ESD
Spring Data Rest Workshop

By Nils Theres, Luc Lehmkuhl, Alex Böning and Carl Rix

Structure

Overview of Spring Data Rest

Why & When?

How does it compare to other frameworks?

Core Concepts

Practical Applications

Workshop

Overview of Spring Data REST

- Web Framework that is part of the Spring Data Project
- Simplifies building RESTful APIs
- **Key features include**
  - Automatic API Endpoint Generation
  - HATEOAS and HAL Support
  - Pagination, Sorting and Filtering
  - JSON Serialization

Why & When should you use Spring Data REST?

Rapid API development

Reducing Boilerplate Code

Data-Centric Applications

Consistency with Spring Ecosystem

How does it compare to other frameworks?

Framework	Spring	Django	Fast API
Language	Java, Kotlin	Python	Python
Ease of Use	Simplified, conventions-over-configurations, opinionated framework	Known for its "batteries-included" philosophy.	Known for its intuitive syntax and Pythonic nature.
Community	Strong and active community, robust documentation.	Extensive documentation, active community.	Growing community due to its recent surge in popularity.
REST API dev.	Easily exposes data models via REST.	Great flexibility in designing APIs.	Asynchronous support offers performance advantages.
Testing	Provides a strong testing framework, supporting various testing methodologies.	Offers a user-friendly testing framework.	Easy testing based on Python’s pytest.
Microservices Archi.	Aligns well with microservices architecture and is considered cloud-native.	Simpler to set up for smaller applications or microservices.	Lightweight nature makes it suitable for microservices. Easily containerized.

Core Concepts

Core Concepts – Annotations

- Provide metadata and configuration
- Control over various aspects of how API is exposed, and entities are handled
- **Common annotations include**
  - @Entity: For domain entities
  - @Service: For services
  - @Component: For components
  - @Autowired: Automatic explicit injection of beans
- Helpful packages can extend features of annotations
  - Lombok, Jakarta and more

Core Concepts – Entity Modelling

- Class Definition
  - @Entity
  - Java class
- Primary Key
  - @Id
  - GenerationType
- Relationships
  - @OneToMany, @ManyToOne, @OneToOne, @ManyToMany
- Field Mappings
  - @Column
  - Data Types (String, int, Date, etc.)
- Validation Constraints
  - @NotNull, @NotBlank, @Min, @Max

```java
@Entity
public class Person {

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

    no usages
    @Column(nullable = false)
    private String name;

    no usages
    private int age;

    no usages
    private Date dateOfBirth;

    no usages
    @OneToOne
    private Passport passport;
}
```

Core Concepts – Repositories & Querying

- Used to export resources and automatically create endpoints
- Different Repository Interfaces (Crud, Jpa, PagingAndSorting, etc.)
- Dynamic Query Creation
- Support for named queries
- Interoperability with projections

```
public interface AuthorRepository extends JpaRepository<Author, Long> {
    no usages  new *
    Optional<Author> findAuthorByName(String name);


    no usages  new *
    @Query("select a from Author a where a.birthdate = ?1")
    Optional<Author> findAuthorByBirthdate(Date birthdate);
}
```

Core Concepts – Projections

- Interface- or Class-based
- Defined using the @Projection annotation
- Support for selective Data Exposure
- Nested Projections
- Support for lazy loading and query optimization

```java
@Projection(types = {Author.class}, name = "detailed")
public interface AuthorDetailedProjection {

    no usages  new *
    Long getId();


    new *
    String getName();


    no usages  new *
    Date getBirthDate();


    no usages  new *
    Set<BookOverviewProjection> getBooks();
}
```

Core Concepts – Validation

- Used to validate entities
- Marker Annotation Support (@NotNull, @NotEmpty, etc.)
- Support for cascade validation (@Valid)
- Error handling with @ControllerAdvice

```java
@Entity
public class Person {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    no usages
    @NotBlank
    private String name;

    no usages
    @Min(15)
    private int age;

    no usages
    @NotNull
    private Date dateOfBirth;
}
```

