# Spring Data Repositories Best Practices

Oliver Gierke & Thomas Darimont

## Oliver Gierke

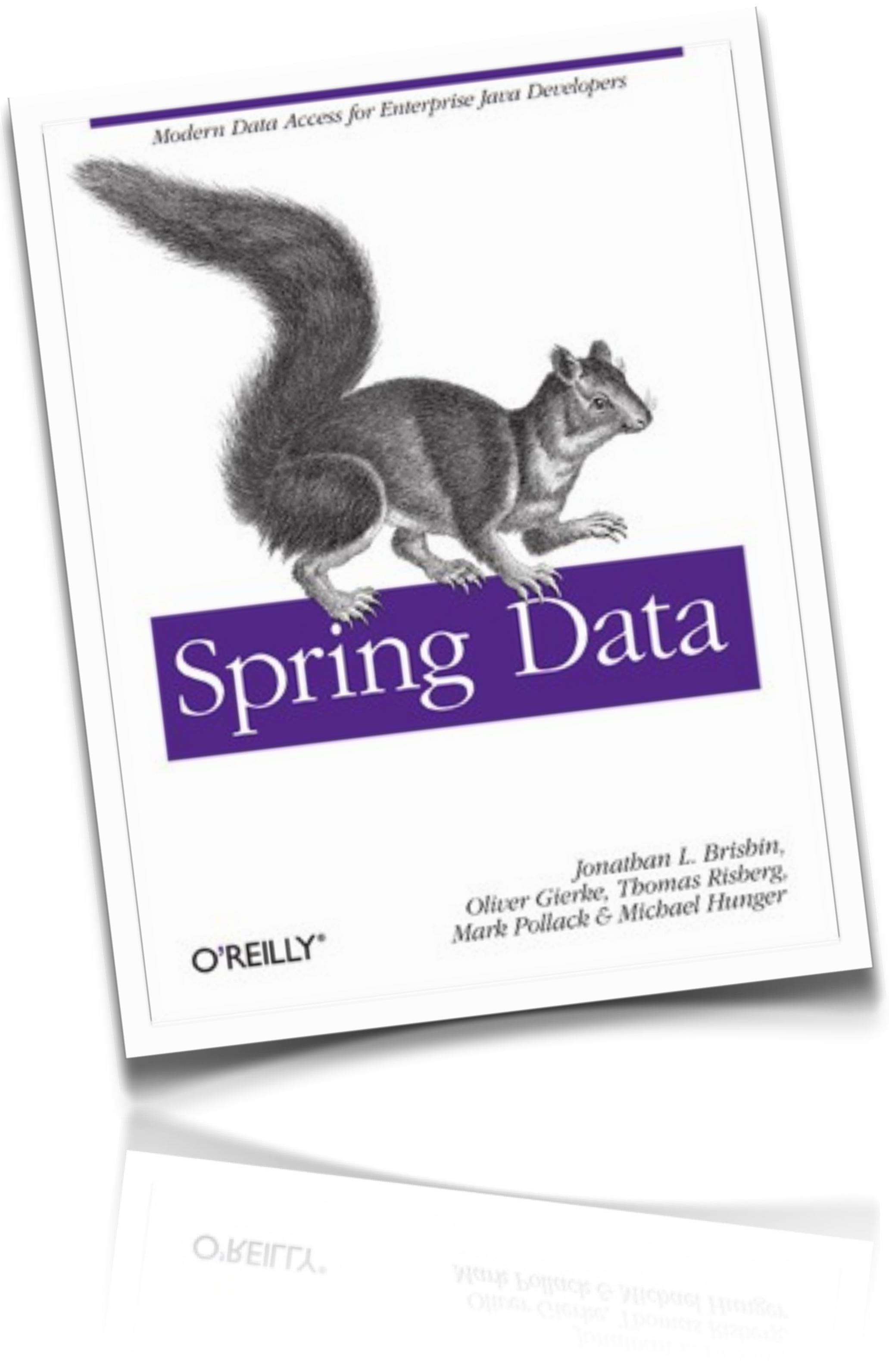
Spring Data Engineer
Project lead Core/JPA/MongoDB

