

Table of Contents

1. [Congratulations](#)
2. [About in28Minutes](#)
3. [Troubleshooting Guide](#)
4. [Getting Started](#)
5. [Spring Microservices - Course Overview](#)
6. [Introduction to Web Services](#)
7. [Restful Web Services with Spring Boot](#)
8. [Best Practices with REST](#)
9. [Microservices with Spring Cloud](#)
10. [Bonus Introduction Sections](#)
11. [Keep Learning in28Minutes](#)

Getting Started

Recommended Versions

Tool/Framework/Language	Recommended Version	More Details
Java	Java 8	http://www.in28minutes.com/spr...
Eclipse	Eclipse Java EE Oxygen	Basics
Spring Boot	Spring Boot 2.0.0.RELEASE	
Spring Cloud	Finchley.M8	

Installation

- Video : https://www.youtube.com/playlist?list=PLBBog2r6uMCSmMVTW_QmDLyASBvovyAO3
- PDF
: https://github.com/in28minutes/SpringIn28Minutes/blob/master/InstallationGuide-JavaEclipseAndMaven_v2.pdf
- More Details : <https://github.com/in28minutes/getting-started-in-5-steps>

Troubleshooting

- A 50 page troubleshooting guide with more than 200 Errors and Questions answered

Spring Microservices - Course Overview

Github Repository :

<https://github.com/in28minutes/spring-microservices/>

Course Overview

Title	Github	
Introduction To Web Services	None	
Restful Web Services with Spring Boot	Project Folder on Github	
Microservices with Spring Cloud	Project Folder on Github	

2 Bonus Sections - Introduction to Spring Boot and JPA

Title	Category	Github
Spring Boot in 10 Steps	Introduction	Project Folder on Github
JPA in 10 Steps	Introduction	Project Folder on Github

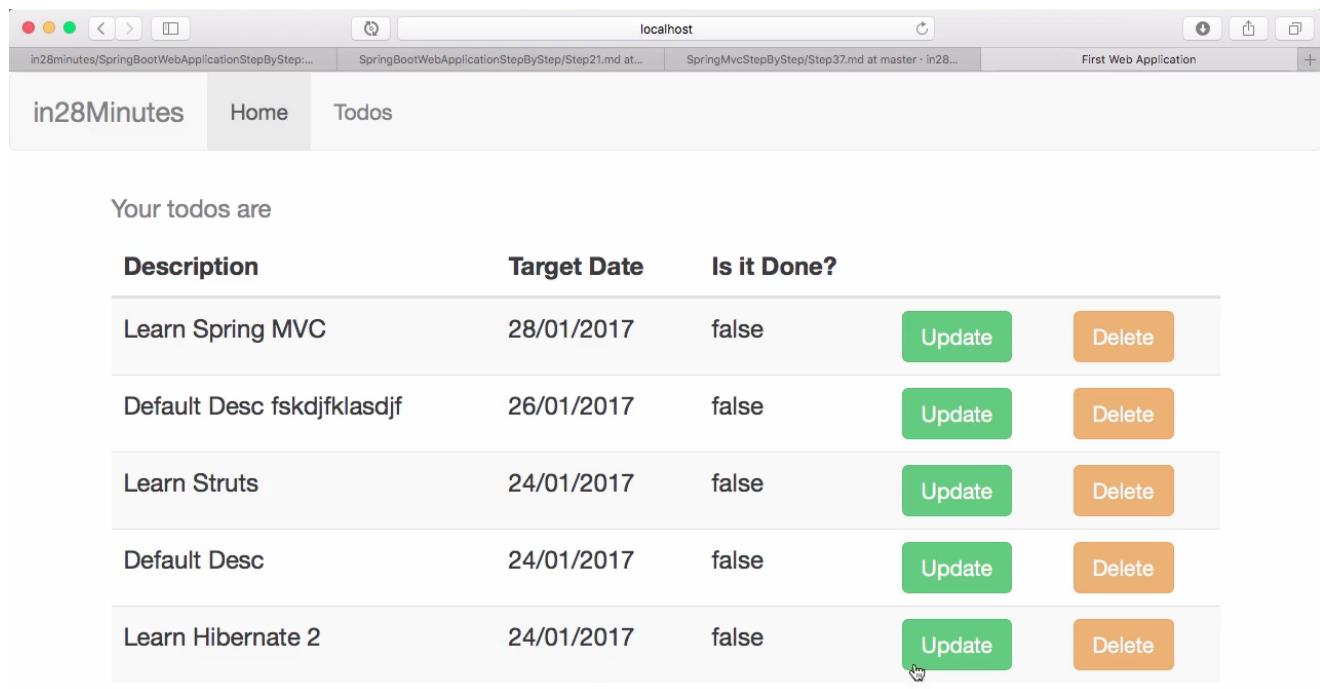
Introduction to Web Services

Introduction to Web Services

- What is a Web Service?
- Important How Questions related to Web Services
- Web Services - Key Terminology
- Introduction to SOAP Web Services
- Introduction to RESTful Web Services
- SOAP vs RESTful Web Services

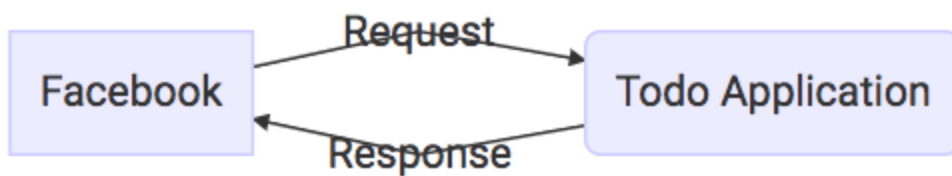
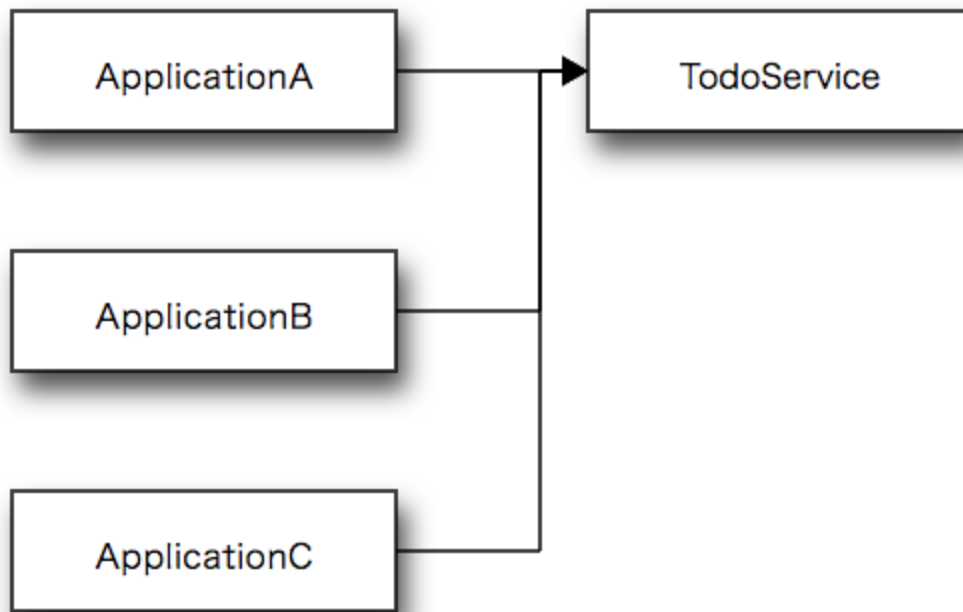
Web Service

Service delivered over the web?



Description	Target Date	Is it Done?	Update	Delete
Learn Spring MVC	28/01/2017	false	Update	Delete
Default Desc fskdjfklsdjf	26/01/2017	false	Update	Delete
Learn Struts	24/01/2017	false	Update	Delete
Default Desc	24/01/2017	false	Update	Delete
Learn Hibernate 2	24/01/2017	false	Update	Delete

Is the Todo Management Application a Web Service?



Web Service - W3C definition

Software system designed to support interoperable machine-to-machine interaction over a network.

