Spring Data Repositories Best Practices

Thomas Darimont
Java User Group Saarland - 18. Treffen

22.03.2016

Thomas Darimont
Software Architect >eurodata AG

Spring Data Commiter
Former Member of Spring Team @ Pivotal

Java User Group
Saarland

t.darimont@eurodata.de
@thomasdarimont

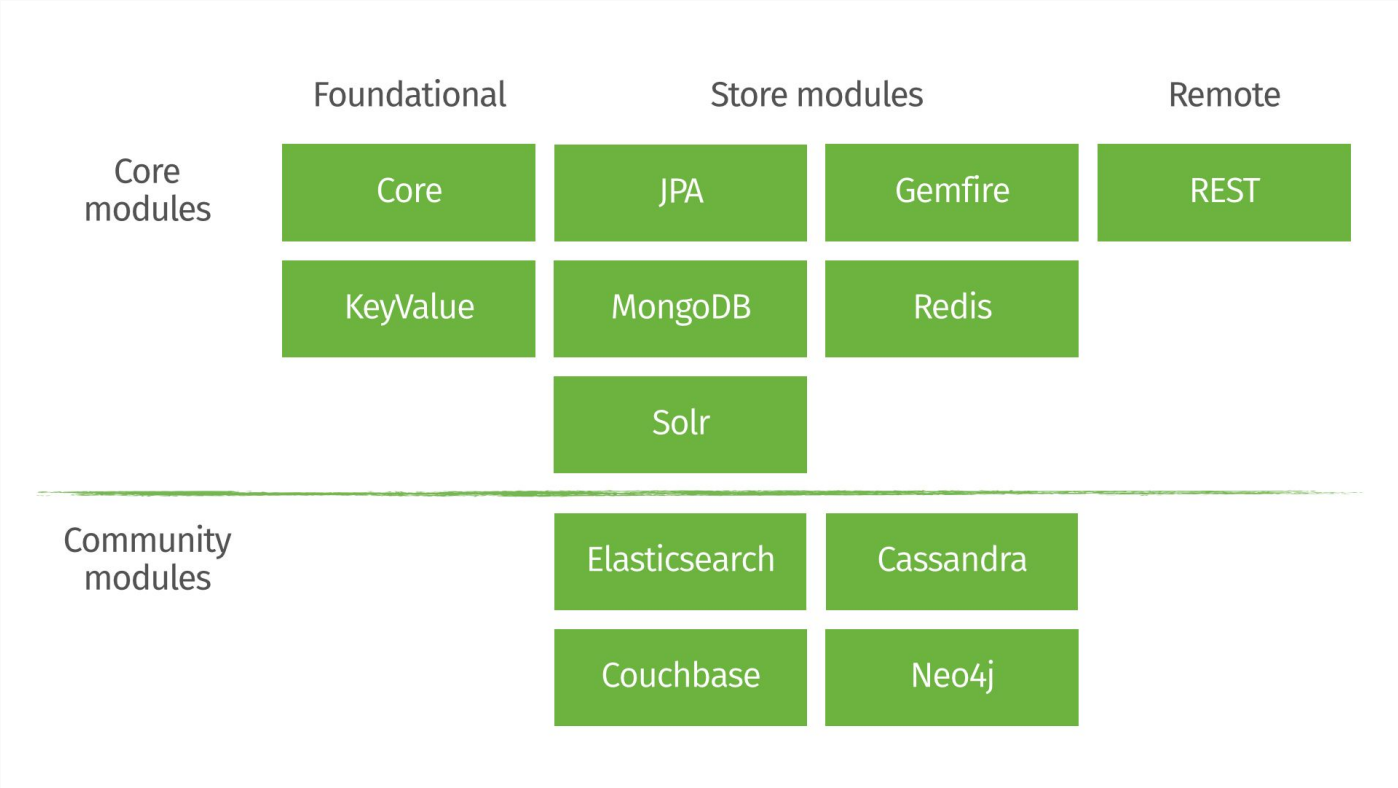
Spring Data

“ Provides a consistent approach to data access for several persistence models ”