ogierke@gopivotal.com
www.olivergierke.de
olivergierke

Thomas Darimont

Spring Data Engineer
Core/JPA/MongoDB

tdarimont@gopivotal.com
www.tutorials.de
thomasdarimont

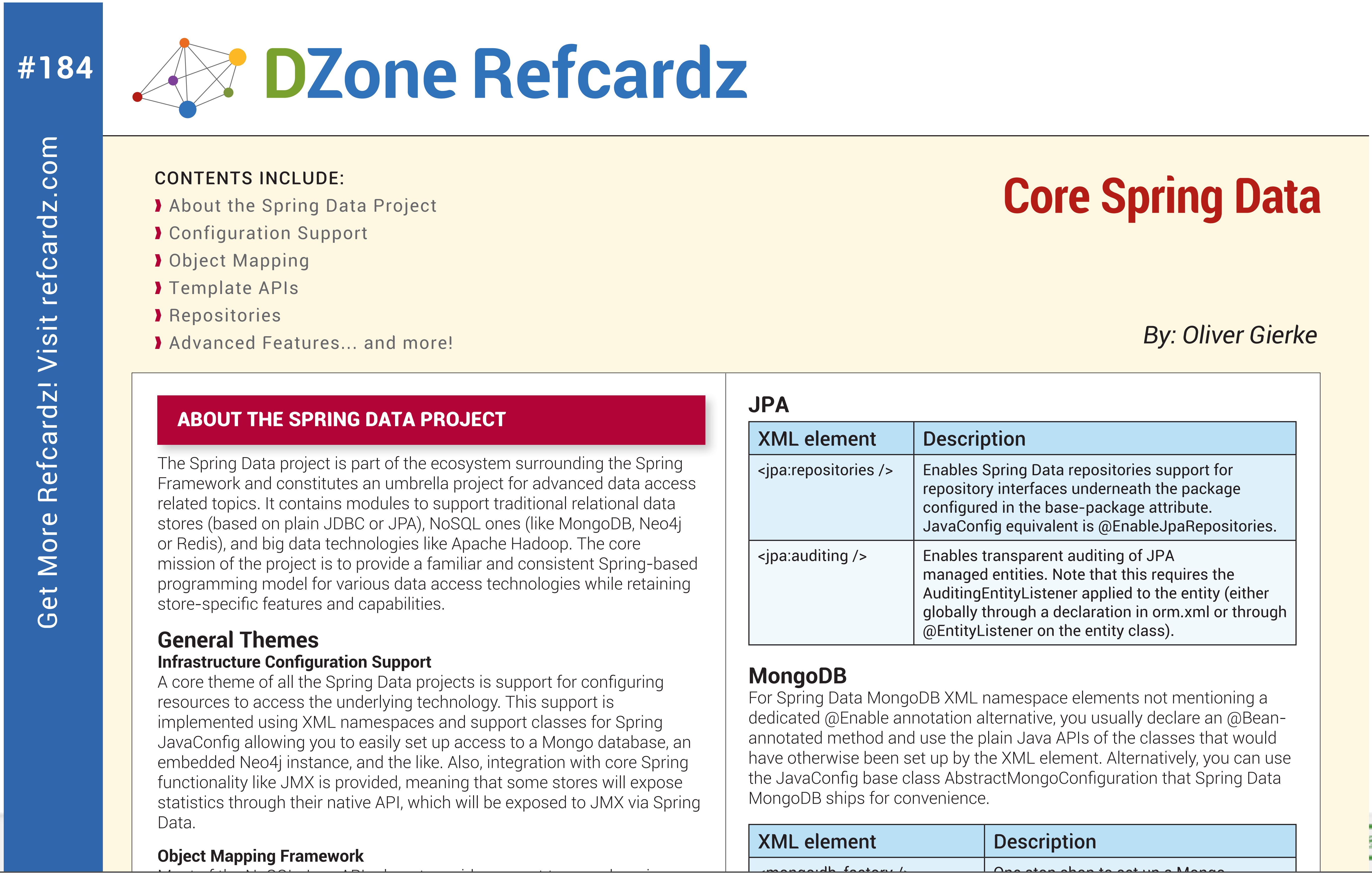

Modern Data Access for Enterprise Java Developers
Spring Data
Jonathan L. Brisbin, Oliver Gierke, Thomas Risberg, Mark Pollack & Michael Hunger
O'REILLY

## Spring Data

Modern Data Access For Enterprise Java

JDBC Hive HBase JPA Pig Splunk Big Data NoSQL Hadoop Redis Roo Gemfire MongoDB Neo4j REST exporter Querydsl Repositories

**1 free copy per attendee!**



#184

Get More Refcardz! Visit refcardz.com

DZone Refcardz

# Core Spring Data

By: Oliver Gierke

CONTENTS INCLUDE:

## ABOUT THE SPRING DATA PROJECT

The Spring Data project is part of the ecosystem surrounding the Spring Framework and constitutes an umbrella project for advanced data access related topics. It contains modules to support traditional relational data stores (based on plain JDBC or JPA), NoSQL ones (like MongoDB, Neo4j or Redis), and big data technologies like Apache Hadoop. The core mission of the project is to provide a familiar and consistent Spring-based programming model for various data access technologies while retaining store-specific features and capabilities.

