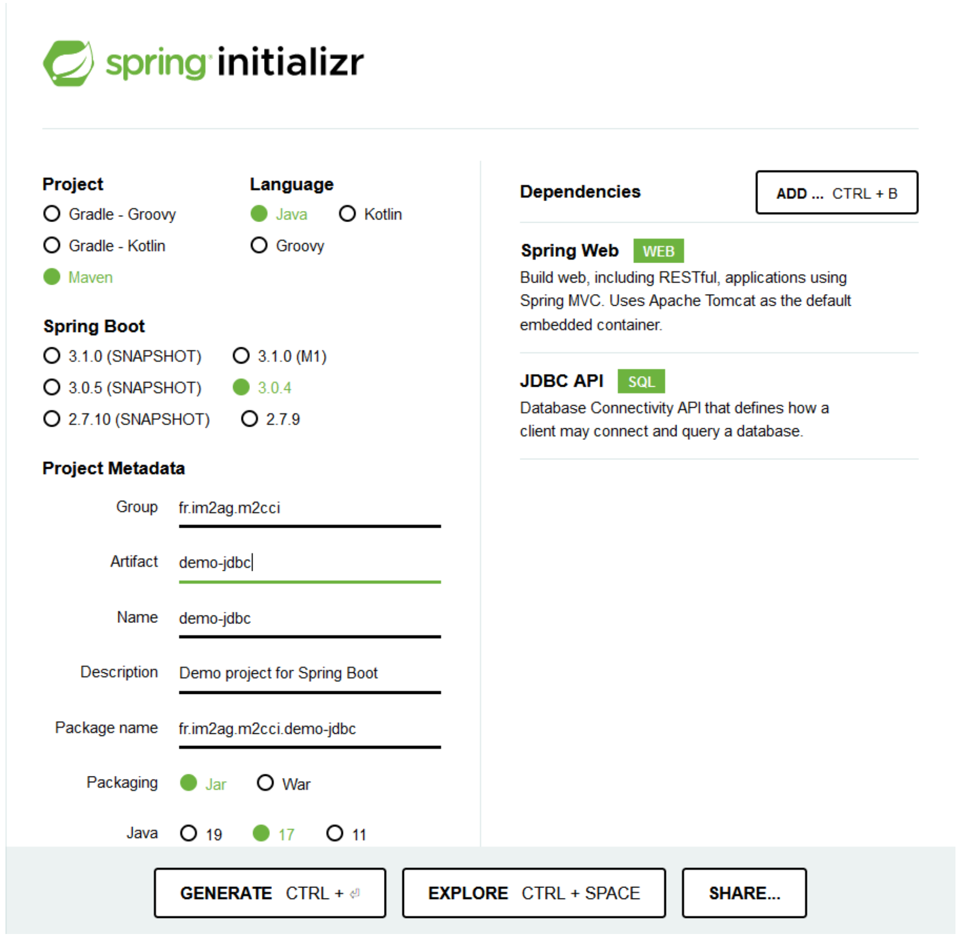
POC : Projet Intégration

Spring Boot JDBC

- avec Spring Boot initializer
  - https://start.spring.io/



spring initializr

Project
- ○ Gradle - Groovy
- ○ Gradle - Kotlin
- ● Maven

Language
- ● Java
- ○ Kotlin
- ○ Groovy

Spring Boot
- ○ 3.1.0 (SNAPSHOT)
- ○ 3.1.0 (M1)
- ○ 3.0.5 (SNAPSHOT)
- ● 3.0.4
- ○ 2.7.10 (SNAPSHOT)
- ○ 2.7.9

Project Metadata

Group	fr.im2ag.m2cci
Artifact	demo-jdbc
Name	demo-jdbc
Description	Demo project for Spring Boot
Package name	fr.im2ag.m2cci.demo-jdbc
Packaging	● Jar ○ War
Java	○ 19 ● 17 ○ 11