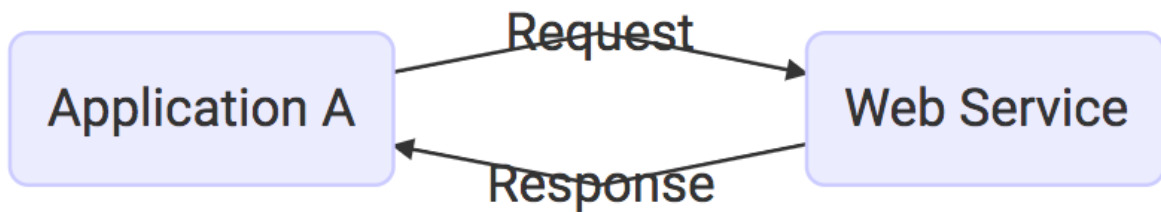
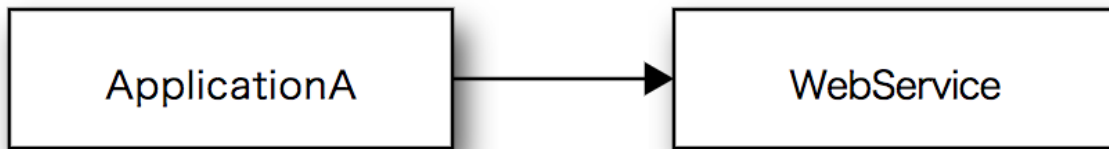
3 Keys

- Designed for machine-to-machine (or application-to-application) interaction

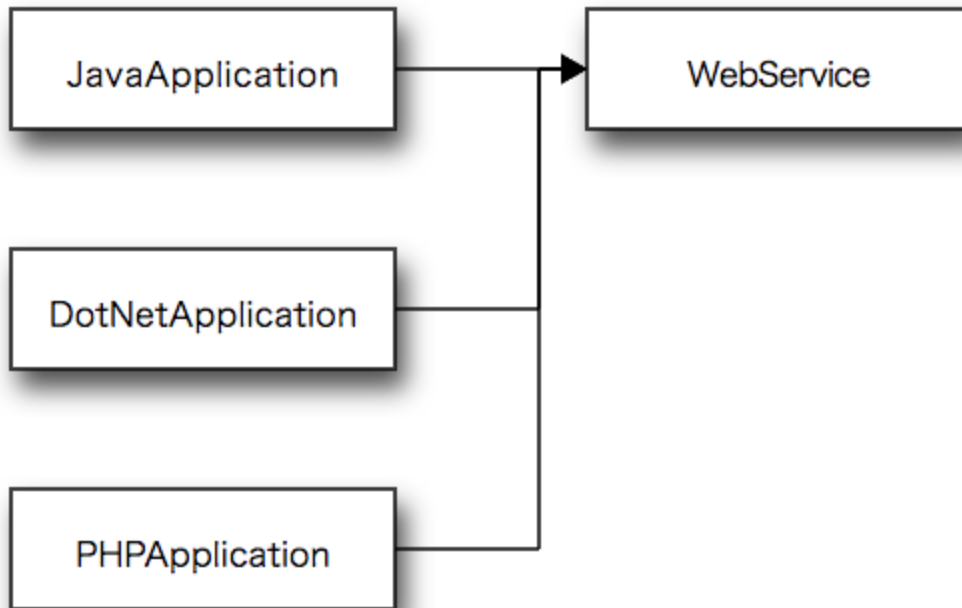
- Should be interoperable - Not platform dependent
- Should allow communication over a network

How?

How does data exchange between applications take place?



How can we make web services platform independent?



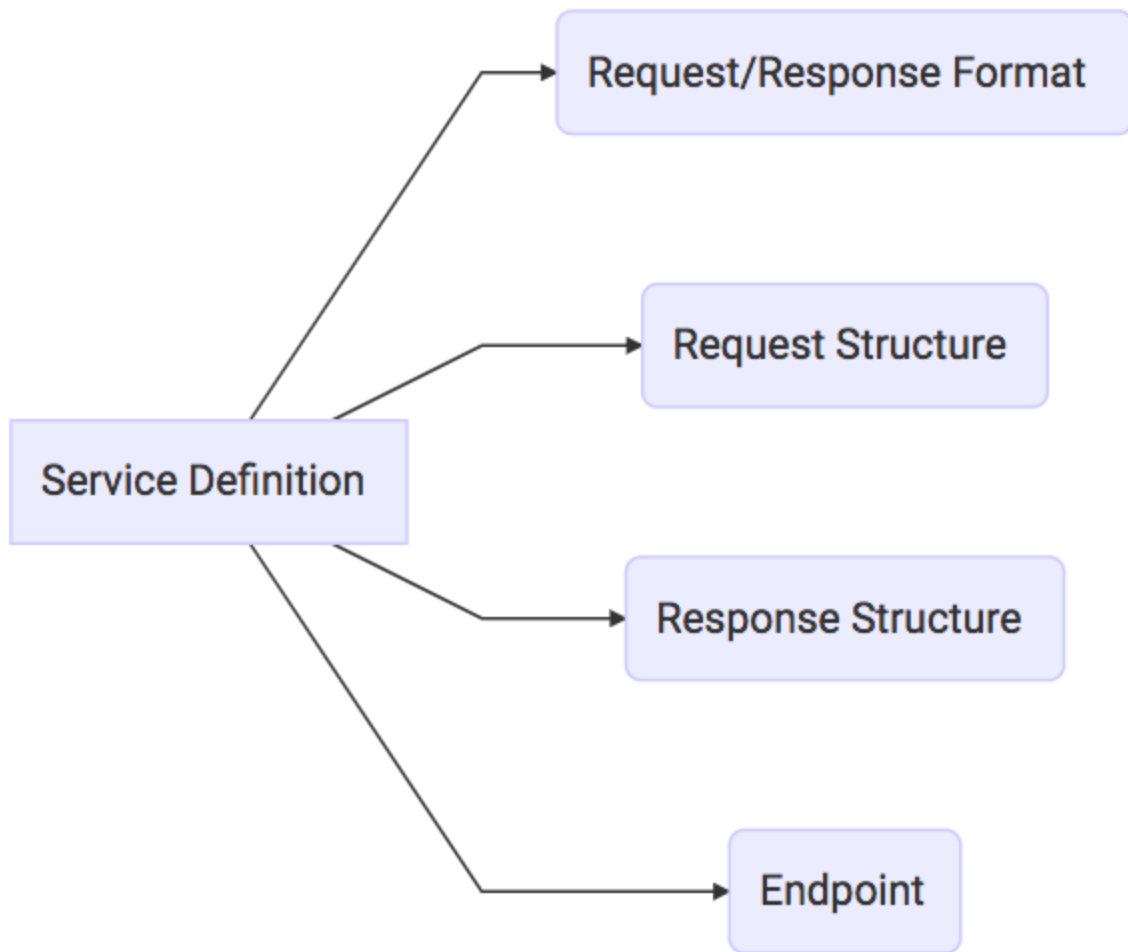
XML

```
<getCourseDetailsRequest>  
  <id>Course1</id>  
</getCourseDetailsRequest>
```

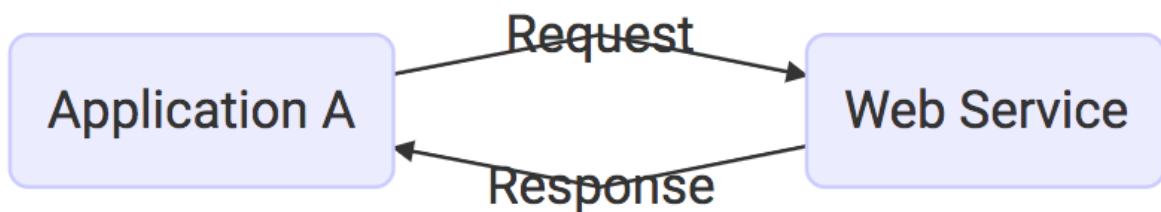
JSON

```
[  
  {  
    "id": 1,  
    "name": "Even",  
    "birthDate": "2017-07-10T07:52:48.270+0000"  
  },  
  {  
    "id": 2,  
    "name": "Abe",  
    "birthDate": "2017-07-10T07:52:48.270+0000"  
  }  
]
```

How does the Application A know the format of Request and Response?



How does Application A and Web Service convert its internal data to (XML or JSON)?



Key Terminology

- Request and Response

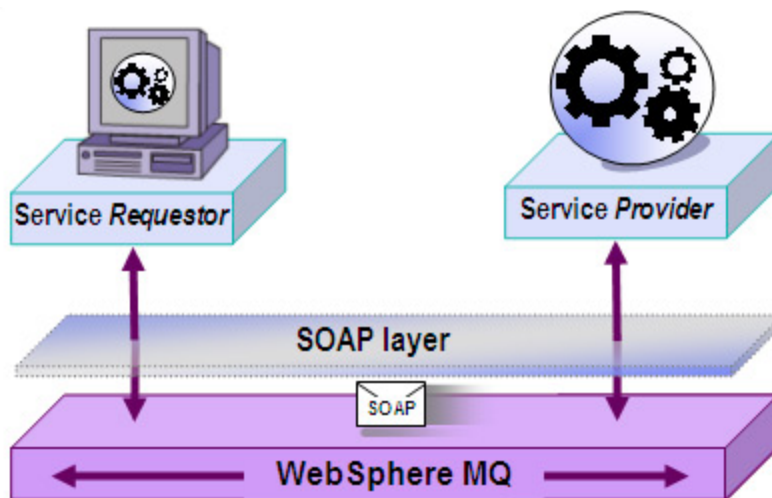
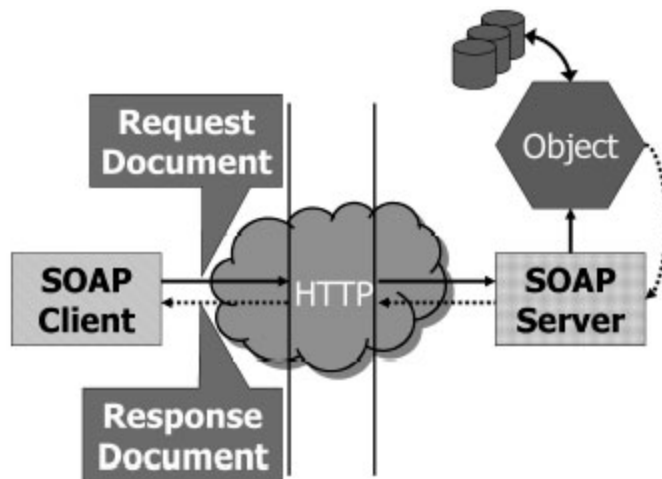
- Message Exchange Format
 - XML and JSON