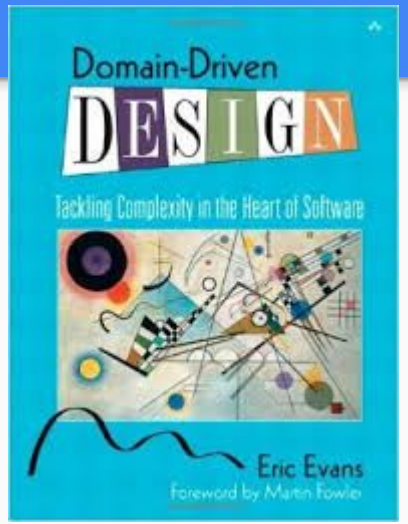
with support for

Configuration, Templates, Exception Translation, Object Mapping, ***Repository Abstraction*** *and various Store Modules*

Spring Data Modules & Store Support



Repositories



Concept from *Domain Driven Design*

A Repository...

"...mediates between the domain and data mapping layers using a collection-like interface for accessing domain objects."

http://martinfowler.com/eaaCatalog/repository.html

Spring Data Repositories

- Pragmatic Data Access API’s
- Interface based Programming Model
- Provides useful Base Abstractions
- Automatic Query Generation
- … and much much more - let’s find out!

Hands On

All code on github!

Steps as individual commits :)

thomasdarimont/sd-repositories-best-practices-javaland

Step 0 - Project Setup

Requirement

“Setup Data Access Infrastructure with Spring, JPA, H2”

1. Spring Source Toolsuite (STS)
2. Spring Boot Starter to create Maven Project
3. Add JPA + H2 Dependency
4. Profit!

Step 0 - Summary

- Spring Boot + Spring Data = Easy!
- Easily configure dependencies
- Defaults application config from classpath

→ Spring Data Infrastructure Ready to use!

Step 1 - Setup Domain Model

Requirement

> *“Define Domain Model, Populate & Test Database”*

- Define basic JPA entities
- Populate Database with a Script
- Test Population

Step 1 - Summary

- We defined Customer Entity + Address ValueObject
- Populate Database via data.sql
- Test Population via Integration Test

Step 2 - Enable JPA Repositories

Requirement

> *"**Customers** can be **saved**, **looked up** by their **id** and **email address**."*

- Define Customer Repository
- Basic CRUD functionality
- Based on Spring Data Repositories

Step 2 - Summary

- Interface-based programming model
- No implementation required
- Proxy with Implementation provided by Spring Data
- Queries derived from method names

Step 3 - Extended CRUD Methods

Requirement

> “Customers can be ***deleted*** and ***listed***!”

Step 3 - Summary

- Switched to CrudRepository
- Exposed CRUD methods
- Broad API exposed

Step 4 - Add Pagination Support

Requirement

> “A User
> … can ***pagewise*** browse through a ***Customer list***.
> … wants to ***browse*** through the ***list*** of Customers ***sorted by lastname*** in ***desc order***.”

Step 4 - Summary

- Switched to PagingAndSortingRepository
- Exposed CRUD methods + Pagination
- Broad API exposed
- Tip: Use Slice instead of Page if possible
  - Slice as a Return type
  - Avoids count queries

Step 5 - Redeclare existing methods

Requirement

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

Step 5 - Summary

- Re-declared methods
- To customize return types
- Customize Behaviour *e.g. TX, Locking, Query, Hints, Fetching*
- Customize Query

Step 6 - Def. your own Abstractions

Requirement

> *“**Products*** shall be ***accessible*** only ***in read-only mode***.”

Step 6 - Summary

- We crafted our own custom abstraction
- Applied by implementing base interface
- Customize return types
- Narrow down the API to the necessary parts

Step 7 - Using custom queries

Requirement

> “As a user, I want to ***look up products by custom attributes***.”