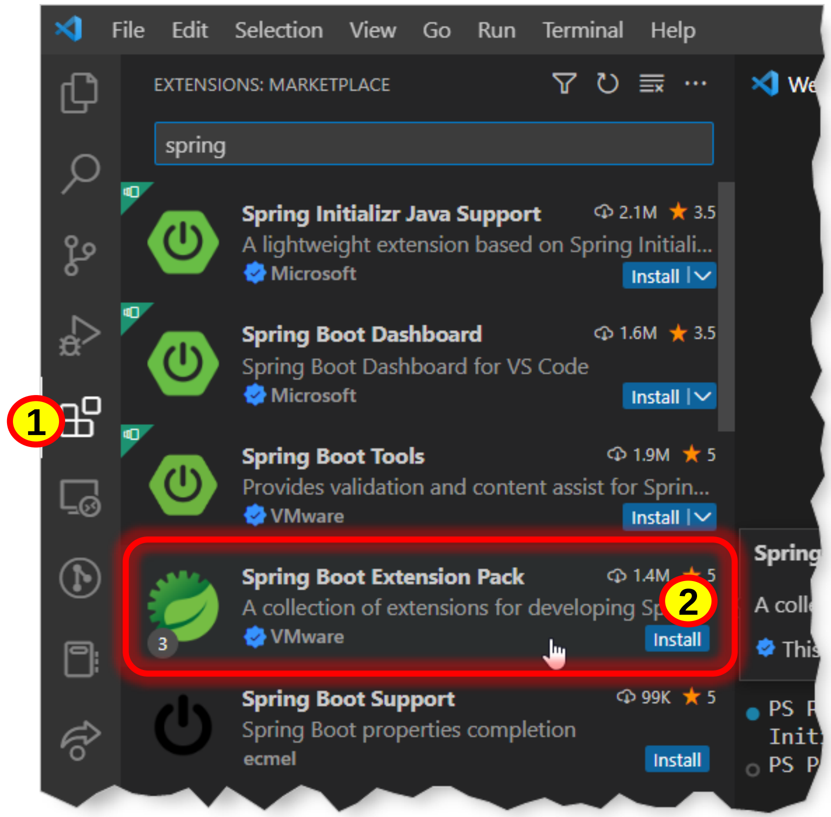
Dependencies ADD ... CTRL + B

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

JDBC API SQL
Database Connectivity API that defines how a client may connect and query a database.

GENERATE CTRL + ⏎ | EXPLORE CTRL + SPACE | SHARE...

- dans VSCode : extension SpringBoot



Spring Initializr Java Support — 2.1M ★ 3.5
A lightweight extension based on Spring Initiali...
Microsoft — Install

Spring Boot Dashboard — 1.6M ★ 3.5
Spring Boot Dashboard for VS Code
Microsoft — Install

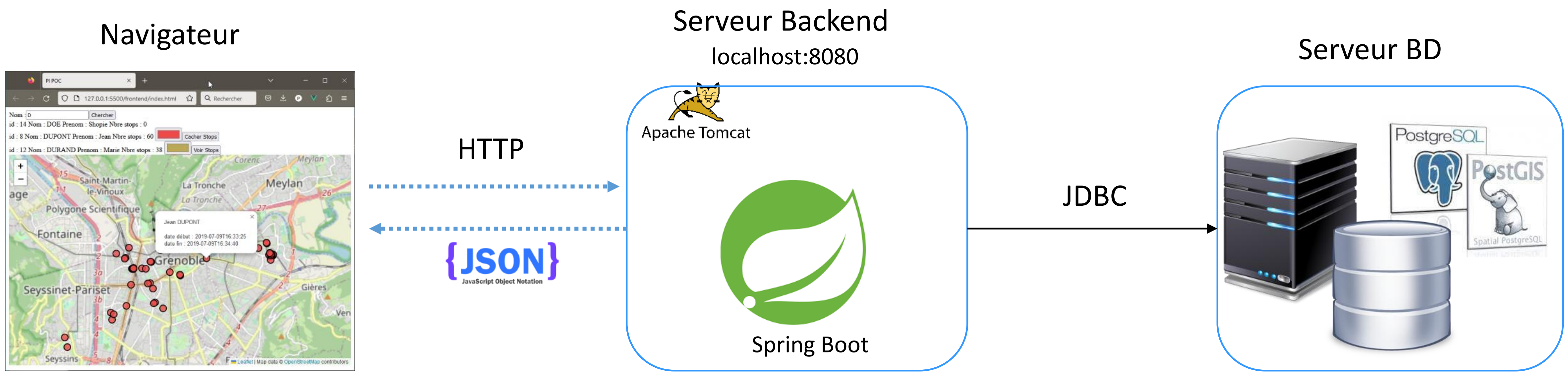
Spring Boot Tools — 1.9M ★ 5
Provides validation and content assist for Sprin...
VMware — Install