Key Terminology

- Service Provider or Server
- Service Consumer or Client
- Service Definition

Key Terminology

- Transport
 - HTTP and MQ



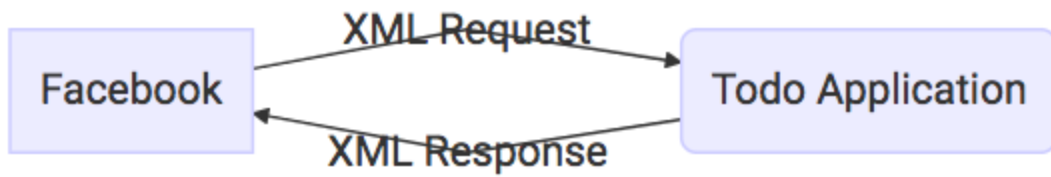
Web Service Groups

- SOAP-based
- REST-styled

SOAP and REST are not really comparable.

SOAP

SOAP?

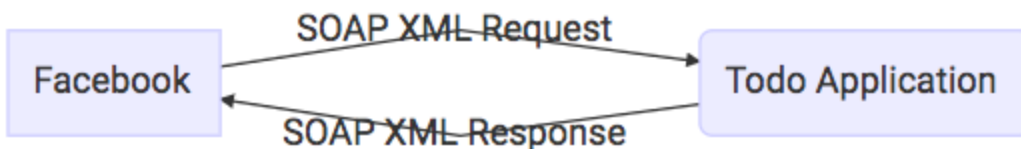


```
<getCourseDetailsRequest>  
  <id>Course1</id>  
</getCourseDetailsRequest>
```

SOAP-ENV: Envelope

SOAP-ENV: Header

SOAP-ENV: Body



```
<SOAP-ENV:Envelope xmlns:SOAP-  
ENV="http://schemas.xmlsoap.org/soap/envelope/">  
  <SOAP-ENV:Header/>  
  <SOAP-ENV:Body>  
    <ns2:getCourseDetailsResponse  
xmlns:ns2="http://in28minutes.com/courses">  
      <ns2:course>
```

```

<ns2:id>Course1</ns2:id>
      <ns2:name>Spring</ns2:name>
      <ns2:description>10 Steps</ns2:description>
    </ns2:course>
  </ns2:getCourseDetailsResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

SOAP

- Format - SOAP XML Request - SOAP XML Response
- Transport
 - SOAP over MQ
 - SOAP over HTTP
- Service Definition
 - WSDL

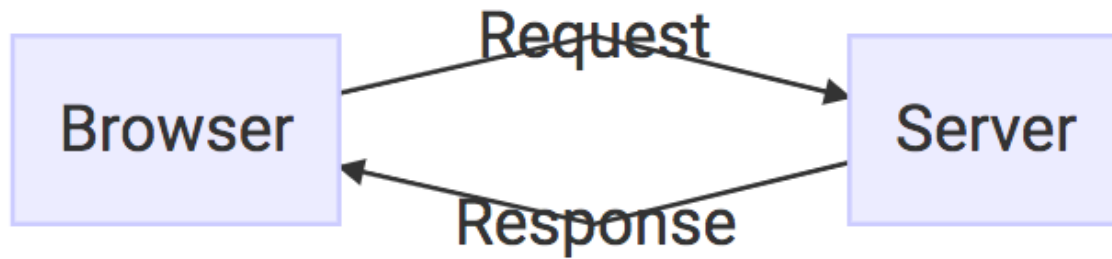
REST

REpresentational State Transfer

REST is a style of software architecture for distributed hypermedia systems

Make best use of HTTP

REST(REpresentational State Transfer)	
HTTP	
HTTP Methods (GET, PUT, POST..)	HTTP Status Codes (200, 404..)



Key abstraction - Resource

- A resource has an URI (Uniform Resource Identifier)
- /users/Ranga/todos/1
- /users/Ranga/todos
- /users/Ranga
- A resource can have different representations
- XML
- HTML
- JSON

Example

- Create a User - POST /users
- Delete a User - DELETE /users/1
- Get all Users - GET /users
- Get one Users - GET /users/1

REST

- Data Exchange Format - No Restriction. JSON is popular
- Transport
 - Only HTTP
- Service Definition
 - No Standard. WADL/Swagger/...

REST vs SOAP

- Restrictions vs Architectural Approach
- Data Exchange Format
- Service Definition
- Transport

-

- Ease of implementation

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <ns2:GetCourseDetailsRequest
xmlns:ns2="http://in28minutes.com/courses">
      <ns2:id>Course1</ns2:id>
    </ns2:GetCourseDetailsRequest>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <ns2:GetCourseDetailsResponse
xmlns:ns2="http://in28minutes.com/courses">
      <ns2:CourseDetails>
        <ns2:id>Course1</ns2:id>
        <ns2:name>Spring</ns2:name>
        <ns2:description>10 Steps</ns2:description>
      </ns2:CourseDetails>
    </ns2:GetCourseDetailsResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Restful Web Services with Spring Boot

Github Folder

<https://github.com/in28minutes/spring-microservices/tree/master/02.restful-web-services>

Restful Web Services with Spring Boot

- Step 01 - Initializing a RESTful Services Project with Spring Boot
- Step 02 - Understanding the RESTful Services we would create in this course
- Step 03 - Creating a Hello World Service
- Step 04 - Enhancing the Hello World Service to return a Bean
- Step 05 - Quick Review of Spring Boot Auto Configuration and Dispatcher Servlet
- Step 06 - Enhancing the Hello World Service with a Path Variable
- Step 07 - Creating User Bean and User Service
- Step 08 - Implementing GET Methods for User Resource
- Step 09 - Implementing POST Method to create User Resource
- Step 10 - Enhancing POST Method to return correct HTTP Status Code and Location
- Step 11 - Implementing Exception Handling - 404 Resource Not Found
- Step 12 - Implementing Generic Exception Handling for all Resources
- Step 13 - Exercise : User Post Resource and Exception Handling
- Step 14 - Implementing DELETE Method to delete a User Resource
- Step 15 - Implementing Validations for RESTful Services
- Step 16 - Implementing HATEOAS for RESTful Services
- Step 17 - Overview of Advanced RESTful Service Features
- Step 18 - Internationalization for RESTful Services
- Step 19 - Content Negotiation - Implementing Support for XML

- Step 20 - Configuring Auto Generation of Swagger Documentation
- Step 21 - Introduction to Swagger Documentation Format
- Step 22 - Enhancing Swagger Documentation with Custom Annotations
- Step 23 - Monitoring APIs with Spring Boot Actuator
- Step 24 - Implementing Static Filtering for RESTful Service
- Step 25 - Implementing Dynamic Filtering for RESTful Service
- Step 26 - Versioning RESTful Services - Basic Approach with URIs
- Step 27 - Versioning RESTful Services - Header and Content Negotiation Approach
- Step 28 - Implementing Basic Authentication with Spring Security
- Step 29 - Overview of Connecting RESTful Service to JPA
- Step 30 - Creating User Entity and some test data
- Step 31 - Updating GET methods on User Resource to use JPA
- Step 32 - Updating POST and DELETE methods on User Resource to use JPA
- Step 33 - Creating Post Entity and Many to One Relationship with User Entity
- Step 34 - Implementing a GET service to retrieve all Posts of a User
- Step 35 - Implementing a POST service to create a Post for a User
- Step 36 - Richardson Maturity Model
- Step 37 - RESTful Web Services - Best Practices

You will learn

- What is a RESTful Web Service?
- How to implement RESTful Web Services with Spring and Spring Boot?
- What are the best practices in designing RESTful Web Services?
- How to design Resources and GET, POST and DELETE operations?
- How to implement Validation for RESTful Web Services?
- How to implement Exception Handling for RESTful Web Services?
- What is HATEOAS? How to implement HATEOAS for a Resource?
- What are the different approach in versioning RESTful Services?
- How to use Postman to execute RESTful Service Requests?
- How to implement basic authentication with Spring Security?
- How to implement filtering for RESTful Services?
- How to monitor RESTful Services with Spring Boot Actuator?
- How to document RESTful Web Services with Swagger?
- How to connect RESTful Services to a backend with JPA?

Useful Links

- POSTMAN - <http://www.getpostman.com>

Links from course examples

- Basic Resources
 - <http://localhost:8080/hello-world>
 - <http://localhost:8080/hello-world-bean>
 - <http://localhost:8080/hello-world/path-variable/Ranga>
 - <http://localhost:8080/users/>
 - <http://localhost:8080/users/1>
- JPA Resources
 - <http://localhost:8080/jpa/users/>
 - <http://localhost:8080/jpa/users/1>
 - <http://localhost:8080/jpa/users/10001/posts>
- Filtering
 - <http://localhost:8080/filtering>
 - <http://localhost:8080/filtering-list>
- Actuator
 - <http://localhost:8080/actuator>
- Versioning
 - <http://localhost:8080/v1/person>
 - <http://localhost:8080/v2/person>
 - <http://localhost:8080/person/param>
 - params=[version=1]
 - <http://localhost:8080/person/param>
- - ◦ params=[version=2]
 - <http://localhost:8080/person/header>
 - headers=[X-API-VERSION=1]
 - <http://localhost:8080/person/header>
 - headers=[X-API-VERSION=2]
 - <http://localhost:8080/person/produces>

- ◦ produces=[application/vnd.company.app-v1+json]
- <http://localhost:8080/person/produces>
 - produces=[application/vnd.company.app-v2+json]
- Swagger
 - <http://localhost:8080/swagger-ui.html>
 - <http://localhost:8080/v2/api-docs>
- H2-Console
 - <http://localhost:8080/h2-console>

Error in the Log