### General Themes

**Infrastructure Configuration Support**

A core theme of all the Spring Data projects is support for configuring resources to access the underlying technology. This support is implemented using XML namespaces and support classes for Spring JavaConfig allowing you to easily set up access to a Mongo database, an embedded Neo4j instance, and the like. Also, integration with core Spring functionality like JMX is provided, meaning that some stores will expose statistics through their native API, which will be exposed to JMX via Spring Data.

**Object Mapping Framework**

### JPA

| XML element | Description |
|---|---|
| <jpa:repositories /> | Enables Spring Data repositories support for repository interfaces underneath the package configured in the base-package attribute. JavaConfig equivalent is @EnableJpaRepositories. |
| <jpa:auditing /> | Enables transparent auditing of JPA managed entities. Note that this requires the AuditingEntityListener applied to the entity (either globally through a declaration in orm.xml or through @EntityListener on the entity class). |

### MongoDB

For Spring Data MongoDB XML namespace elements not mentioning a dedicated @Enable annotation alternative, you usually declare an @Bean-annotated method and use the plain Java APIs of the classes that would have otherwise been set up by the XML element. Alternatively, you can use the JavaConfig base class AbstractMongoConfiguration that Spring Data MongoDB ships for convenience.

| XML element | Description |
|---|---|
| <mongo:db-factory /> | One-stop shop to set up a Mongo |

## Hands on

## Sample code

https://github.com/olivergierke/repositories-deepdive

## Step 0

Initial project setup

" *How to get a Spring Data JPA based project up and running quickly?*

## Summary

Check out Spring Boot

Easily configure dependencies for your project

Defaults application config based on classpath

# Step 1

Basic JPA infrastructure setup

" *Persistence technology of choice is* ***JPA****. The application uses* ***JavaConfig*** *and sample data contained in* ***data.sql****.*

## Summary

Easy setup through JavaConfig

XML-less JPA configuration

## Step 2

Quickstart

"*The implementation of the persistence layer will be based on the* ***Spring Data repositories*** *abstraction. Customers can be* ***saved, looked up*** ***by*** *their* ***id, email address.***

## Summary

Interface-based programming model

No implementation required

Queries derived from method names

# Step 3

Extended CRUD methods

" *Customers can be **deleted** and **obtained all at once.***

## Summary

Switched to CrudRepository

Exposed CRUD methods

Broad API exposed

## Step 4

Pagination

" *Customers can be accessed* ***page by page.***

## Summary

Switched to PagingAndSortingRepository

Exposed CRUD methods and paging ones

Broad API exposed

## Step 5

Re-declaring existing CRUD methods

"*CustomerRepository.findAll() should rather return a **List**. The transaction timeout for **save(…)** should be customized to **10 seconds**.*

# Summary

Re-declare methods to customize

Return types

Annotation config (Tx, Locking, Query, Hints)

## Step 6

Introducing a read-only repository base interface

" *Products shall be accessible in **read-only** mode only.*

## Summary

Craft custom base interface

Return types

Narrow down the API to the necessary parts

## Step 7

Using manually defined queries

" *As a user, I want to look up* ***products*** *by their* ***custom attributes.***

## Summary

@Query annotation

JPA named queries

Spring Data named queries

## Step 8

Flexible predicate execution

" *As a user, I want to search for customers by* ***first name, last name, email address*** *and* ***any combination*** *of them*

# Summary

Querydsl - type safe queries for Java

QuerydslPredicateExecutor

## Side track:
## Repository proxies

# Proxy mechanism

Using Spring's JDK proxy support

Bootstrap through Factory

Spring FactroyBean / CDI

XML namespace / @EnableRepositories

# Proxy mechanism

Query interface

CRUD implementation class

QueryDslPredicateExecutor

Custom implementation extension

## Step 9

Custom implementations for  repositories

" *As an admin user, I'd like to use custom code to* ***delete all products*** *beyond a given* ***price.***

## Summary

Provide custom implementation

Base class support (Querydsl)

## Stuff on top

Spring MVC integration

Spring Data REST

Spring Boot

# Up and beyond

# Supported modules

JPA

MongoDB

Neo4j

Gemfire

## Community modules

Solr

Elasticsearch

Couchbase

FuzzyDB

(Cassandra)

## Further sessions

Tackling Big Data Complexity with Spring - Wed, 2:30pm

Your Data, Your Search, Elasticsearch - Wed, 2:30pm

Spring Data Community Lightning Talks - Thu, 8:30am

Researching Cancer in the Cloud with Spring - Thu, 12:45pm

## Summary

## Interface-based programming model

## Start simple, get more sophisticated

## Declarative query execution

## Flexible predicate execution

## Custom implementation

## CDI integration