Step 7 - Summary

- You can customize the queries
- Via @Query annotation
- JPA named queries
  - @NamedQuery
  - <named-query> in orm.xml
- Spring Data named queries properties file
  - e.g. jpa-named-queries.properties

Step 8 - Flexible Predicate execution

Requirement

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

Step 8 - Summary

- Querydsl - type safe queries for Java
- extend QuerydslPredicateExecutor
- Flexible query model

Step 9 - Custom Implementations

Requirement

> “As an admin user, I'd like to *use custom code* to *raise* all prices before winter sale.”

Step 9 - Summary

- Provide custom implementation
- Dependency Injection
- Base class support
  (Querydsl, Hibernate, Jdbc-DaoSupport)

Step 10 - Custom base class

Requirement

> *“I'd like to use a custom base class with new methods for all repositories.”*

Step 10 - Summary

- Provide custom base class
- via *@EnableJpaRepository(repositoryBaseClass=MyRepoBaseClass.class)*
- Reuse existing functionality and add your own

Step 11 - Predicates from MVC Request

Requirement

> ”I'd like to create flexible query predicates based on http request/URL parameters”

Step 11 - Summary

- Bind Request/URL Parameters to Predicate
- QueryDSL Predicate as Parameter in your MVC Controller method

Stuff on top

- Spring MVC integration
  - Native support for Pagination
  - Inject Domain Instances into your MVC handlers
  - Expose parts of Domain Model via Projections
- Spring Data REST
- Spring Boot Integration
- Spring Security Integration