```
Resolved exception caused by Handler execution:
org.springframework.http.converter.HttpMessageNotWritableEx
ception:
No converter found for return value of type:
class
com.in28minutes.rest.webservices.restfulwebservices.HelloWo
rldBean
```

- This happened because there were no getters in HelloWorldBean class

Questions to Answer

- What is dispatcher servlet?
- Who is configuring dispatcher servlet?
- What does dispatcher servlet do?
- How does the HelloWorldBean object get converted to JSON?
- Who is configuring the error mapping?
- Mapping servlet: 'dispatcherServlet' to [/]
- Mapped "{[/hello-world],methods=[GET]}" onto public java.lang.String com.in28minutes.rest.webservices.restfulwebservices.HelloWorldController.helloWorld()
- Mapped "{[/hello-world-bean],methods=[GET]}" onto public com.in28minutes.rest.webservices.restfulwebservices.HelloWorldBean com.in28minutes.rest.webservices.restfulwebservices.HelloWorldController.helloWorldBean()

- Mapped "{[/error]}" onto public
org.springframework.http.ResponseEntity<java.util.Map<java.lang.String,
java.lang.Object>>
org.springframework.boot.autoconfigure.web.servlet.error.BasicErrorController.err
or(javax.servlet.http.HttpServletRequest)
- Mapped "{[/error],produces=[text/html]}" onto public
org.springframework.web.servlet.ModelAndView
org.springframework.boot.autoconfigure.web.servlet.error.BasicErrorController.err
orHtml(javax.servlet.http.HttpServletRequest,javax.servlet.http.HttpServletResponse)

Example Requests

GET <http://localhost:8080/users>

```
[
  {
    "id": 1,
    "name": "Adam",
    "birthDate": "2017-07-19T04:40:20.796+0000"
  },
  {
    "id": 2,
    "name": "Eve",
    "birthDate": "2017-07-19T04:40:20.796+0000"
  },
  {
    "id": 3,
    "name":
      "Jack",
    "birthDate": "2017-07-19T04:40:20.796+0000"
  }
]
```

GET <http://localhost:8080/users/1>

```
{
  "id":
```

```
1,
  "name": "Adam",
  "birthDate": "2017-07-19T04:40:20.796+0000"
}
```

POST <http://localhost:8080/users>

```
{
  "name": "Ranga",
  "birthDate": "2000-07-19T04:29:24.054+0000"
}
```

GET <http://localhost:8080/users/1000>

- Get request to a **non existing resource.**
- The response shows **default error message structure auto configured** by Spring Boot.

```
{
  "timestamp": "2017-07-19T05:28:37.534+0000",
  "status": 404,
  "error": "Not Found",
  "message": "id-500",
  "path": "/users/500"
}
```

GET <http://localhost:8080/users/1000>

- Get request to a non existing resource.
- The response shows a Customized Message Structure

```
{
  "timestamp": "2017-07-19T05:31:01.961+0000",
  "message": "id-500",
  "details": "Any details you would want to add"
}
```

- }

POST <http://localhost:8080/users> with Validation Errors
Request

```
{
  "name": "R",
  "birthDate": "2000-07-19T04:29:24.054+0000"
}
```

Response - 400 Bad Request

```
{
  "timestamp": "2017-07-19T09:00:27.912+0000",
  "message": "Validation Failed",
  "details":
"org.springframework.validation.BeanPropertyBindingResult:
1 errors\nField error in object 'user' on field 'name':
rejected value [R]; codes
[Size.user.name,Size.name,Size.java.lang.String,Size];
arguments
[org.springframework.context.support.DefaultMessageSourceRe
solvable: codes [user.name,name]; arguments []; default
message [name],2147483647,2]; default message [Name should
have atleast 2 characters]"
}
```

GET <http://localhost:8080/users/1> with HATEOAS

```
{
  "id": 1,
  "name": "Adam",
  "birthDate": "2017-07-19T09:26:18.337+0000",
  "_links": {
    "all-users": {
      "href": "http://localhost:8080/users"
    }
  }
}
```

XML Representation of Resources

GET <http://localhost:8080/users>

- Accept application/xml

```
<List>
  <item>
    <id>2</id>
    <name>Eve</name>
    <birthDate>2017-07-19T10:25:20.450+0000</birthDate>
  </item>
  <item>
    <id>3</id>
    <name>Jack</name>
    <birthDate>2017-07-19T10:25:20.450+0000</birthDate>
  </item>
  <item>
    <id>4</id>
    <name>Ranga</name>
    <birthDate>2017-07-19T10:25:20.450+0000</birthDate>
  </item>
</List>
```

POST <http://localhost:8080/users>

- Accept : application/xml
- Content-Type : application/xml

Request

```
<item>
  <name>Ranga</name>
  <birthDate>2017-07-19T10:25:20.450+0000</birthDate>
</item>
```

Response

- Status - 201 Created

Generating Swagger Documentation

```
public static final Contact DEFAULT_CONTACT = new
Contact (
    "Ranga Karanam", "http://www.in28minutes.com",
```

```

    "in28minutes@gmail.com");

    public static final ApiInfo DEFAULT_API_INFO = new
    ApiInfo(
        "Awesome API Title", "Awesome API Description",
        "1.0",
        "urn:tos", DEFAULT_CONTACT,
        "Apache 2.0",
        "http://www.apache.org/licenses/LICENSE-2.0");

    private static final Set<String>
    DEFAULT_PRODUCES_AND_CONSUMES =
        new HashSet<String>(Arrays.asList("application/json",
            "application/xml"));

    @Bean
    public Docket api() {
        return new Docket(DocumentationType.SWAGGER_2)
            .apiInfo(DEFAULT_API_INFO)
            .produces(DEFAULT_PRODUCES_AND_CONSUMES)
            .consumes(DEFAULT_PRODUCES_AND_CONSUMES);
    }

```

Resource Method description

```

    @GetMapping("/users/{id}")
    @ApiOperation(value = "Finds Users by id",
        notes = "Also returns a link to retrieve all users with
        rel - all-users")
    public Resource<User> retrieveUser(@PathVariable int id)
    {

```

API Model

```

    @ApiModel(value="User Details", description="Contains all
    details of a user")
    public class User

```

```

{

    @Size(min=2, message="Name should have at least 2
characters")
    @ApiModelProperty(notes = "Name should have at least 2
characters")
    private String name;

    @Past
    @ApiModelProperty(notes = "Birth Date should be in the
Past")
    private Date birthDate;

```

Filtering

Code

```

@JsonIgnoreProperties(value={"field1"})
public class SomeBean {

    private String field1;

    @JsonIgnore
    private String field2;

    private String field3;

```

Response

```

{
    "field3": "value3"
}

```

Versioning

- Media type versioning (a.k.a “content negotiation” or “accept header”)
 - GitHub
- (Custom) headers versioning
 - Microsoft

- URI Versioning
 - Twitter
- Request Parameter versioning
 - Amazon
- Factors
- URI Pollution
- Misuse of HTTP Headers
- Caching
- Can we execute the request on the browser?
- API Documentation
- No Perfect Solution

More

- https://www.mnot.net/blog/2011/10/25/web_api_versioning_smackdown
- <http://urthen.github.io/2013/05/09/ways-to-version-your-api/>
- <http://stackoverflow.com/questions/389169/best-practices-for-api-versioning>
- <http://www.lexicalscope.com/blog/2012/03/12/how-are-rest-apis-versioned/>
- <https://www.3scale.net/2016/06/api-versioning-methods-a-brief-reference/>

Table Structure