Spring Boot Extension Pack — 1.4M ★ 5
A collection of extensions for developing Sp
VMware — Install

Spring Boot Support — 99K ★ 5
Spring Boot properties completion
ecmel — Install

POC : Projet Intégration

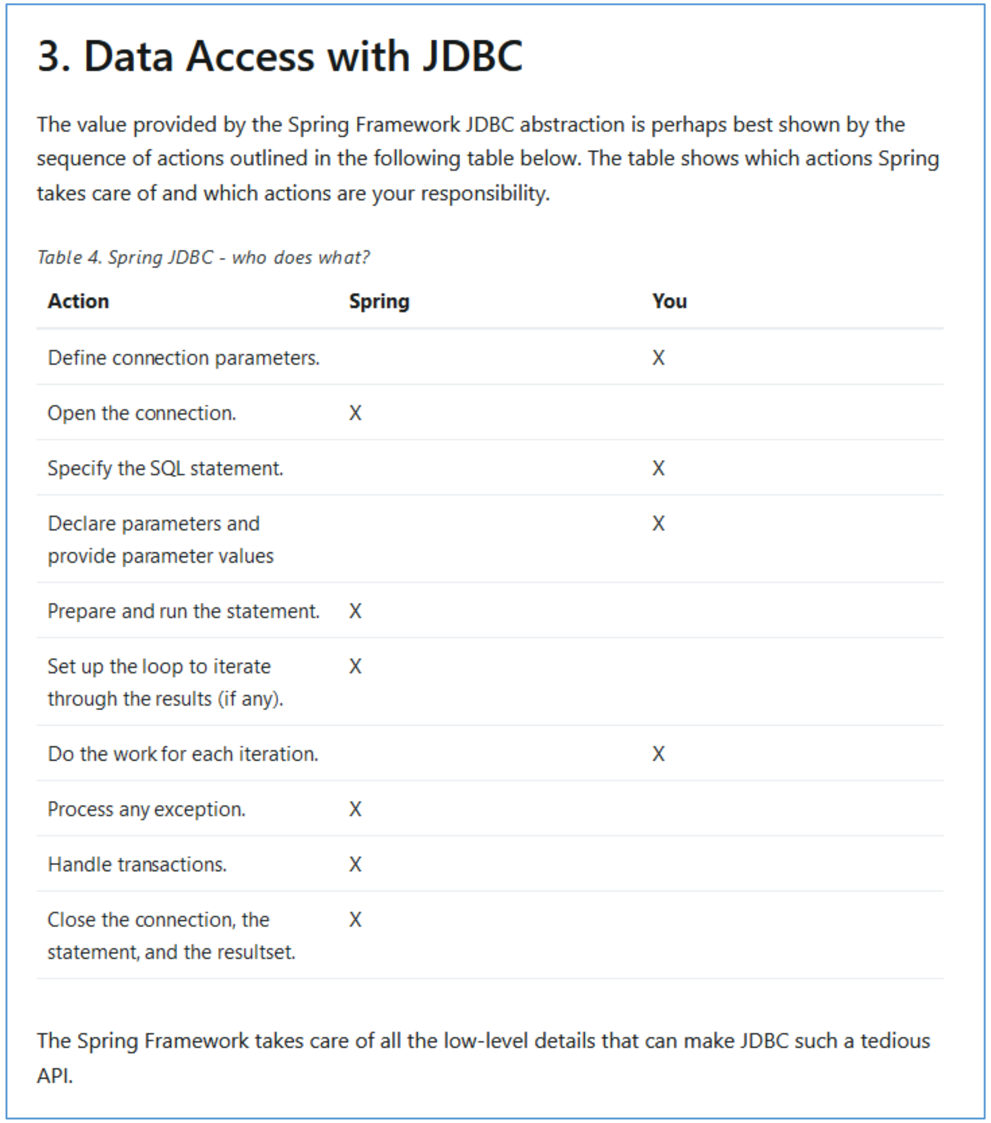
- https://gricad-gitlab.univ-grenoble-alpes.fr/enseignement1/m2cci/projet-integration/pi-poc


