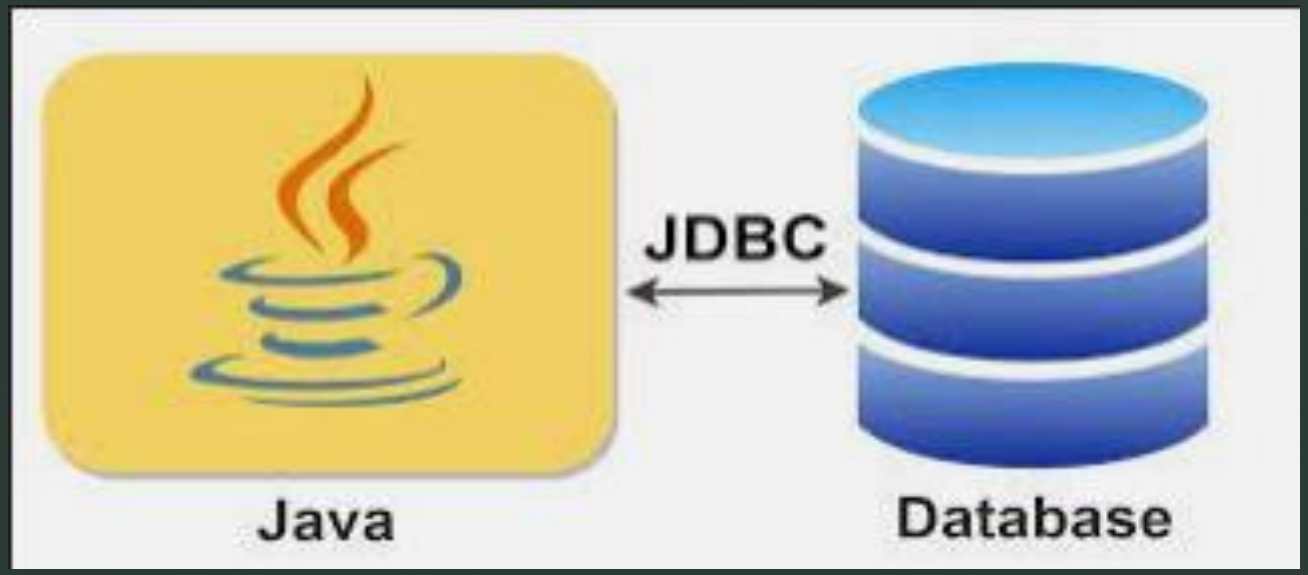
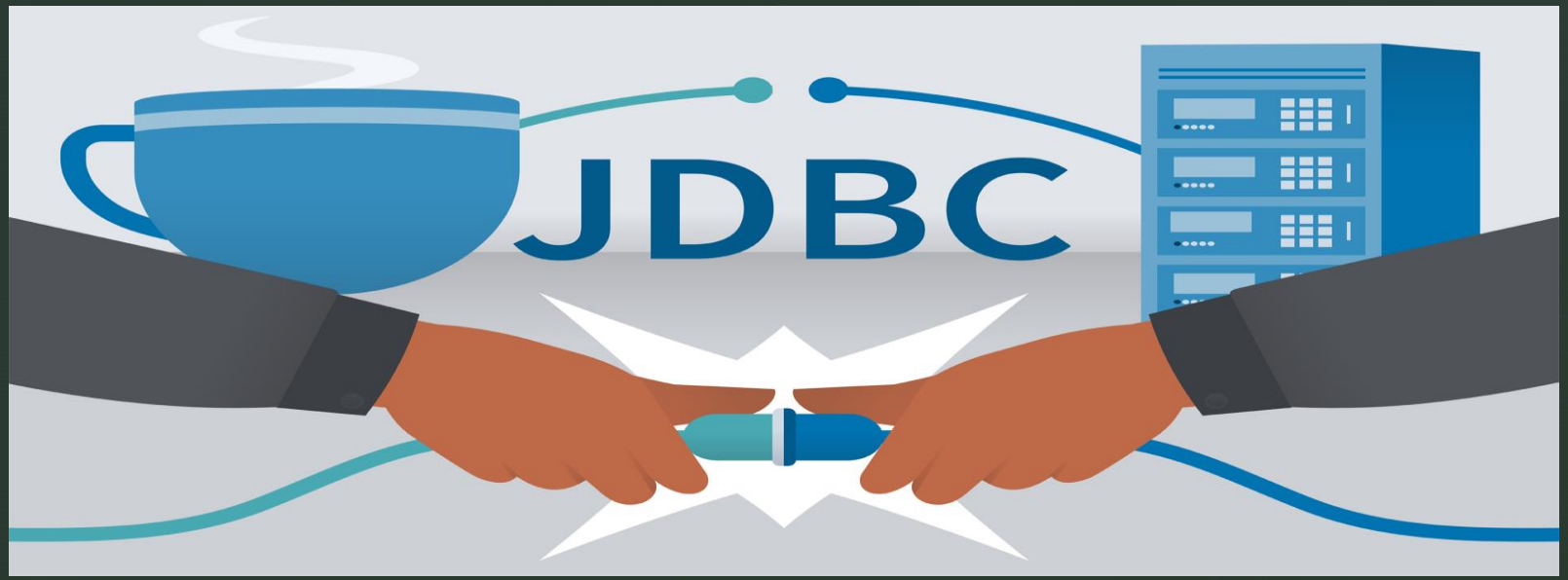
Spring Data Access with JdbcTemplate

