Spring Boot

Data Access - JDBC

Marten Deinum

Spring Boot

Data Access - JDBC: Topics

- Data Access with JDBC
- Spring Framework - JdbcTemplate
- Spring Boot - Auto Configuration
- Spring Boot - Testing
- Spring Boot - Spring Data JDBC

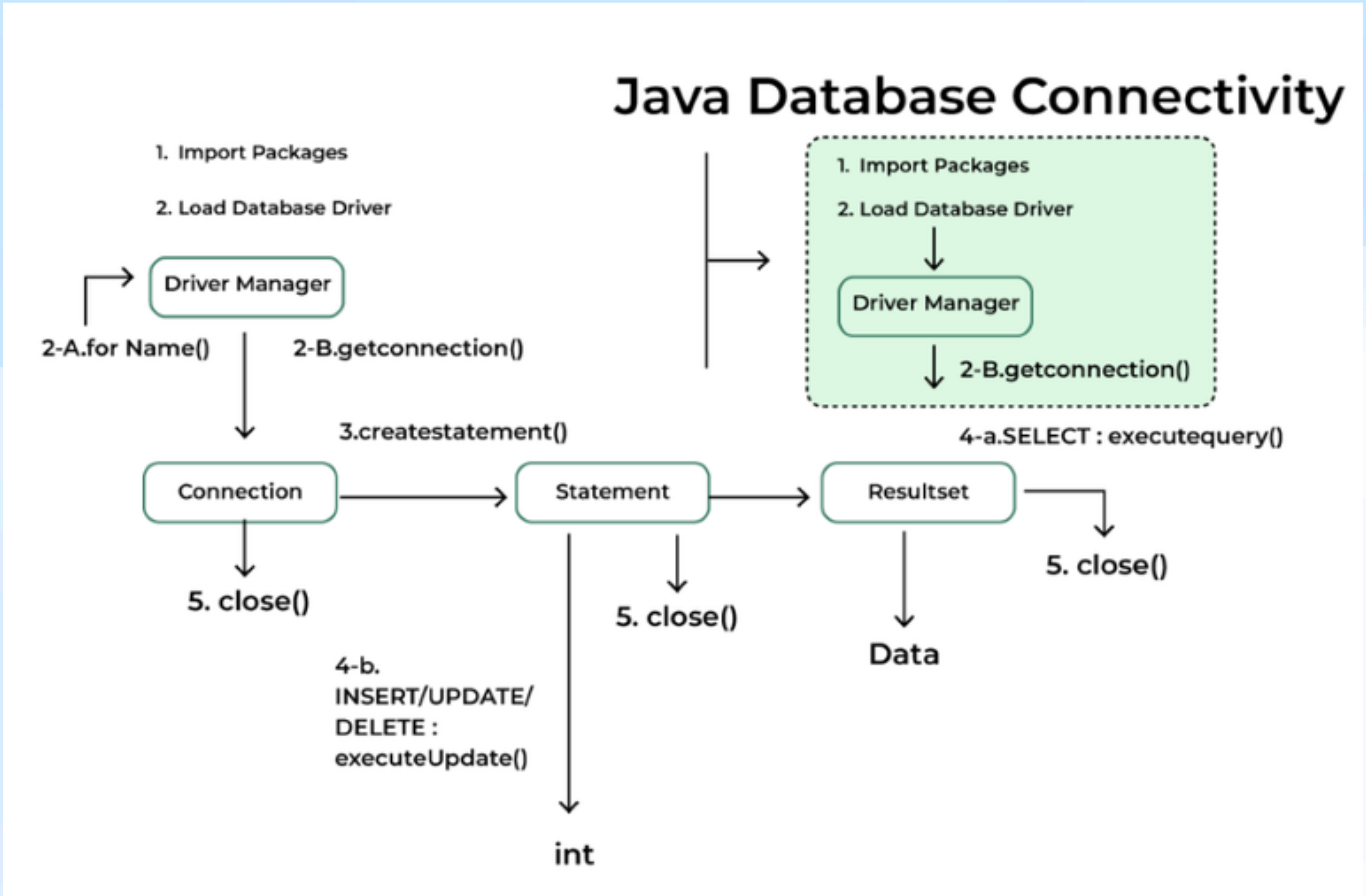
Spring Boot

Data Access with JDBC

- JDBC - Java Database Connectivity
  - Standard API to interfact with databases (Oracle, PostgreSQL, etc.)
- Acts as a bridge between your Java application and the database
- Uses a so-called JDBC driver (different for each database)
- Can be error prone to work with (although try-with-resources helps)

Spring Boot

Data Access with JDBC



Spring Boot

Data Access with JDBC

```
@Override
public Optional<Book> findByIsbn(String isbn) {
  String FIND_BY_ISBN_SQL = "SELECT isbn, title, authors FROM book WHERE isbn = ?";
  try (var conn = dataSource.getConnection();
       var ps = conn.prepareStatement(FIND_BY_ISBN_SQL)) {
    ps.setString(1, isbn);

    try (var rs = ps.executeQuery()) {
      if (rs.next()) {
        return Optional.of(mapToBook(rs));
      }
    }
  } catch (SQLException ex) {
    throw new DataRetrievalFailureException(ex.getMessage(), ex);
  }
  return Optional.empty();
}
```

Spring Boot

Data Access with JDBC

```
@Override
public Optional<Book> findByIsbn(String isbn) {
  String FIND_BY_ISBN_SQL = "SELECT isbn, title, authors FROM book WHERE isbn = ?";
  try (var conn = dataSource.getConnection();
       var ps = conn.prepareStatement(FIND_BY_ISBN_SQL)) {
    ps.setString(1, isbn);

    try (var rs = ps.executeQuery()) {
      if (rs.next()) {
        return Optional.of(mapToBook(rs));
      }
    }
  } catch (SQLException ex) {
    throw new DataRetrievalFailureException(ex.getMessage(), ex);
  }
  return Optional.empty();
}
```

The bold part is the part you should be writing.