Navigateur
PI POC
127.0.0.1:5500/frontend/index.html
Rechercher
Nom : D
Chercher
id : 14 Nom : DOE Prenom : Shopie Nbre stops : 0
id : 8 Nom : DUPONT Prenom : Jean Nbre stops : 60
Cacher Stops
id : 12 Nom : DURAND Prenom : Marie Nbre stops : 38
Voir Stops
Jean DUPONT
date début : 2019-07-09T16:33:25
date fin : 2019-07-09T16:34:40
HTTP
{JSON}
JavaScript Object Notation
Serveur Backend
localhost:8080
Apache Tomcat
Spring Boot
JDBC
Serveur BD
PostgreSQL
PostGIS
Spatial PostgreSQL

POC : Projet Intégration

Spring boot : JDBC

- documentation : https://docs.spring.io/spring-framework/docs/current/reference/html/data-access.html#jdbc



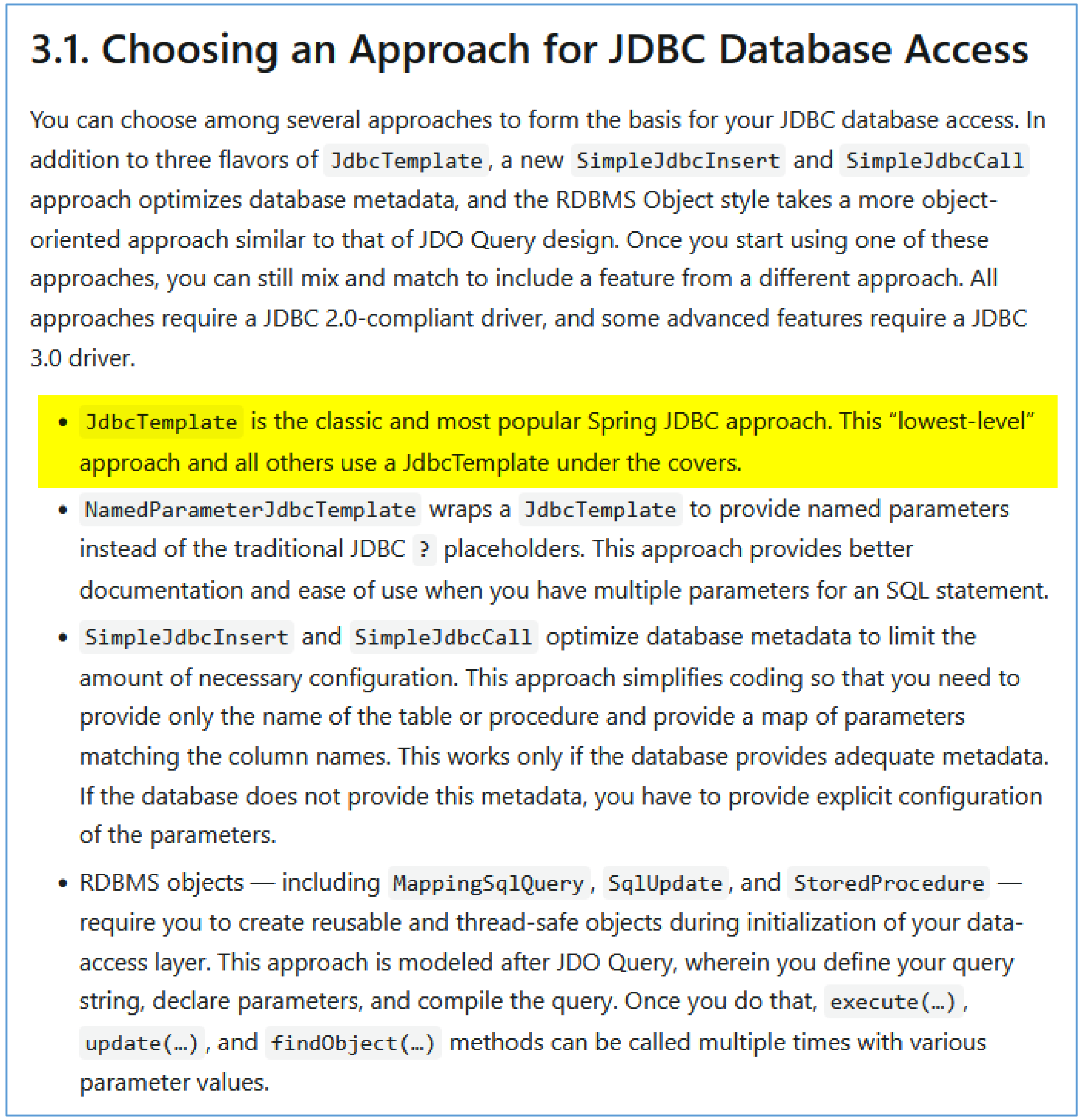
3. Data Access with JDBC

The value provided by the Spring Framework JDBC abstraction is perhaps best shown by the sequence of actions outlined in the following table below. The table shows which actions Spring takes care of and which actions are your responsibility.

