

# Spring Data REST

# What is Spring Data REST?

# What is Spring Data REST?

---

- Spring Data REST exposes your Spring Data Repositories as REST Service
- REST Service built on top of Spring HATEOAS
- Uses HAL by default to render links

# Hypermedia & REST

# Principles of REST

---

- **Resources** expose easily understood directory structure URIs.
- **Representations** transfer JSON or XML to represent data objects and attributes.
- **Messages** use HTTP methods explicitly (for example, GET, POST, PUT, and DELETE).
- **Stateless** interactions store no client context on the server between requests. State dependencies limit and restrict scalability. The client holds session state.

# Hypermedia

---

- Important aspect of REST
- For building services that decouple client and server
- Allowing them to evolve independently
- Representations returned for REST resources contain data and links

# Spring Data REST Setup

# Maven Dependencies

---

- Adding Spring Data REST Starter along with Spring Data JPA and H2

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-rest</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
  <groupId>com.h2database</groupId>
  <artifactId>h2</artifactId>
</dependency>
```



# Setup

---

- Starter configures infrastructure Spring beans automatically
  - Spring MVC
  - Spring HATEOAS
  - Jackson Marshaller

# Entities & Repositories

# Creating JPA Entities

---

```
@Entity
public class Employee {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private String firstName;
    private String lastName;

    protected Employee() {}

    public Employee(String firstName, String lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
    }

    @Override
    public String toString() {
        return String.format(
            "Employee[id=%d, firstName='%s', lastName='%s']",
            id, firstName, lastName);
    }
}
```

# JPA & Rest Repository

---

- All Repository automatically exposed as REST Service
- @RepositoryRestResource used to override defaults

```
@RepositoryRestResource(collectionResourceRel = "worker", path = "worker")
```

```
public interface EmployeeRepository extends  
    JpaRepository<Employee, Long>{
```

```
    List<Employee> findByLastName(String lastName);
```

```
    @Query("select e from Employee e where e.lastName = ?")  
    List<Employee> findQueryByLastName(String name);
```

```
}
```

```
spring.data.rest.basePath=/api
```

# HAL Browser

# HAL Browser

The screenshot shows the HAL Browser web application running in a browser. The address bar shows `localhost:8080/browser/index.html#`. The page title is "The HAL Browser".

**Explorer**

URL: `/` Go!

**Custom Request Headers**

**Properties**

`{}`

**Links**

rel	title	name / index	docs	GET	NON-GET
persons					
addresses					
profile					

**Inspector**

**Response Headers**

200 OK

Date: Mon, 03 Aug 2015 19:55:45 GMT  
Server: Apache-Coyote/1.1  
Transfer-Encoding: chunked  
Content-Type: application/hal+json;charset=UTF-8

**Response Body**

```
{
  "_links": {
    "persons": {
      "href": "http://localhost:8080/persons?projection",
      "templated": true
    },
    "addresses": {
      "href": "http://localhost:8080/addresses"
    },
    "profile": {
      "href": "http://localhost:8080/alps"
    }
  }
}
```

`<dependency>`

`<groupId>org.springframework.data</groupId>`

`<artifactId>spring-data-rest-hal-browser</artifactId>`

`</dependency>`

# Important notes

# Important notes

---

- You are exposing with Spring Data REST your whole Database
- Tightly coupled to the database, changing data structure becomes very difficult
- Links should be rendered depending on your business needs -> Spring HATEOAS