```
create table user (  
  id integer not null,  
  birth_date timestamp,  
  name varchar(255),  
  primary key (id) );  
  
create table post (  
  id integer not null,  
  description varchar(255),  
  user_id integer,  
  primary key (id) );  
  
alter table post  
add constraint post_to_user_foreign_key foreign key  
(user_id) references user;
```

Step 01 - Initializing a RESTful Services Project with Spring Boot

Creating a Spring Project with Spring Initializr is a cake walk.

Spring Initializr <http://start.spring.io/> is great tool to bootstrap your Spring Boot projects.

The screenshot shows the Spring Initializr web application. At the top, it says "SPRING INITIALIZR bootstrap your application now". Below this, there's a header "Generate a Maven Project with Java and Spring Boot 2.0.0 (SNAPSHOT)". The main content is divided into two columns: "Project Metadata" and "Dependencies".

Project Metadata

- Artifact coordinates
- Group:
- Artifact:

Dependencies

- Add Spring Boot Starters and dependencies to your application
- Search for dependencies:
- Selected Dependencies: Web DevTools JPA H2

At the bottom, there is a green button labeled "Generate Project" with a plus icon and a refresh icon.

As shown in the image above, following steps have to be done

- Launch Spring Initializr and choose the following
 - Choose `com.in28minutes.rest.webservices` as Group
 - Choose `restful-web-services` as Artifact
 - Choose Release \geq 2.0.0 (Avoid SNAPSHOT!)
 - Choose following dependencies
 - Web
 - DevTools
 - JPA
 - H2
- Click Generate Project.
- Import the project into Eclipse.
- If you want to understand all the files that are

- part of this project, you can go here.

Step 02 - Understanding the RESTful Services we would create in this course

Social Media Application Resource Mappings

User -> Posts

- Retrieve all Users - GET /users
- Create a User - POST /users
- Retrieve one User - GET /users/{id} -> /users/1
- Delete a User - DELETE /users/{id} -> /users/1
- Retrieve all posts for a User - GET /users/{id}/posts
- Create a posts for a User - POST /users/{id}/posts
- Retrieve details of a post - GET /users/{id}/posts/{post_id}

Step 03 - Creating a Hello World Service

```
@RestController
public class HelloWorldController {

    @GetMapping(path = "/hello-world")
    public String helloWorld() {
        return "Hello World";
    } }
```

Step 04 - Enhancing the Hello World Service to return a Bean

```
@GetMapping(path = "/hello-world-bean")
public HelloWorldBean helloWorldBean() {
    return new HelloWorldBean("Hello World");
}
```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/HelloWorldBean.java New

```
package
```

```
com.in28minutes.rest.webservices.restfulwebservices;  
public class HelloWorldBean {  
  
    private String message;  
  
    public HelloWorldBean(String message) {  
        this.message = message;  
    }  
  
    public String getMessage() {  
        return message;  
    }  
  
    public void setMessage(String message) {  
        this.message = message;  
    }  
  
    @Override  
    public String toString() {  
        return String.format("HelloWorldBean [message=%s]",  
message);  
    }  
}
```

Step 05 - Quick Review of Spring Boot Auto Configuration and Dispatcher Servlet

Let us understand Spring Boot Auto Configuration in depth

- <http://www.springboottutorial.com/spring-boot-auto-configuration>

Step 06 - Enhancing the Hello World Service with a Path Variable

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/HelloWorldController.java

```
package
com.in28minutes.rest.webservices.restfulwebservices;

import org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.PathVariable;
import
org.springframework.web.bind.annotation.RestController;

//Controller
@RestController
public class HelloWorldController {

    @GetMapping(path = "/hello-world")
    public String helloWorld() {
        return "Hello World";
    }

    @GetMapping(path = "/hello-world-bean")
    public HelloWorldBean helloWorldBean() {
        return new HelloWorldBean("Hello World");
    }

    ///hello-world/path-variable/in28minutes
    @GetMapping(path = "/hello-world/path-variable/{name}")
    public HelloWorldBean
helloWorldPathVariable(@PathVariable String name) {
        return new HelloWorldBean(String.format("Hello World,
%s", name));
    }

}
```

/src/main/resources/application.properties Modified

New Lines

```
logging.level.org.springframework = info
```

Step 07 - Creating User Bean and User Service

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/HelloWorldBean.java

Package Change

```
package  
com.in28minutes.rest.webservices.restfulwebservices.helloworld;  
rld;
```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/HelloWorldController.java

Package Change

```
package  
com.in28minutes.rest.webservices.restfulwebservices.helloworld;  
rld;
```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/user/User.java New

```
package  
com.in28minutes.rest.webservices.restfulwebservices.user;  
  
import java.util.Date;  
  
public class User {  
  
    private Integer id;  
  
    private String  
name;  
  
    private Date birthDate;  
  
    public User(Integer id, String name, Date birthDate)
```

```
{
    super();
    this.id = id;
    this.name = name;
    this.birthDate = birthDate;
}

public Integer getId() {
    return id;
}

public void setId(Integer id) {
    this.id = id;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public Date getBirthDate() {
    return birthDate;
}

public void setBirthDate(Date birthDate) {
    this.birthDate = birthDate;
}

@Override
public String toString() {
    return String.format("User [id=%s, name=%s,
```

```
        birthDate=%s]", id, name, birthDate);
    }

}
```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/user/UserDaoService.java New

```
package
com.in28minutes.rest.webservices.restfulwebservices.user;

import java.util.ArrayList;
import java.util.Date;
import java.util.List;

import org.springframework.stereotype.Component;

@Component
public class UserDaoService {
    private static List<User> users = new ArrayList<>();

    private static int usersCount = 3;

    static {
        users.add(new User(1, "Adam", new Date()));
        users.add(new User(2, "Eve", new Date()));
        users.add(new User(3, "Jack", new Date()));
    }

    public List<User> findAll() {
        return users;
    }

    public User save(User user)
    {
        if (user.getId() == null)
```



```

{
    user.setId(++usersCount);
}
users.add(user);
return user;
}

public User findOne(int id) {
    for (User user : users) {
        if (user.getId() == id) {
            return user;
        }
    }
    return null;
}
}

```

Step 08 - Implementing GET Methods for User Resource

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/user/UserResource.java New

```

package
com.in28minutes.rest.webservices.restfulwebservices.user;

import java.util.List;

import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.PathVariable;
import
org.springframework.web.bind.annotation.RestController;

@RestController

```

```

public class UserResource {

    @Autowired
    private UserDaoService service;

    @GetMapping("/users")
    public List<User> retrieveAllUsers() {
        return service.findAll();
    }

    @GetMapping("/users/{id}")
    public User retrieveUser(@PathVariable int id) {
        return service.findOne(id);
    }

}

```

/src/main/resources/application.properties Modified

New Lines

```

#This is not really needed as this is the default after
2.0.0.RELEASE spring.jackson.serialization.write-dates-as-
timestamps=false

```

Step 09 - Implementing POST Method to create User Resource

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/user/UserResource.java Modified

```

// input - details of user
// output - CREATED & Return the created URI
@PostMapping("/users")
public void createUser(@RequestBody User user) {
    User savedUser = service.save(user);
}

```

Step 10 - Enhancing POST Method to return correct HTTP Status Code and Location

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/user/UserResource.java Modified

```
// input - details of user
// output - CREATED & Return the created URI
@PostMapping("/users")
public ResponseEntity<Object> createUser(@RequestBody
User user) {
    User savedUser = service.save(user);
    // CREATED
    // /user/{id}          savedUser.getId()

    URI location = ServletUriComponentsBuilder
        .fromCurrentRequest()
        .path("/{id}")
        .buildAndExpand(savedUser.getId()).toUri();

    return ResponseEntity.created(location).build();
}
```

Step 11 - Implementing Exception Handling - 404 Resource Not Found

Step 12 - Implementing Generic Exception Handling for all Resources

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/exception/CustomizedResponseEntityExceptionHandler.java New

```
package
com.in28minutes.rest.webservices.restfulwebservices.excepti
on;

import java.util.Date;
```

```

import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import
org.springframework.web.bind.annotation.ControllerAdvice;
import
org.springframework.web.bind.annotation.ExceptionHandler;
import
org.springframework.web.bind.annotation.RestController;
import org.springframework.web.context.request.WebRequest;
import
org.springframework.web.servlet.mvc.method.annotation.Respo
nseEntityExceptionHandler;

import
com.in28minutes.rest.webservices.restfulwebservices.user.UserNot
NotFoundException;

@ControllerAdvice
@RestController
public class CustomizedResponseEntityExceptionHandler
extends ResponseEntityExceptionHandler {

    @ExceptionHandler(Exception.class)
    public final ResponseEntity<Object>
handleAllExceptions(Exception ex, WebRequest request) {
        ErrorDetails errorDetails = new ErrorDetails(new
Date(), ex.getMessage(),
            request.getDescription(false));
        return new ResponseEntity(errorDetails,
HttpStatus.INTERNAL_SERVER_ERROR);
    }

    @ExceptionHandler(UserNotFoundException.class)
    public final ResponseEntity<Object>
handleUserNotFoundException(UserNotFoundException ex,
WebRequest request) {
        ErrorDetails errorDetails = new ErrorDetails(new

Date(),

```

```
ex.getMessage(),
        request.getDescription(false));
    return new ResponseEntity(errorDetails,
HttpStatus.NOT_FOUND);
}