Table 4. Spring JDBC - who does what?

Action	Spring	You
Define connection parameters.		X
Open the connection.	X	
Specify the SQL statement.		X
Declare parameters and provide parameter values		X
Prepare and run the statement.	X	
Set up the loop to iterate through the results (if any).	X	
Do the work for each iteration.		X
Process any exception.	X	
Handle transactions.	X	
Close the connection, the statement, and the resultset.	X	

The Spring Framework takes care of all the low-level details that can make JDBC such a tedious API.



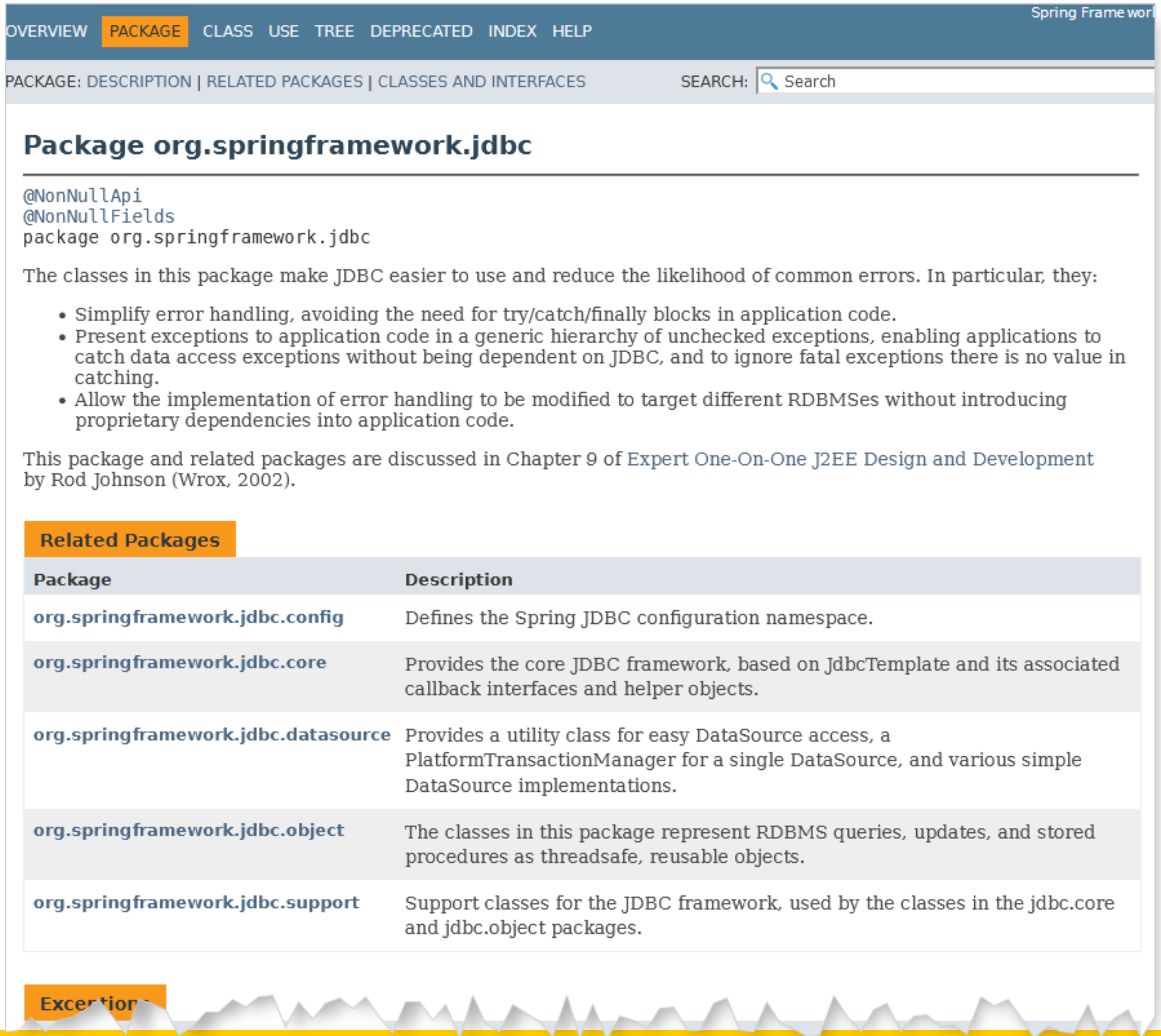
3.1. Choosing an Approach for JDBC Database Access