Zespół:
M.B, J.M, A.R



Co to JDBC?



Jak powstało JDBC?

JDBC wydano jako część JDK 1.1 w 1997 roku, JDBC był jedną z najwcześniejszych bibliotek opracowanych dla języka Java.

JDBC było początkowo zaprojektowane jako API po stronie klienta, umożliwiające klientowi Javy interakcję ze źródłem danych. Zmieniło się to wraz z wydaniem JDBC 2.0, które zawierało opcjonalny pakiet obsługujący połączenia JDBC po stronie serwera.

JDBC zalety/wady

- Jest w stanie odczytać każdą bazę danych.
- Automatycznie tworzy z bazy danych dane w formacie XML.
- Nie wymaga konwertowania zawartości
- Zapewnia wsparcie zarówno dla przetwarzania synchronicznego jak i asynchronicznego.
- Obsługuje moduły.

- Jest bardzo wrażliwe
- Nie pozwala na pojedynczą sekwencję do aktualizacji lub wstawiania wielu tabel.
- Trudności w obsłudze wyjątków
- Zła wydajność w przypadku wielu połączeń.

Implementacja JDBC

- nawiązanie połączenia z bazą danych
- wysłanie instrukcji SQL,
- przetworzenie wyników

```
Connection con = DriverManager.getConnection (
            "jdbc:odbc:wombat", "login", "password");
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("SELECT a, b, c FROM Table1");
while (rs.next()) {
  int x = getInt("a");
  String s = getString("b");
  float f = getFloat("c");
  }
```

JDBC Template

JdbcTemplateClass

- Główna klasa w paczce JDBC core
- Ułatwia korzystanie z JDBC (nie ma potrzeby tworzyć i kończyć połączenia)

Utworzenie instancji JdbcTemplate

```
private final JdbcTemplate jdbcTemplate;

public VowelCountRepository(JdbcTemplate jdbcTemplate)
{
    this.jdbcTemplate = jdbcTemplate;
}
```