}
```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/exception/ErrorDetails.java New

```
package
com.in28minutes.rest.webservices.restfulwebservices.exception;

import java.util.Date;

public class ErrorDetails {
    private Date timestamp;
    private String message;
    private String details;

    public ErrorDetails(Date timestamp, String message,
String details) {
        super();
        this.timestamp = timestamp;
        this.message = message;
        this.details = details;
    }

    public Date getTimestamp() {
        return timestamp;
    }

    public String getMessage() {
        return message;
    }
}
```

```

    }

    public String getDetails() {
        return details;
    }

}

```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/user/UserNotFoundException.java New

```

package
com.in28minutes.rest.webservices.restfulwebservices.user;

import org.springframework.http.HttpStatus;
import
org.springframework.web.bind.annotation.ResponseStatus;

@ResponseStatus(HttpStatus.NOT_FOUND) public class
UserNotFoundException extends RuntimeException {
    public UserNotFoundException(String message) {
        super(message);
    }
}

```

Step 13 - Exercise : User Post Resource and Exception Handling

Step 14 - Implementing DELETE Method to delete a User Resource

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/user/UserDaoService.java Modified

```

public User deleteById(int id) {
    Iterator<User> iterator = users.iterator();
    while (iterator.hasNext()) {
        User user =

```

```

iterator.next();
    if (user.getId() == id) {
        iterator.remove();
        return user;
    }
}
return null;
}

```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservice/user/UserResource.java Modified

```

@GetMapping("/users/{id}")
public User retrieveUser(@PathVariable int id) {
    User user = service.findOne(id);

    if(user==null)
        throw new UserNotFoundException("id-"+ id);

    return user;
}

@DeleteMapping("/users/{id}")
public void deleteUser(@PathVariable int id) {
    User user = service.deleteById(id);

    if(user==null)
        throw new UserNotFoundException("id-"+ id);
}

//
// input - details of user
// output - CREATED & Return the created URI
@PostMapping("/users")
public ResponseEntity<Object> createUser(@RequestBody
User user)

```

```

{
    User savedUser = service.save(user);
    // CREATED
    // /user/{id}      savedUser.getId()

    URI location = ServletUriComponentsBuilder
        .fromCurrentRequest()
        .path("/{id}")
        .buildAndExpand(savedUser.getId()).toUri();

    return ResponseEntity.created(location).build();
}

```

Step 15 - Implementing Validations for RESTful Services

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/exception/CustomizedResponseEntityExceptionHandler.java Modified

```

@Override
protected ResponseEntity<Object>
handleMethodArgumentNotValid(MethodArgumentNotValidException ex,
    HttpHeaders headers, HttpStatus status, WebRequest request) {
    ErrorDetails errorDetails = new ErrorDetails(new Date(),
        "Validation Failed",
        ex.getBindingResult().toString());
    return new ResponseEntity(errorDetails,
        HttpStatus.BAD_REQUEST);
}

```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/user/User.java Modified

```

@Size(min=2, message="Name should have atleast 2
characters")
private String

```



```
name;
```

```
@Past
```

```
private Date birthDate;
```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/user/UserResource.java Modified

```
public ResponseEntity<Object> createUser(@Valid
@RequestBody User user) {
```

Step 16 - Implementing HATEOAS for RESTful Services

/pom.xml Modified

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

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/user/UserResource.java Modified

```
@GetMapping("/users/{id}")
public Resource<User> retrieveUser(@PathVariable int id)
{
    User user = service.findOne(id);

    if(user==null)
        throw new UserNotFoundException("id-"+ id);

    //"all-users", SERVER_PATH + "/users"
    //retrieveAllUsers
    Resource<User> resource = new Resource<User>(user);

    ControllerLinkBuilder linkTo =
```

```

linkTo(methodOn(this.getClass()).retrieveAllUsers()));

resource.add(linkTo.withRel("all-users"));

//HATEOAS

return resource;
}

//HATEOAS

@PostMapping("/users")
public ResponseEntity<Object> createUser(@Valid
@RequestBody User user) {
    User savedUser = service.save(user);
    // CREATED
    // /user/{id}      savedUser.getId()

    URI location = ServletUriComponentsBuilder
        .fromCurrentRequest()
        .path("/{id}")
        .buildAndExpand(savedUser.getId()).toUri();

    return ResponseEntity.created(location).build();
}

```

Step 17 - Overview of Advanced RESTful Service Features

- Step 18 - Internationalization for RESTful Services
- Step 19 - Content Negotiation - Implementing Support for XML
- Step 20 - Configuring Auto Generation of Swagger Documentation
- Step 21 - Introduction to Swagger Documentation Format
- Step 22 - Enhancing Swagger Documentation with Custom Annotations

- Step 23 - Monitoring APIs with Spring Boot Actuator
- Step 24 - Implementing Static Filtering for RESTful Service
- Step 25 - Implementing Dynamic Filtering for RESTful Service
- Step 26 - Versioning RESTful Services - Basic Approach with URIs
- Step 27 - Versioning RESTful Services - Header and Content Negotiation Approach
- Step 28 - Implementing Basic Authentication with Spring Security

Step 18 - Internationalization for RESTful Services

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/RestfulWebServicesApplication.java Modified

New Lines

```
import java.util.Locale; import
org.springframework.context.annotation.Bean;
import
org.springframework.context.support.ResourceBundleMessageSo
urce;
import org.springframework.web.servlet.LocaleResolver;
import
org.springframework.web.servlet.i18n.SessionLocaleResolver;

@Bean
public LocaleResolver localeResolver() {
    SessionLocaleResolver localeResolver = new
SessionLocaleResolver();
    localeResolver.setDefaultLocale(Locale.US);
    return localeResolver;
}

@Bean
public ResourceBundleMessageSource messageSource() {
    ResourceBundleMessageSource messageSource = new
ResourceBundleMessageSource();
    messageSource.setBasename("messages");
    return messageSource;
}
```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/helloworld/HelloWorldController.java Modified

New Lines

```
import java.util.Locale;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.MessageSource;
import
org.springframework.web.bind.annotation.RequestHeader;

@Autowired private MessageSource messageSource;

@GetMapping(path = "/hello-world-internationalized")
public String helloWorldInternationalized(
    @RequestHeader(name="Accept-Language", required=false)
    Locale locale) {
    return messageSource.getMessage("good.morning.message",
    null, locale);
}
```

/src/main/resources/messages.properties New

```
good.morning.message=Good Morning
```

/src/main/resources/messages_fr.properties New

```
good.morning.message=Bonjour
```

/src/main/resources/messages_nl.properties New

```
good.morning.message=Goede Morgen
```

Step 18 Part 2 - Simplifying Internationalization for RESTful Services

Use AcceptHeaderLocaleResolver

```

@SpringBootApplication
public class RestfulWebServicesApplication {

    ....

    @Bean
    public LocaleResolver localeResolver() {
        AcceptHeaderLocaleResolver localeResolver = new
AcceptHeaderLocaleResolver();
        localeResolver.setDefaultLocale(Locale.US);
        return localeResolver;
    }

```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/helloworld/HelloWorldController.java

```

@GetMapping(path = "/hello-world-internationalized")
public String helloWorldInternationalized() {
    return messageSource.getMessage("good.morning.message",
                                     null,
                                     LocaleContextHolder.getLocale());
}

```

Use MessageSource configuration from application.properties

```
spring.messages.basename=messages
```

Step 19 - Content Negotiation - Implementing Support for XML

/pom.xml Modified

New Lines

```

<dependency>
    <groupId>com.fasterxml.jackson.dataformat</groupId>
    <artifactId>jackson-dataformat-xml</artifactId>
</dependency>

```

Step 20 - Configuring Auto Generation of Swagger Documentation

Step 21 - Introduction to Swagger Documentation Format

Step 22 - Enhancing Swagger Documentation with Custom Annotations

/pom.xml Modified

New Lines

```
<dependency>
  <groupId>io.springfox</groupId>
  <artifactId>springfox-swagger2</artifactId>
  <version>2.4.0</version>
</dependency>

<dependency>
  <groupId>io.springfox</groupId>
  <artifactId>springfox-swagger-ui</artifactId>
  <version>2.4.0</version>
</dependency>
```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/SwaggerConfig.java New

```
package
com.in28minutes.rest.webservices.restfulwebservices;
import java.util.Arrays; import java.util.HashSet;
import java.util.Set;

import org.springframework.context.annotation.Bean;
import
org.springframework.context.annotation.Configuration;

import springfox.documentation.service.ApiInfo;
```

```
import springfox.documentation.service.Contact;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import
springfox.documentation.swagger2.annotations.EnableSwagger2
;

@Configuration
@EnableSwagger2
public class SwaggerConfig {

    public static final Contact DEFAULT_CONTACT = new
Contact(
        "Ranga Karanam", "http://www.in28minutes.com",
        "in28minutes@gmail.com");

    public static final ApiInfo DEFAULT_API_INFO = new
ApiInfo(
        "Awesome API Title", "Awesome API Description",
        "1.0",
        "urn:tos", DEFAULT_CONTACT,
        "Apache 2.0",
        "http://www.apache.org/licenses/LICENSE-2.0");

    private static final Set<String>
DEFAULT_PRODUCES_AND_CONSUMES =
        new HashSet<String>(Arrays.asList("application/json",
            "application/xml"));