You can choose among several approaches to form the basis for your JDBC database access. In addition to three flavors of `JdbcTemplate`, a new `SimpleJdbcInsert` and `SimpleJdbcCall` approach optimizes database metadata, and the RDBMS Object style takes a more object-oriented approach similar to that of JDO Query design. Once you start using one of these approaches, you can still mix and match to include a feature from a different approach. All approaches require a JDBC 2.0-compliant driver, and some advanced features require a JDBC 3.0 driver.

- `JdbcTemplate` is the classic and most popular Spring JDBC approach. This "lowest-level" approach and all others use a JdbcTemplate under the covers.
- `NamedParameterJdbcTemplate` wraps a `JdbcTemplate` to provide named parameters instead of the traditional JDBC `?` placeholders. This approach provides better documentation and ease of use when you have multiple parameters for an SQL statement.
- `SimpleJdbcInsert` and `SimpleJdbcCall` optimize database metadata to limit the amount of necessary configuration. This approach simplifies coding so that you need to provide only the name of the table or procedure and provide a map of parameters matching the column names. This works only if the database provides adequate metadata. If the database does not provide this metadata, you have to provide explicit configuration of the parameters.
- RDBMS objects — including `MappingSqlQuery`, `SqlUpdate`, and `StoredProcedure` — require you to create reusable and thread-safe objects during initialization of your data-access layer. This approach is modeled after JDO Query, wherein you define your query string, declare parameters, and compile the query. Once you do that, `execute(…)`, `update(…)`, and `findObject(…)` methods can be called multiple times with various parameter values.

POC : Projet Intégration SpringBoot JDBC

- Packages https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/jdbc/package-summary.html



OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP
Spring Framework
PACKAGE: DESCRIPTION | RELATED PACKAGES | CLASSES AND INTERFACES
SEARCH: Search

Package org.springframework.jdbc

```
@NonNullApi
@NonNullFields
package org.springframework.jdbc
```

The classes in this package make JDBC easier to use and reduce the likelihood of common errors. In particular, they:

- Simplify error handling, avoiding the need for try/catch/finally blocks in application code.
- Present exceptions to application code in a generic hierarchy of unchecked exceptions, enabling applications to catch data access exceptions without being dependent on JDBC, and to ignore fatal exceptions there is no value in catching.
- Allow the implementation of error handling to be modified to target different RDBMSes without introducing proprietary dependencies into application code.

This package and related packages are discussed in Chapter 9 of Expert One-On-One J2EE Design and Development by Rod Johnson (Wrox, 2002).

