCY)

```
<dependency>
    <groupId> com.mysql </groupId>
    <artifactId> mysql-connector-j </artifactId>
    <version> 8.3.0 </version>
</dependency>
```

A Connection Object is Your Interface with the Database

Create a connection using DriverManager:

```
Connection conn = DriverManager.getConnection(
"jdbc:mysql://localhost:3306/mydb",
"username",
"password"
);
```

- URL breakdown:

  `jdbc:mysql` → protocol + DB type
  `localhost:3306` → host + port (3306 is MySQL's default)
  `mydb` → the specific database to connect to

- Credentials:

  Credentials are **DB-level** (not application user) credentials.
  They are managed by the DBMS itself.

Statement and ResultSet Power Every Database Query

- Step 1 — Create a Statement:

```
Statement stmt = conn.createStatement();
```

- Step 2 — Execute the query:

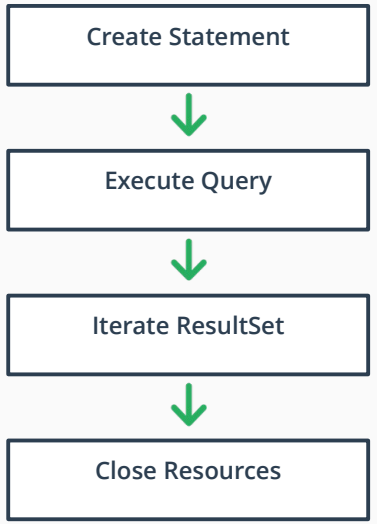
```
ResultSet rs = stmt.executeQuery("SELECT * FROM users");
```

- Step 3 — Iterate the ResultSet:

```
while (rs.next()) {
    System.out.println(rs.getString("name"));
}
```

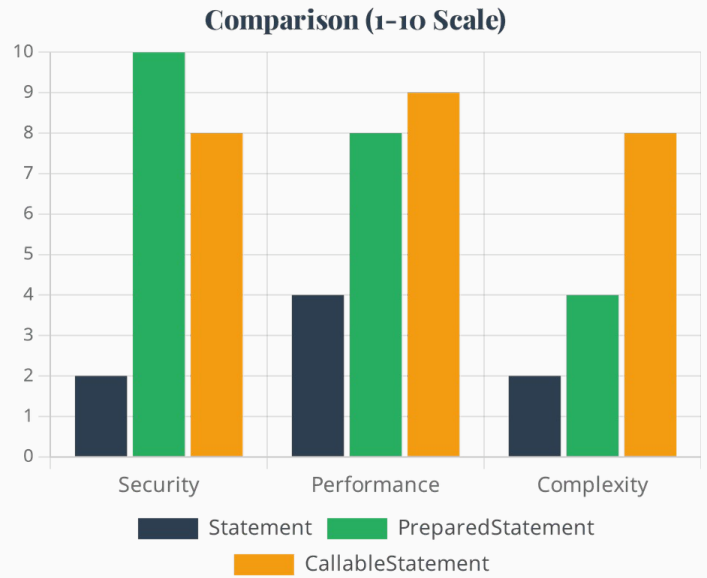
- Step 4 — Close all resources:

```
rs.close();
stmt.close();
conn.close();
```



Choose the Right Statement Type for Every Use Case

Statement Type	Use Case	Key Benefit
Statement	Simple, static SQL	Easy to use
PreparedStatement	Parameterized SQL	Prevents SQL injection
CallableStatement	Stored procedures	Encapsulated DB logic



- Always prefer PreparedStatement for any query with user input

  SQL injection is the #1 database vulnerability — parameterization prevents it

- PreparedStatement is also faster for repeated queries

  Queries are precompiled by the database engine

Spring Provides a "Batteries Included" Backend Framework

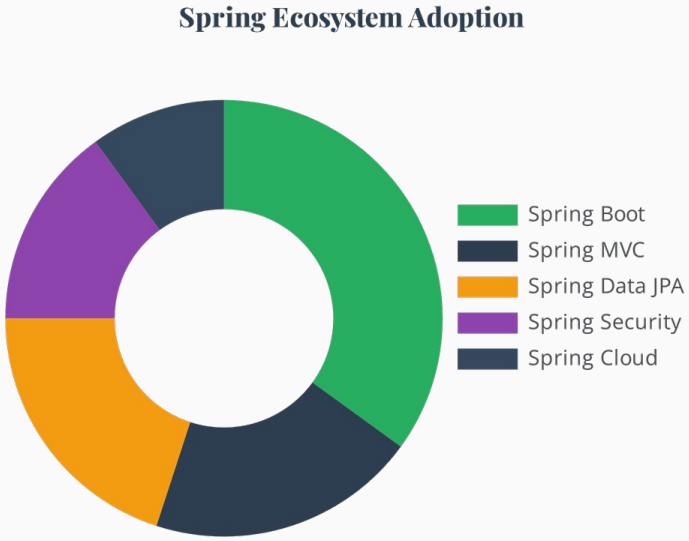
- A **framework** is an opinionated library that enforces a project structure

- Spring's core principles:

  Inversion of Control (IoC) — the framework manages object lifecycles
  Dependency Injection (DI) — components receive their dependencies automatically
  Convention over Configuration — sensible defaults reduce boilerplate

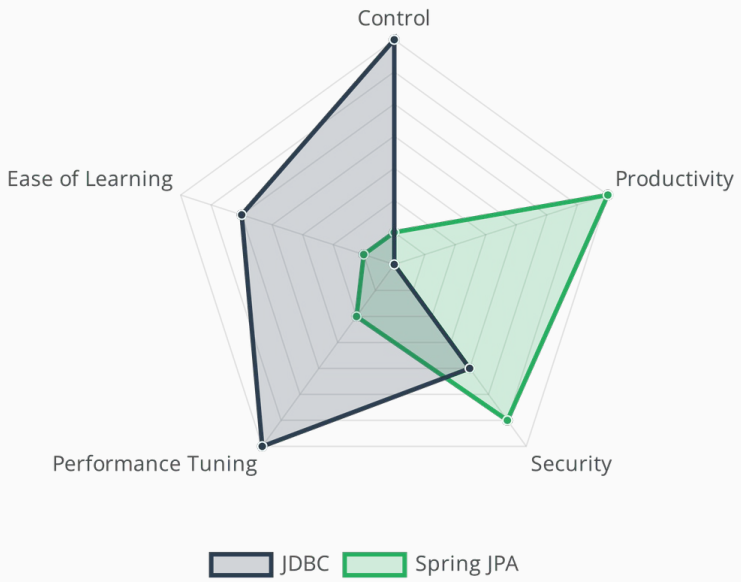
- Spring covers: Web (Spring MVC), Security, Data (JPA), Cloud, Batch, and more

- Used by the majority of enterprise Java backends worldwide



Spring JPA Trades Control for Productivity — Know When to Use Each

Feature	JDBC	Spring JPA (Hibernate)
Level	Low-level	High-level (ORM)
SQL	Manual, explicit	Optional / auto-generated
Boilerplate code	High	Low
Developer control	Full	Abstracted
Learning curve	Easier start	Heavier concepts
Best for	Fine-tuned queries	Rapid CRUD development



- **ORM (Object-Relational Mapping):** Maps Java classes to DB tables automatically
- Spring JPA uses **Hibernate** under the hood
- **Persistence:** Saving data permanently to secondary storage (disk / DB records)