Summary

Spring Data Repositories

Interface-based programming model

Spring Data Repositories

Start simple, get more sophisticated

Spring Data Repositories

Declarative Query Execution

Spring Data Repositories

Flexible Predicate Execution

Spring Data Repositories

Custom Implementations

Spring Data Repositories

CDI Integration

Spring Data Repositories

Spring Security Integration

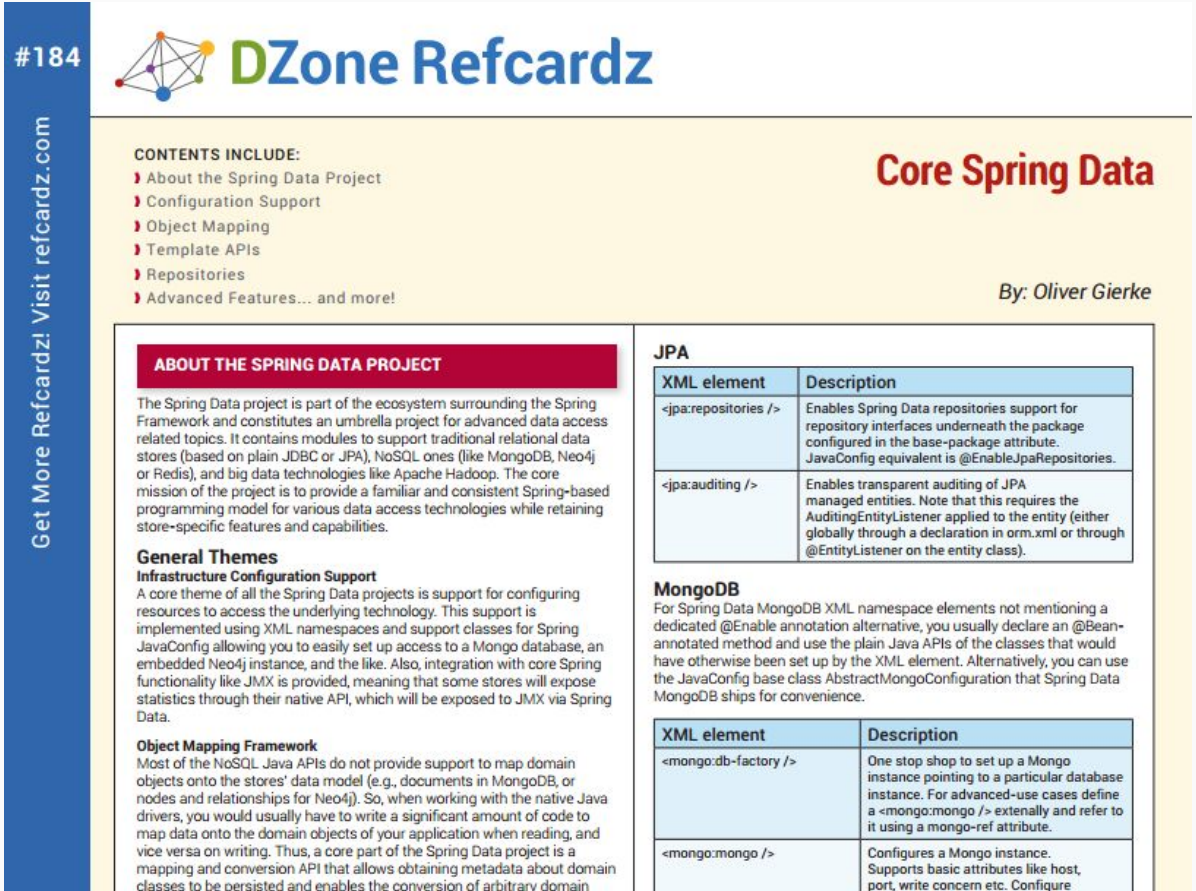
Spring Data Examples

spring-projects/spring-data-examples

Questions


QUESTIONS
FOUND IN GOOGLE AUTOCOMPLETE
WHY DO WHALES JUMP
WHY ARE WITCHES GREEN
WHY ARE THERE MIRRORS ABOVE BEDS
WHY DO I SAY UH
WHY IS SEA SALT BETTER
WHY ARE THERE TREES IN THE MIDDLE OF FIELDS
WHY IS THERE NOT A POKEMON MMO
WHY IS THERE LAUGHING IN TV SHOWS
WHY ARE THERE DOORS ON THE FREEWAY
WHY ARE THERE SO MANY SHOWS ON E RUNNING
WHY AREN'T THERE ANY COUNTRIES IN ANTARCTICA
WHY ARE THERE SORRY SOUNDS IN MINECRAFT
WHY IS THERE KICKING IN MY STOMACH
WHY ARE THERE TWO SLASHES AFTER HTTP
WHY ARE THERE CELEBRITIES
WHY DO SNAKES EXIST
WHY DO OYSTERS HAVE PEARLS
WHY ARE DUCKS CALLED DUCKS
WHY DO THEY CALL IT THE CLAP
WHY ARE KYLE AND CARTMAN FRIENDS
WHY IS THERE AN ARROW ON FANG'S HEAD
WHY ARE TEXT MESSAGES BLUE
WHY ARE THERE MUSTACHES ON CLOTHES
WHY ARE THERE MUSTACHES ON CARS
WHY ARE THERE MUSTACHES EVERYWHERE
WHY ARE THERE SO MANY BIRDS IN OHIO
WHY IS THERE SO MUCH RAIN IN OHIO
WHY IS OHIO WEATHER SO WEIRD
WHY ARE THERE MALE AND FEMALE BIKES
WHY ARE THERE SQUIRRELS
WHY ARE THERE TINY SPIDERS IN MY HOUSE
WHY DO SPIDERS COME INSIDE
WHY ARE THERE HUGE SPIDERS IN MY HOUSE
WHY ARE THERE LOTS OF SPIDERS IN MY HOUSE
WHY ARE THERE SPIDERS IN MY ROOM
WHY ARE THERE SO MANY SPIDERS IN MY ROOM
WHY DO SPIDER BITES ITCH
WHY IS DYING SO SCARY
WHY DO KNEES CLICK
WHY AREN'T THERE E-GARDENS
WHY IS ISOLATION BAD
WHY DO BOYS LIKE ME
WHY DON'T BOYS LIKE ME
WHY IS LYING GOOD
WHY DO TREES DIE
WHY DO TESTICLES MOVE
WHY ARE THERE PSYCHICS
WHY ARE HATS SO EXPENSIVE
WHY DO YOUR BOOGERS HURT
WHY ARE THERE SLAVES IN THE BIBLE
WHY DO TWINS HAVE DIFFERENT FINGERPRINTS
WHY ARE AMERICANS AFRAID OF DRAGONS
WHY IS HTTPS CROSSED OUT IN RED
WHY IS THERE A LINE THROUGH HTTPS
WHY IS HTTPS IMPORTANT
WHY AREN'T MY ARMS GROWING
WHY AREN'T ECONOMISTS RICH
WHY DO AMERICANS CALL IT SOCCER
WHY ARE MY EARS RINGING
WHY ARE THERE SO MANY AVENGERS
WHY ARE THE AVENGERS FIGHTING THE X-MEN
WHY IS WOLVERINE NOT IN THE AVENGERS
WHY ARE THERE SO MANY CROWS IN ROCHESTER, MN