Related Packages

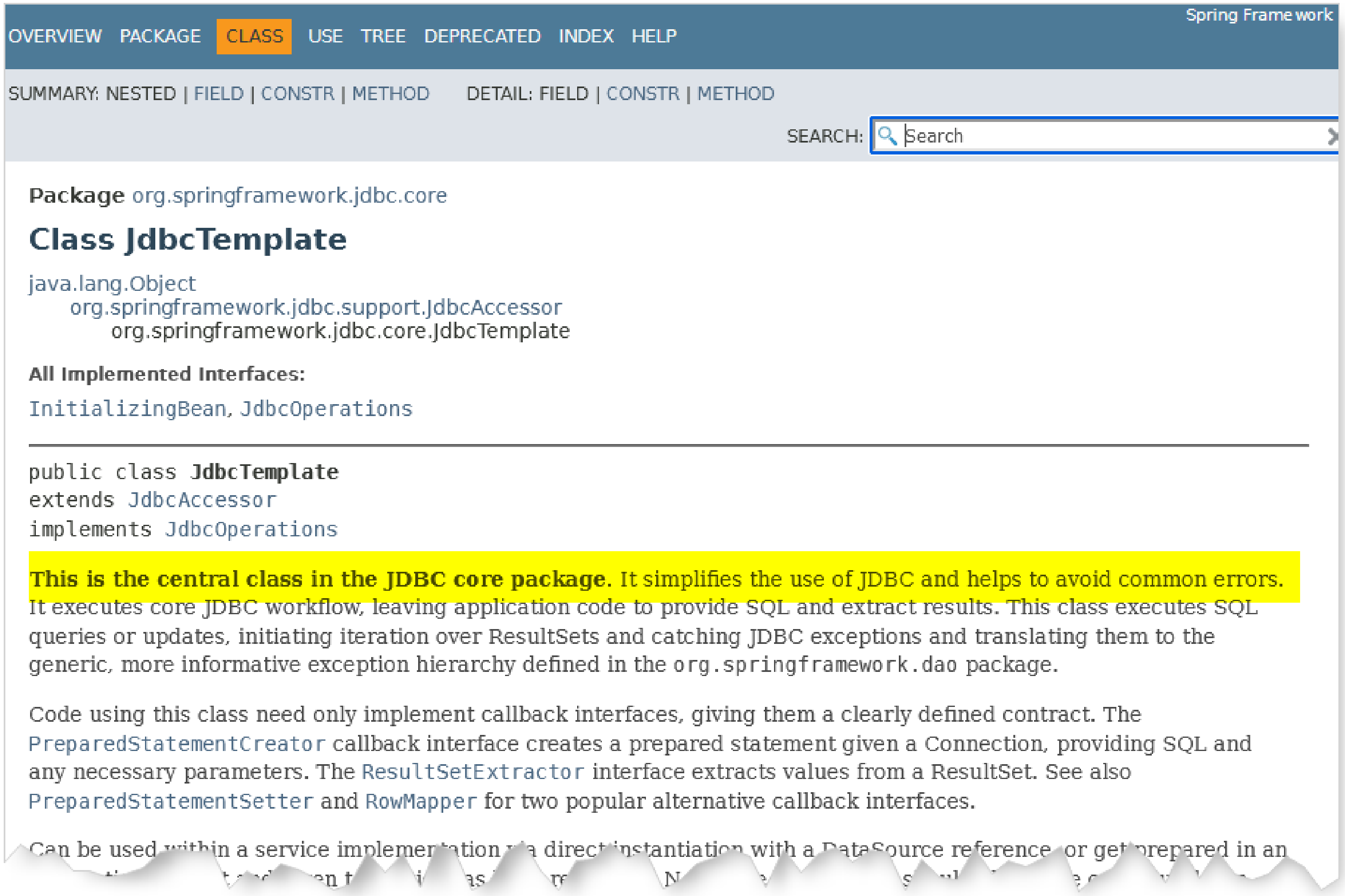
Package	Description
org.springframework.jdbc.config	Defines the Spring JDBC configuration namespace.
org.springframework.jdbc.core	Provides the core JDBC framework, based on JdbcTemplate and its associated callback interfaces and helper objects.
org.springframework.jdbc.datasource	Provides a utility class for easy DataSource access, a PlatformTransactionManager for a single DataSource, and various simple DataSource implementations.
org.springframework.jdbc.object	The classes in this package represent RDBMS queries, updates, and stored procedures as threadsafe, reusable objects.
org.springframework.jdbc.support	Support classes for the JDBC framework, used by the classes in the jdbc.core and jdbc.object packages.

Exception

POC : Projet Intégration SpringBoot JDBC

- JDBC Template

https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/jdbc/core/JdbcTemplate.html


Spring Framework
OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD
SEARCH: Search

Package org.springframework.jdbc.core

Class JdbcTemplate

java.lang.Object
    org.springframework.jdbc.support.JdbcAccessor
        org.springframework.jdbc.core.JdbcTemplate

All Implemented Interfaces:

`InitializingBean`, `JdbcOperations`

```
public class JdbcTemplate
extends JdbcAccessor
implements JdbcOperations
```

This is the central class in the JDBC core package. It simplifies the use of JDBC and helps to avoid common errors. It executes core JDBC workflow, leaving application code to provide SQL and extract results. This class executes SQL queries or updates, initiating iteration over ResultSets and catching JDBC exceptions and translating them to the generic, more informative exception hierarchy defined in the `org.springframework.dao` package.

Code using this class need only implement callback interfaces, giving them a clearly defined contract. The `PreparedStatementCreator` callback interface creates a prepared statement given a Connection, providing SQL and any necessary parameters. The `ResultSetExtractor` interface extracts values from a ResultSet. See also `PreparedStatementSetter` and `RowMapper` for two popular alternative callback interfaces.

Can be used within a service implementation via direct instantiation with a DataSource reference, or get prepared in an