    @Bean
    public Docket api() {
        return new Docket(DocumentationType.SWAGGER_2)
            .apiInfo(DEFAULT_API_INFO)
            .produces(DEFAULT_PRODUCES_AND_CONSUMES)
            .consumes(DEFAULT_PRODUCES_AND_CONSUMES);
    }
}
```

```
}
```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/UserApiDocumentationConfig.java New

```
package
com.in28minutes.rest.webservices.restfulwebservices;

import io.swagger.annotations.Contact;
import io.swagger.annotations.ExternalDocs;
import io.swagger.annotations.Info;
import io.swagger.annotations.License; import
io.swagger.annotations.SwaggerDefinition;

@SwaggerDefinition(
    info = @Info(
        description = "Awesome Resources",
        version = "V12.0.12",
        title = "Awesome Resource API",
        contact = @Contact(
            name = "Ranga Karanam",
            email = "ranga.karanam@in28minutes.com",
            url = "http://www.in28minutes.com"
        ),
        license = @License(
            name = "Apache 2.0",
            url =
"http://www.apache.org/licenses/LICENSE-2.0"
        ),
        consumes = {"application/json", "application/xml"},
        produces = {"application/json", "application/xml"},
        schemes = {SwaggerDefinition.Scheme.HTTP,
SwaggerDefinition.Scheme.HTTPS},
        externalDocs = @ExternalDocs(value = "Read This For
Sure", url = "http://in28minutes.com")
    )
)
```



```
)  
public interface UserApiDocumentationConfig {  
  
}
```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/user/User.java Modified

```
@ApiModelProperty(description="All details about the user. ")  
public class User {  
  
    private Integer id;  
  
    @Size(min=2, message="Name should have at least 2  
characters")  
    @ApiModelProperty(notes="Name should have at least 2  
characters")  
    private String name;  
  
    @Past  
    @ApiModelProperty(notes="Birth date should be in the  
past")  
    private Date birthDate;
```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/user/UserResource.java Modified

```
    @GetMapping("/users/{id}")  
    public Resource<User> retrieveUser(@PathVariable int id)  
    {  
        User user = service.findOne(id);  
  
        if(user==null)  
            throw new UserNotFoundException("id-"+ id);  
  
        // "all-users", SERVER_PATH +
```

```

"/users"
    //retrieveAllUsers
    Resource<User> resource = new Resource<User>(user);

    ControllerLinkBuilder linkTo =

linkTo(methodOn(this.getClass()).retrieveAllUsers());

    resource.add(linkTo.withRel("all-users"));

    //HATEOAS

    return resource;
}

```

Step 23 - Monitoring APIs with Spring Boot Actuator

/pom.xml Modified

New Lines

```

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>

<dependency>
    <groupId>org.springframework.data</groupId>
    <artifactId>spring-data-rest-hal-browser</artifactId>
</dependency>

```

application.properties Modified

```
management.endpoints.web.exposure.include=*
```

Step 24 - Implementing Static Filtering for RESTful Service

Step 25 - Implementing Dynamic Filtering for RESTful Service

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/UserApiDocumentationConfig.java Deleted

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/filtering/FilteringController.java New

```
package
com.in28minutes.rest.webservices.restfulwebservices.filtering;

import java.util.Arrays;
import java.util.List;

import
org.springframework.http.converter.json.MappingJacksonValue
;
import org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.RestController;

import com.fasterxml.jackson.databind.ser.FilterProvider;
import
com.fasterxml.jackson.databind.ser.impl.SimpleBeanPropertyF
ilter; import
com.fasterxml.jackson.databind.ser.impl.SimpleFilterProvide
r;

@RestController public class FilteringController {

    // field1,field2
    @GetMapping("/filtering")
    public MappingJacksonValue retrieveSomeBean() {
        SomeBean someBean = new SomeBean("value1", "value2",
"value3");

        SimpleBeanPropertyFilter filter =
SimpleBeanPropertyFilter.filterOutAllExcept("field1",
```

```

        "field2");

        FilterProvider filters = new
SimpleFilterProvider().addFilter("SomeBeanFilter", filter);

        MappingJacksonValue mapping = new
MappingJacksonValue(someBean);

        mapping.setFilters(filters);

        return mapping;
    }

    // field2, field3
    @GetMapping("/filtering-list")
    public MappingJacksonValue retrieveListOfSomeBeans() {
        List<SomeBean> list = Arrays.asList(new
SomeBean("value1", "value2", "value3"),
        new SomeBean("value12", "value22", "value32"));

        SimpleBeanPropertyFilter filter =
SimpleBeanPropertyFilter.filterOutAllExcept("field2",
"field3");

        FilterProvider filters = new
SimpleFilterProvider().addFilter("SomeBeanFilter", filter);

        MappingJacksonValue mapping = new
MappingJacksonValue(list);

        mapping.setFilters(filters);

        return mapping;
    }
}

```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/filtering/SomeBean.java New

```
package
com.in28minutes.rest.webservices.restfulwebservices.filtering;

import com.fasterxml.jackson.annotation.JsonFilter;
@JsonFilter("SomeBeanFilter")
public class SomeBean {

    private String field1;

    private String field2;

    private String field3;

    public SomeBean(String field1, String field2, String
field3) {
        super();
        this.field1 = field1;
        this.field2 = field2;
        this.field3 = field3;
    }

    public String getField1() {
        return field1;
    }

    public void setField1(String field1) {
        this.field1 = field1;
    }

    public String getField2() {
        return field2;
    }
}
```

```

}

public void setField2(String field2) {
    this.field2 = field2;
}

public String getField3() {
    return field3;
}

public void setField3(String field3) {
    this.field3 = field3;
}

}

```

Step 26 - Versioning RESTful Services - Basic Approach with URIs

Step 27 - Versioning RESTful Services - Header and Content Negotiation Approach

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/versioning/Name.java New

```

package
com.in28minutes.rest.webservices.restfulwebservices.versioning;

public class Name {
    private String firstName;
    private String lastName;

    public Name() {
    }

    public Name(String firstName, String lastName) {
    }
}

```

```

super();
    this.firstName = firstName;
    this.lastName = lastName;
}

public String getFirstName() {
    return firstName;
}

public void setFirstName(String firstName) {
    this.firstName = firstName;
}

public String getLastName() {
    return lastName;
}

public void setLastName(String lastName) {
    this.lastName = lastName;
}
}

```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/versioning/PersonV1.java New

```

package
com.in28minutes.rest.webservices.restfulwebservices.versioning;

public class PersonV1 {
    private String name;

    public PersonV1() {
        super();
    }

    public PersonV1(String name)

```

```

{

super();

    this.name = name;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

}

```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/versioning/PersonV2.java New

```

package
com.in28minutes.rest.webservices.restfulwebservices.versioning;

public class PersonV2 {
    private Name name;

    public PersonV2() {
        super();
    }

    public PersonV2(Name name) {
        super();
        this.name = name;
    }

    public Name getName()

```



```

{
    return name;

}

public void setName(Name name) {
    this.name = name;
}

}

```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/versioning/PersonVersioningController.java New

```

package
com.in28minutes.rest.webservices.restfulwebservices.versioning;

import org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.RestController;
@RestController
public class PersonVersioningController {

    @GetMapping("v1/person")
    public PersonV1 personV1() {
        return new PersonV1("Bob Charlie");
    }

    @GetMapping("v2/person")
    public PersonV2 personV2() {
        return new PersonV2(new Name("Bob", "Charlie"));
    }

    @GetMapping(value = "/person/param", params =
"version=1")
    public PersonV1 paramV1()

```

```
{
    return new PersonV1("Bob Charlie");
}

@GetMapping(value = "/person/param", params =
"version=2")
public PersonV2 paramV2() {
    return new PersonV2(new Name("Bob", "Charlie"));
}

@GetMapping(value = "/person/header", headers = "X-API-
VERSION=1")
public PersonV1 headerV1() {
    return new PersonV1("Bob Charlie");
}

@GetMapping(value = "/person/header", headers = "X-API-
VERSION=2")
public PersonV2 headerV2() {
    return new PersonV2(new Name("Bob", "Charlie"));
}

@GetMapping(value = "/person/produces", produces =
"application/vnd.company.app-v1+json")
public PersonV1 producesV1() {
    return new PersonV1("Bob Charlie");
}

@GetMapping(value = "/person/produces", produces =
"application/vnd.company.app-v2+json")
public PersonV2 producesV2() {
    return new PersonV2(new Name("Bob", "Charlie"));
}
```

```
}
```

Step 28 - Implementing Basic Authentication with Spring Security

/pom.xml Modified

New Lines

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

/src/main/resources/application.properties Modified

New Lines