WHY IS PSYCHIC WEAK TO BUG
WHY DO CHILDREN GET CANCER
WHY IS POSEIDON ANGRY WITH ODYSSEUS
WHY IS THERE ICE IN SPACE
WHY ARE THERE ANTS IN MY LAPTOP
WHY IS EARTH TILTED
WHY IS SPACE BLACK
WHY IS OUTER SPACE SO COLD
WHY ARE THERE PYRAMIDS ON THE MOON
WHY IS NASA SHUTTING DOWN
WHY ARE THERE GHOSTS
WHY IS THERE AN OWL IN MY BACKYARD
WHY IS THERE AN OWL OUTSIDE MY WINDOW
WHY IS THERE AN OWL ON THE DOLLAR BILL
WHY DO OWLS ATTACK PEOPLE
WHY ARE AK 47S SO EXPENSIVE
WHY ARE THERE HELICOPTERS CIRCLING MY HOUSE
WHY ARE THERE GODS
WHY ARE THERE TWO SPOOKS
WHY ARE MY BOOBS ITCHY
WHY ARE CIGARETTES LEGAL
WHY ARE THERE DUCKS IN MY POOL
WHY IS JESUS WHITE
WHY IS THERE LIQUID IN MY EAR
WHY DO Q-TIPS FEEL GOOD
WHY DO GOOD PEOPLE DIE
WHY ARE DOGS AFRAID OF FIREWORKS
WHY IS THERE NO KING IN ENGLAND
WHY IS MT VESUVIUS THERE
WHY DO THEY SAY T MINUS
WHY ARE THERE OBELISKS
WHY ARE WRESTLERS ALWAYS WET
WHY ARE OCEANS BECOMING MORE ACIDIC
WHY IS ARWEN DYING
WHY AREN'T MY QUAIL LAYING EGGS
WHY AREN'T MY QUAIL EGGS HATCHING
WHY AREN'T THERE ANY FOREIGN MILITARY BASES IN AMERICA
WHY IS LIFE SO BORING
WHY IS SEX SO IMPORTANT
WHY AREN'T THERE GUNS IN HARRY POTTER
WHY IS STEALING WRONG
WHY ARE THERE FEMALE MR MIMES
WHY IS THERE HELL IF GOD FORGIVES
WHY IS GPS FREE

DZone Refcard



#184

Get More Refcardz! Visit refcardz.com

DZone Refcardz

CONTENTS INCLUDE:

- About the Spring Data Project
- Configuration Support
- Object Mapping
- Template APIs
- Repositories
- Advanced Features... and more!

Core Spring Data

By: Oliver Gierke

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

Most of the NoSQL Java APIs do not provide support to map domain objects onto the stores' data model (e.g., documents in MongoDB, or nodes and relationships for Neo4j). So, when working with the native Java drivers, you would usually have to write a significant amount of code to map data onto the domain objects of your application when reading, and vice versa on writing. Thus, a core part of the Spring Data project is a mapping and conversion API that allows obtaining metadata about domain classes to be persisted and enables the conversion of arbitrary domain

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 instance pointing to a particular database instance. For advanced-use cases define a <mongo:mongo /> extenally and refer to it using a mongo-ref attribute.
<mongo:mongo />	Configures a Mongo instance. Supports basic attributes like host, port, write concern etc. Configure