Spring Data in 10 minutes

Corneil du Plessis

corneil.duplessis@gmail.com
about.me/corneil
@corneil

Introduction

- Assumptions
  - You know JPA the Java Persistence API
  - You know the Spring Framework or Dependency Injection
- Take away
  - Some appreciation for polyglot persistence
  - Some understanding of Spring Data

What is Polyglot persistence?

- RDBMS doesn't scale to Internet (millions of users and billions of transactions).
- Ability to interact with multiple data stores depending on specific use cases
- MongoDB or Couchbase for Customer data
- RDBMS for financial transactions
- Elasticsearch, Cassandra or HBase for site clicks and audit trails
- Hadoop for Big Data
- Neo4J, OrientDB for Graph Data
- RDF Stores for Semantic Reasoning

Why Spring Data?

- Best practice indicates you should have Data Access Components or a Data Service Layer.
- How often do you make simple mistakes interacting with EntityManager?
- Different API for each store.
- It should be easier...

What is Spring Data?

- Spring Data is an open source project managed by SpringSource.
- Spring Data relies on Spring Framework.
- Spring Data provides a set of common patterns for persisting data using existing libraries.
- Spring Data provides a Repository Interface
- Spring Data provides finder methods
- Spring Data provides support for QueryDSL
- Spring Data simplifies polyglot persistence

Spring Data sub projects

- Commons
- JPA
- JDBC Extensions
- MongoDB
- Neo4J
- Redis
- HBase
- Hadoop
- GemFire
- REST provider
- Community Projects
  - OrientDB
  - RDF
  - Couchbase
  - Cassandra
  - Elasticsearch
  - DynamoDB

How Spring Data?

- Configure Spring Data for specific technology
- Define Entity as per relevant library
- Declare Repository Interface
- Spring Data will provide Repository Implementation
- Spring Framework will inject Repository implementation where needed

Spring Data Samples – Configuration

- EntityManagerFactory as normal for JPA
-
```
<jpa:repositories
 base-package="...springdata.demo" />
```

OR

-
```
@EnableJpaRepositories("...springdata.demo")
```

Spring Data Sample – Entity

```
@Entity
public class User {
    @Id
    Long id;
    String fullName;
    String gender;
    Date dateOfBirth;
}
```

Spring Data Sample – Repository

- ```
  @Repository
  public interface UserRepository
    extends CrudRepository<User> {
  }
  ```

- Crud Repository provides type-safe methods for save, delete, load
  ```

Spring Data Sample - Injection

```
@Service
public class MyDataServiceImpl {
  @AutoWired
  protected UserRepository userRepo;
  public void saveUser(User user) {
    userRepo.save(user);
  }
}
```

Spring Data Sample – Finders

- ```
  @Repository
  public interface UserRepository
    extends CrudRepository<User> {
    List<User> findByGender(
            String genderCode);
    List<User> findByDateOfBirthBetween(
            Date startDate, Date endDate);
  }
  ```
- Crud Repository provides query predicates based on method name like:

  findByNameOrderByDateOfBirthDesc
  findByNameORSurname
  ```

Spring Data Sample – QueryDSL

- ```
  @Entity
  @QueryEntity
  public class User {
      @Id
      private Long id;
      private String fullName;
      private String gender;
      private Date dateOfBirth;
  }
  ```

- QueryDSL generates QUser for creating type-safe query.

- ```
  import static Quser.*;
  List<User> users = userRepo.findAll(user.gender.eq('M'))
  ```

Spring Data – Notes

- User code is same on MongoDB, JPA, Neo4J, Redis etc.
- Entity Mapping is usually specific to technology.
- Common Repository methods can be implemented.
- Implementation specific queries:

```
@Query(“select * from User u where u.gender = ?”)
List<User> findUsersForGender(String code)
```

Spring Data – Questions

- Demo code at
  https://github.com/corneil/spring-data-demo

- Please Star the repo on github.com if you like the project.
- Like the slides on Slideshare