What to do with SQL Exceptions?

Spring Boot
JdbcTemplate

Spring Boot

JdbcTemplate

- Simplifies the use of the JDBC API
    - Eliminates boilerplate
    - SQL Exception translation
- Can still access the underlying JDBC API if needed

Spring Boot

Data Access with JdbcTemplate

```
@Override
public Optional<Book> findByIsbn(String isbn) {
  String FIND_BY_ISBN_SQL = "SELECT isbn, title, authors FROM book WHERE isbn = ?”;
  try {
    var dbResult = this.jdbcTemplate.queryForObject(FIND_BY_ISBN_SQL, (rs, c) →
mapToBook(rs), isbn);
    return Optional.of(dbResult);
  } catch (IncorrectResultSetDataAccessException ex) {
    return Optional.empty();
  }
 }
```

- Get the connection
- Create and execute statement
- process the result
- handle exceptions
- Close used resources (if needed)

Spring Boot

Data Access with JdbcTemplate

- `JdbcTemplate` still requires a `DataSource`!
- Create a template once and re-use it
    - it is thread safe after it has been constructed
    - Quite heavy to construct
- When to use
    - When you are using JDBC
    - Clean up legacy code
    - Testing

Spring Boot

Data Access with JdbcClient

- Simplifies the use of the JDBC API
  - Eliminates boilerplate
  - SQL Exception translation
- Can still access the underlying JDBC API if needed
- More of a fluent API

Spring Boot

Data Access with JdbcClient

```java
@Override
public Optional<Book> findByIsbn(String isbn) {
  String FIND_BY_ISBN_SQL = "SELECT isbn, title, authors FROM book WHERE isbn = ?”;
  return this.jdbcClient.sql(FIND_BY_ISBN_SQL)
      .param(isbn)
      .query((rs, c) → mapToBook(rs))
      .optional();
}
```

- Get the connection
- Create and execute statement
- process the result
- handle exceptions
- Close used resources (if needed)

Spring Boot

Auto-Configuration

Spring Boot

Auto-Configuration

- Spring Boot will automatically create a DataSource
    - When JDBC driver is detected
    - Appropriate `spring.datasource` properties have been set (if needed)
- Will create a `JdbcTemplate` and `JdbcClient` when on classpath
- Setup transaction management to work with JDBC
    - Use `@Transactional` in your code

Spring Boot

Auto-Configuration

- Schema management with Liquibase or Flyway
- Simple schema management through `schema.sql` and `data.sql`
  - Automatically applied when embedded database is detected (H2, HSQLDB or Apache Derby)
  - Explicitly needs to be enabled for “real” databases
-

Spring Boot

Excercise:

31-spring-boot-jdbc

Spring Boot

Testing JDBC

Spring Boot

Data Access with JDBC - Testing

- Spring Boot provides extensive support for testing JDBC
- Use `@JdbcTest` to quickly be able to test JDBC code
    - Will auto-configure a minimal application context with only JDBC related setup and classes
    - Will replace the real database for an in-memory one
    - Setup transactions for tests
- Will also work with Docker through TestContainers support
    - Auto start/stop db container for tests with setup

Spring Boot

Data Access with JDBC - Testing

- When using Spring Data repositories use `@DataJdbcTest`
  - This will also setup Spring Data JDBC repositories

Spring Boot

Spring Data JDBC

“Spring Data’s mission is to provide a familiar and consistent, Spring-based programming model for data access while still retaining the special traits of the underlying data store.

It makes it easy to use data access technologies, relational and non-relational databases, map-reduce frameworks, and cloud-based data services. This is an umbrella project which contains many subprojects that are specific to a given database. The projects are developed by working together with many of the companies and developers that are behind these exciting technologies.
”

Spring Data - Reference Guide

Spring Boot

Spring Data JDBC

- Support for many different data stores
  - JDBC
  - Apache Cassandra
  - Redis
  - LDAP
  - JPA
  - Neo4J
- Reduces even more “boilerplate” code
  - By some considered magic (but it is actual clever use of proxies)

Spring Boot

Spring Data JDBC

- Declarative repositores
    - Implementation “generated” at runtime
    - Uses AOP
    - Derived queries from method names
    - Explicit queries through `@Query`
    - Still extensible through fragments
- Templates (akin `JdbcTemplate`) for most technologies (`CassandraTemplate`, `LdapTemplate` etc)

Spring Boot

Spring Data JDBC

```java
public interface JdbcBookRepository extends ListCrudRepository<Book, Long> {

 Optional<Book> findByIsbn(String isbn);
}
```

- Provides default method for `save`, `findById`, `deleteById` etc.
- Creates dynamic call for `findByIsbn` based on the class and mappings
- Paging support available (if supported by underlying datastore)

Spring Boot

Spring Data JDBC - Spring Boot

- Spring Boot detects presence of Spring Data projects
  - Will setup detected projects (JDBC, JPA etc.)
  - Will setup scanning and creation of runtime repositories
- Might need some help if multiple technologies are being used
  - Cannot always detect which repository is for which technology

Spring Boot

Excercise (cont.):
31-spring-boot-jdbc