JDBC Template

Konfiguracja DataSource

```
@Configuration
@ComponentScan("com.baeldung.jdbc")
public class SpringJdbcConfig {
    @Bean
    public DataSource mysqlDataSource() {
        DriverManagerDataSource dataSource = new DriverManagerDataSource();
        dataSource.setDriverClassName("com.mysql.jdbc.Driver");
        dataSource.setUrl("jdbc:mysql://localhost:3306/springjdbc");
        dataSource.setUsername("guest_user");
        dataSource.setPassword("guest_password");

        return dataSource;
    }
}
```

```
<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource"
  destroy-method="close">
    <property name="driverClassName" value="com.mysql.jdbc.Driver"/>
    <property name="url" value="jdbc:mysql://localhost:3306/springjdbc"/>
    <property name="username" value="guest_user"/>
    <property name="password" value="guest_password"/>
</bean>
```

JDBC Template

- Wykonanie zapytania

```
SqlRowSet rs =
        jdbcTemplate.queryForRowSet(
                sql: "SELECT * FROM VowelCount");

while (rs.next())
{
    allVowels.replace(
            rs.getString( columnLabel: "Vowel"),
            rs.getInt( columnLabel: "Vcount"));
}
```

- Zmiana danych

```
int numRowsUpdated =
        jdbcTemplate.update(
                sql: "UPDATE VowelCount SET Vcount = Vcount + ? WHERE Vowel = ?",
                entry.getValue(),
                entry.getKey());
```

Lombok

Project Lombok (od teraz Lombok) jest opartą na adnotacjach biblioteką Java, która pozwala na redukcję kodu typu boilerplate. Lombok oferuje różne adnotacje mające na celu zastąpienie kodu w Javie, który jest znany z tego, że jest szablonowy, powtarzalny lub żmudny w pisaniu. Magia dzieje się w czasie kompilacji, kiedy biblioteka wstrzykuje kod bajtowy reprezentujący pożądany i szablonowy kod do plików .class. Ta biblioteka może zostać też użyta w naszych środowiskach.