Core Concepts – Events

- Entity Lifecycle Events (@PrePersist, @PostLoad)
- Repository Events (@BeforeCreate, @AfterSave, etc.)
- Annotation Driven with Event Classes
- Support for conditional Events

```java
@Component
@RepositoryEventHandler
public class AuthorEventHandler {

    // Before creating an author
    no usages  new *
    @HandleBeforeCreate
    public void handleAuthorCreate(Author author) {
    }

    // Before saving an author
    no usages  new *
    @HandleBeforeSave
    public void handleAuthorSave(Author author) {
    }

    // After deleting an author
    new *
    @HandleAfterDelete
    public void handleAuthorDelete(Author author) {
    }
}
```

Core Concepts – HATEAOS

- Resource Representation
- Model Conversion
- Dynamic Link Creation
- Affordances
- HAL Format compliant


```
{
  "_links": {
    "loanRecords": {
      "href": "http://localhost:8080/loanRecords{?page,size,sort}",
      "templated": true
    },
    "authors": {
      "href": "http://localhost:8080/authors{?page,size,sort,projection}",
      "templated": true
    },
    "borrowers": {
      "href": "http://localhost:8080/borrowers{?page,size,sort}",
      "templated": true
    },
```

Core Concepts – ALPS

- Semantic Descriptor Profiles
- Support for Custom Types
- Automatic Documentation Integration
- Extendable


```
"alps": {
  "version": "1.0",
  "descriptor": [
    {
      "id": "author-representation",
      "href": "http://localhost:8080/profile/authors",
      "descriptor": [
        {
          "name": "name",
          "type": "SEMANTIC"
        },
        {
          "name": "birthdate",
          "type": "SEMANTIC"
        },
        {
          "name": "books",
          "type": "SAFE",
          "rt": "http://localhost:8080/profile/books#book-representation"
        }
      ]
    },
    {
      "id": "create-authors",
      "name": "authors",
      "type": "UNSAFE",
      "descriptor": [],
      "rt": "#author-representation"
    },
```

Practical Applications

E-Commerce Platforms

Banking and Financial Services

Healthcare Systems

Content Management Systems

Internet of Things

Questions

Quiz

Workshop

Assignment: Comprehensive Library Management API

- **Core Activities**
  - Entity Modelling
  - Repository Creation
  - Service Layer Incorporation
  - Endpoint Generation

- **Learning Outcomes**
  - Master Spring Data Rest Fundamentals
  - Gain practical Experience in building RESTful Services
  - Understand Entity Relationships
  - Learn about Error Handling and Validation

Next Steps

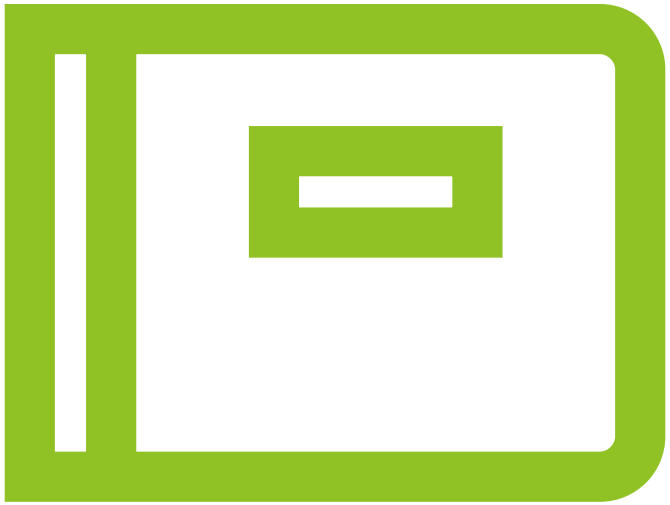
Clone the Repository

Load the Project in Your IDE

Explore the Project Structure

Read the Assignment Description

Test the Application with the provided sample Requests



Demo