

Pivotal

Cloud Native Workshop

Introduction to Spring Data Rest

Pivotal Cloud Foundry

Pivotal

Spring Data Rest



- **Review Spring Data**
- Spring Data Rest

Pivotal

What type of data?

Spring Data

- Spring Data JPA
- Spring Data MongoDB
- Spring Data Redis
- Spring Data Solr
- Spring Data GemFire
- Spring Data Rest



Import the required dependency

Add the JPA starter to the pom.xml

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
```

Repositories

Tired of Creating/Maintaining Boilerplate Code?

Use Spring Repositories. CRUD support added with no implementation required.

```
public interface CrudRepository<T, ID extends Serializable> extends Repository<T, ID> {  
  
    // Saves the given entity  
    <S extends T> S save(S entity);  
  
    // Returns the entity identified by the given id.  
    T findOne(ID primaryKey);  
  
    // Returns all entities.  
    Iterable<T> findAll();  
  
    // Deletes the given entity.  
    void delete(T entity);  
  
    // ... more functionality omitted.  
}
```

Pivotal

© Copyright 2015 Pivotal. All rights reserved.

5

Defining your own repository interface

Extend from the given repository and provide the **domain** and **id** classes:

```
public interface CitiesRepository extends JpaRepository<Cities, Long>{  
  
}
```

Pivotal

© Copyright 2015 Pivotal. All rights reserved.

6

Add required methods as needed

```
public interface PersonRepository extends JpaRepository<User, Long> {  
  
    List<Person> findByEmailAddressAndLastname  
        (EmailAddress emailAddress, String lastname);  
  
    // Enables the distinct flag for the query  
    List<Person> findDistinctPeopleByLastnameOrFirstname  
        (String lastname, String firstname);  
  
    List<Person> findPeopleDistinctByLastnameOrFirstname  
        (String lastname, String firstname);  
  
    // Enabling ignoring case for an individual property  
    List<Person> findByLastnameIgnoreCase (String lastname);  
}
```

Query creation from method names

- Strip prefixes : `find...By`, `read...By`, and `get...By`
- Introducing clause: `Distinct`
- First `By` acts as a delimiter to indicate start of criteria
- `And` and `Or`
- `Between`, `LessThan`, `GreaterThan`, `Like`
- `IgnoreCase`

@Query as an alternative to keywords

A JPA based repository using the @Query annotation.

```
public interface UserRepository extends JpaRepository<User, Long> {  
    @Query("select u from User u where u.emailAddress = ?1")  
    User findByEmailAddress(String emailAddress);  
}
```

Spring Data Rest



- Review Spring Data
- **Spring Data Rest**

Spring Data Rest

The **goal** of the **Spring Data REST** project is to provide a solid foundation on which to expose **CRUD** operations to your **repository managed entities** using plain **HTTP REST semantics**.

Import the required dependency

Add the Spring Data Rest starter to the pom.xml:

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-rest</artifactId>
</dependency>
```

Export repositories

```
public interface OrderRepository extends CrudRepository<Order, Long> {  
    List<Order> findByDate(@Param("date") Date date);  
}
```

- For this repository, Spring Data REST exposes a collection resource at `/orders`.
- The path is derived from the un-capitalized, pluralized, simple class name of the domain class being managed.
- It also exposes an item resource for each of the items managed by the repository under the URI template `/orders/{id}`.
- Custom queries are exported to `/search`. E.g. `/search/findByDate`

RESTful API



HATEOAS

(Hypermedia as the Engine of Application State)

- Provides information to navigate the **REST** interface dynamically by including hypermedia links with responses
- Differs from **SOA** based systems and WSDL-driven interfaces, in that a separate fixed specification is distributed
- **HAL** - Hypertext Application Language



Pivotal

© Copyright 2015 Pivotal. All rights reserved.

15

ALPS (Application-Level Profile)

- Alps is a data format for defining simple descriptions of application-level semantics.
- Provides metadata on how interact with the system.
- Provides details on domain representation, operations



Pivotal

© Copyright 2015 Pivotal. All rights reserved.

16

ALPS explained

<http://localhost:8080/alps/persons>

```
{
  "version" : "1.0",
  "descriptors" : [ {
    //representation of domain
    "id" : "person-representation",
    "descriptors" : [ {
      "name" : "firstName",
      "type" : "SEMANTIC"
    }, {
      "name" : "lastName",
      "type" : "SEMANTIC"
    }, {
      "name" : "id",
      "type" : "SEMANTIC"
    }
  ]
}, {
  "id" : "create-persons", //operations
  "name" : "persons",
  "type" : "UNSAFE",
  "rt" : "#person-representation"
}
...
```

```
...
}, {
  "id" : "get-persons",
  "name" : "persons",
  "type" : "SAFE",
  "rt" : "#person-representation"
}, {
  "id" : "delete-person",
  "name" : "person",
  "type" : "IDEMPOTENT",
  "rt" : "#person-representation"
}, {
  "id" : "patch-person",
  "name" : "person",
  "type" : "UNSAFE",
  "rt" : "#person-representation"
}, {
  "id" : "update-person",
  "name" : "person",
  "type" : "IDEMPOTENT",
  "rt" : "#person-representation"
}, {
  "id" : "get-person",
  "name" : "person",
  "type" : "SAFE",
  "rt" : "#person-representation"
} ]
}
```

Pivotal

© Copyright 2015 Pivotal. All rights reserved.

17

Pivotal

A NEW PLATFORM FOR A NEW ERA