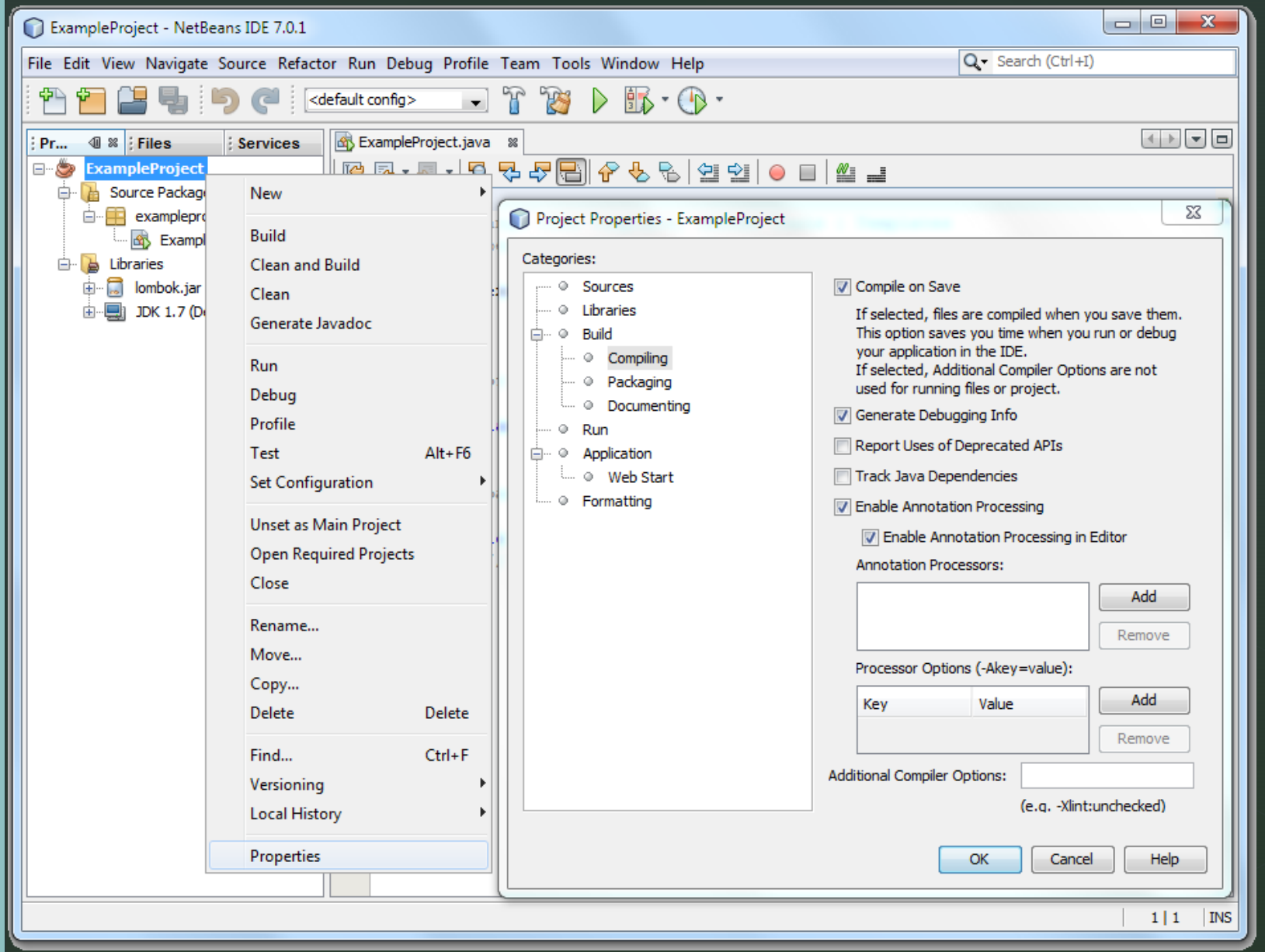
Lombok instalacja

pom.xml + maven-compiler-plugin

```xml
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.20</version>
    <scope>provided</scope>
</dependency>
```

```xml
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-compiler-plugin</artifactId>
<version>3.5.1</version>
<configuration>
    <source>11</source> <!-- depending on your project -->
    <target>11</target> <!-- depending on your project -->
    <annotationProcessorPaths>
        <path>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <version>1.18.20</version>
        </path>
    </annotationProcessorPaths>
</configuration>
</plugin>
```

Lombok instalacja w środowiskach



Netbeans =
biblioteka .jar

Intellij Idea,
VSCode = znajdź
I zainstaluj plugin

Kod w lomboku vs java vanilla

```java
@Getter

@Setter

public class Author {

    private int id;

    private String name;

    @Setter(AccessLevel.PROTECTED)

    private String surname;

}
```

```java
public int getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public String getSurname() {
    return surname;
}

protected void setSurname(String surname) {
    this.surname = surname;
}
```

Lombok wady I zalety użytkowania

```java
@Data
public class Author {
    private final int id;
    private String name;
    private String surname;
}
```

```java
public class Author {
    private final int id;
    private String name;
    private String surname;

    public Author(int id) {
        this.id = id;
    }

    public int getId() {
        return id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getSurname() {
        return surname;
    }

    public void setSurname(String surname) {
        this.surname = surname;
    }

    @Override
    public int hashCode() {
       final int PRIME = 31;
       int result = 1;
       result = prime * result + getId();
       result = prime * result + ((getName() == null) ? 0 : getName().hashCode());
       result = prime * result + ((getSurname() == null) ? 0 : getSurname().hashCode());
       return result;
    }

    @Override
    public boolean equals(Object o) {
       if (o == this) return true;
       if (!(o instanceof Author)) return false;
       Author other = (Author) o;
       if (!other.canEqual((Object)this)) return false;
       if (this.getId() == null ? other.getId() != null : !this.getId().equals(other.getId())) retur
       if (this.getName() == null ? other.getName() != null : !this.getName().equals(other.getName()
       if (this.getSurname() == null ? other.getSurname() != null : !this.getSurname().equals(other.
       return true;
    }
}
```

~~JavaDoc~~

~~Debugging~~