```
spring.security.filter.dispatcher-types=request
spring.security.user.name=username
spring.security.user.password=password
```

Step 29 - Overview of Connecting RESTful Service to JPA

- Step 30 - Creating User Entity and some test data
- Step 31 - Updating GET methods on User Resource to use JPA
- Step 32 - Updating POST and DELETE methods on User Resource to use JPA
- Step 33 - Creating Post Entity and Many to One Relationship with User Entity
- Step 34 - Implementing a GET service to retrieve all Posts of a User
- Step 35 - Implementing a POST service to create a Post for a User

Step 30 - Creating User Entity and some test data

Step 31 - Updating GET methods on User Resource to use JPA

Step 32 - Updating POST and DELETE methods on User Resource to use JPA

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/user/User.java Modified

New Lines

```
@ApiModelProperty(description="All details about the user. ")

@Entity
public class User {

    @Id
    @GeneratedValue
    private Integer id;
```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/user/UserJPAResource.java New

```
package
com.in28minutes.rest.webservices.restfulwebservices.user;

import static
org.springframework.hateoas.mvc.ControllerLinkBuilder.linkTo;
import static
org.springframework.hateoas.mvc.ControllerLinkBuilder.methodOn;

import java.net.URI;
import java.util.List;
import java.util.Optional;

import javax.validation.Valid;

import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.hateoas.Resource;
import org.springframework.hateoas.mvc.ControllerLinkBuilder;
import org.springframework.http.ResponseEntity;
import
org.springframework.web.bind.annotation.DeleteMapping;
```

```
import org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;

import
org.springframework.web.bind.annotation.RestController;
import
org.springframework.web.servlet.support.ServletUriComponent
sBuilder;

@RestController
public class UserJPAResource {

    @Autowired
    private UserDaoService service;

    @Autowired
    private UserRepository  userRepository;

    @GetMapping("/jpa/users")
    public List<User> retrieveAllUsers() {
        return userRepository.findAll();
    }

    @GetMapping("/jpa/users/{id}")
    public Resource<User> retrieveUser(@PathVariable int id)
    {
        Optional<User> user = userRepository.findById(id);

        if(!user.isPresent())
            throw new UserNotFoundException("id-"+ id);

        //"all-users", SERVER_PATH + "/users"
        //retrieveAllUsers
        Resource<User> resource = new
```

```

Resource<User>(user.get());

    ControllerLinkBuilder linkTo =

linkTo(methodOn(this.getClass()).retrieveAllUsers());

    resource.add(linkTo.withRel("all-users"));

    //HATEOAS

    return resource;
}

@DeleteMapping("/jpa/users/{id}")
public void deleteUser(@PathVariable int id) {
    User user = service.deleteById(id);

    if(user==null)
        throw new UserNotFoundException("id-"+ id);
}

//
// input - details of user
// output - CREATED & Return the created URI

//HATEOAS

@PostMapping("/jpa/users")
public ResponseEntity<Object> createUser(@Valid
@RequestBody User user) {
    User savedUser = service.save(user);

    URI location = ServletUriComponentsBuilder
        .fromCurrentRequest()

```

```

        .path("/{id}")
        .buildAndExpand(savedUser.getId()).toUri();

        return ResponseEntity.created(location).build();

    }

}

```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/user/UserRepository.java New

```

package
com.in28minutes.rest.webservices.restfulwebservices.user;

import
org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface UserRepository extends JpaRepository<User,
Integer>{

}

```

/src/main/resources/application.properties Modified

New Lines

```

management.endpoints.web.exposure.include=*
spring.jpa.show-sql=true
spring.h2.console.enabled=true

```

/src/main/resources/data.sql New

```

insert into user values(1, sysdate(), 'AB'); insert into
user values(2, sysdate(), 'Jill');
insert into user values(3, sysdate(), 'Jam');

```

Step 33 - Creating Post Entity and Many to One Relationship

with User Entity

Step 34 - Implementing a GET service to retrieve all Posts of a User

Step 35 - Implementing a POST service to create a Post for a User

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/user/Post.java New

```
package
com.in28minutes.rest.webservices.restfulwebservices.user;

import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue; import
javax.persistence.Id;
import javax.persistence.ManyToOne;

import com.fasterxml.jackson.annotation.JsonIgnore;

@Entity
public class Post {

    @Id
    @GeneratedValue
    private Integer id;
    private String description;

    @ManyToOne(fetch=FetchType.LAZY)
    @JsonIgnore
    private User user;

    public Integer getId() {
        return
```



```

id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }

    @Override
    public String toString() {
        return String.format("Post [id=%s, description=%s]",
id, description);
    }

}

```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/user/PostRepository.java New

```
package
```

```
com.in28minutes.rest.webservices.restfulwebservices.user;

import
org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository

public interface PostRepository extends JpaRepository<Post,
Integer>{

}
```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/user/User.java Modified

```
@OneToMany(mappedBy="user")
private List<Post> posts;
```

/src/main/java/com/in28minutes/rest/webservices/restfulwebservices/user/UserJPAResource.java Modified

```
@RestController
public class UserJPAResource {

    @Autowired
    private UserRepository userRepository;

    @Autowired
    private PostRepository postRepository;

    @GetMapping("/jpa/users")
    public List<User> retrieveAllUsers() {
        return userRepository.findAll();
    }
}
```

```

}

@GetMapping("/jpa/users/{id}")
public Resource<User> retrieveUser(@PathVariable int id)
{
    Optional<User> user = userRepository.findById(id);

    if (!user.isPresent())
        throw new UserNotFoundException("id-" +

id);

    // "all-users", SERVER_PATH + "/users"
    // retrieveAllUsers
    Resource<User> resource = new Resource<User>
(user.get());

    ControllerLinkBuilder linkTo =
linkTo(methodOn(this.getClass()).retrieveAllUsers());

    resource.add(linkTo.withRel("all-users"));

    // HATEOAS

    return resource;
}

@DeleteMapping("/jpa/users/{id}")
public void deleteUser(@PathVariable int id) {
    userRepository.deleteById(id);
}

//
// input - details of user
// output - CREATED & Return the created

```

URI

```
// HATEOAS
```

```
@PostMapping("/jpa/users")
public ResponseEntity<Object> createUser(@Valid
@RequestBody User user) {
    User savedUser = userRepository.save(user);

    URI location =
ServletUriComponentsBuilder.fromCurrentRequest().path("/{id}").buildAndExpand(savedUser.getId())

.toUri();

    return ResponseEntity.created(location).build();
}

@GetMapping("/jpa/users/{id}/posts")
public List<Post> retrieveAllUsers(@PathVariable int id)
{
    Optional<User> userOptional =
userRepository.findById(id);

    if(!userOptional.isPresent()) {
        throw new UserNotFoundException("id-" + id);
    }

    return userOptional.get().getPosts();
}

@PostMapping("/jpa/users/{id}/posts")
public ResponseEntity<Object> createPost(@PathVariable
int id, @RequestBody Post post)
```

```

{

    Optional<User> userOptional =
userRepository.findById(id);

    if(!userOptional.isPresent()) {
        throw new UserNotFoundException("id-" + id);
    }

    User user = userOptional.get();

    post.setUser(user);

    userRepository.save(post);

    URI location =
ServletUriComponentsBuilder.fromCurrentRequest().path("/{id
}").buildAndExpand(post.getId())
        .toUri();

    return ResponseEntity.created(location).build();

}

}

```

/src/main/resources/data.sql Modified

New Lines

```

insert into user values(10001, sysdate(), 'AB');
insert into user values(10002, sysdate(), 'Jill');
insert into user values(10003, sysdate(), 'Jam');
insert into post values(11001, 'My First Post', 10001);
insert into post values(11002, 'My Second Post', 10001);

```

Best Practices with REST

Richardson Maturity Model

Level 0

Expose SOAP web services in REST style

- <http://server/getPosts>
- <http://server/deletePosts>
- <http://server/doThis>

Level 1

- Expose Resources with proper URI
 - <http://server/accounts>
 - <http://server/accounts/10>
- Improper use of HTTP Methods

Level 2

- Level 1 + HTTP Methods

Level 3

- Level 2 + HATEOAS
 - Data + Next Possible Actions

Best Practices in RESTful Design

- Consumer First
- Make best use of HTTP
 - Request Methods
 - GET
 - POST
 - PUT

- ○ ○ DELETE
- Response Status
 - 200 - SUCCESS
 - 404 - RESOURCE NOT FOUND
 - 400 - BAD REQUEST
 - 201 - CREATED
 - 401 - UNAUTHORIZED
 - 500 - SERVER ERROR
- No Secure Info in URI
- Use Plurals
 - Prefer /users to /user
 - Prefer /users/1 to /user/1
- Use Nouns for Resources
- For Exceptions
 - Define a Consistent Approach
 - /search
 - PUT /gists/{id}/star
 - DELETE /gists/{id}/star
- Consumer First
- Define Organizational Standards
 - YARAS - <https://github.com/darrin/yaras>
 - Naming Resources
 - Request Response Structures
 - Common Features Standardization
 - Error Handling
 - Versioning
 - Searching
 - Filtering
 - Support for Mock Responses
 - HATEOAS
- Build a Framework
- Focus on Decentralized Governance