

CDP : Spring boot Developer

- Define and demonstrate Spring boot
- Developing and deploying application using spring boot
- Design and develop web applications using spring boot
- Develop and deploy applications using JPA and Spring Boot.
- Develop and deploy rest application
- Define and Demonstrate Actuator
- Define and demonstrate HAL
- Define and Demonstrate spring boot messaging (RabbitMQ)
- Develop and deploy application for Spring boot documentation
- Developing end to end application using spring boot
- Developing and deploying cloud native application.

| CDP : Infogain Certified Spring Boot Developer Schedule | | |
|---|-----------|----------------|
| Date | Day | Topic |
| 18-Jul-18 | Wednesday | Spring Boot -1 |
| 20-Jul-18 | Friday | Spring Boot -2 |
| 26-Jul-18 | Thursday | Spring Boot-3 |
| TBD | | |

Session 2 - Objective

- Develop and deploy applications using JPA and Spring Boot.
- Develop and deploy rest application
- Define and Demonstrate Actuator
- Define and demonstrate HAL

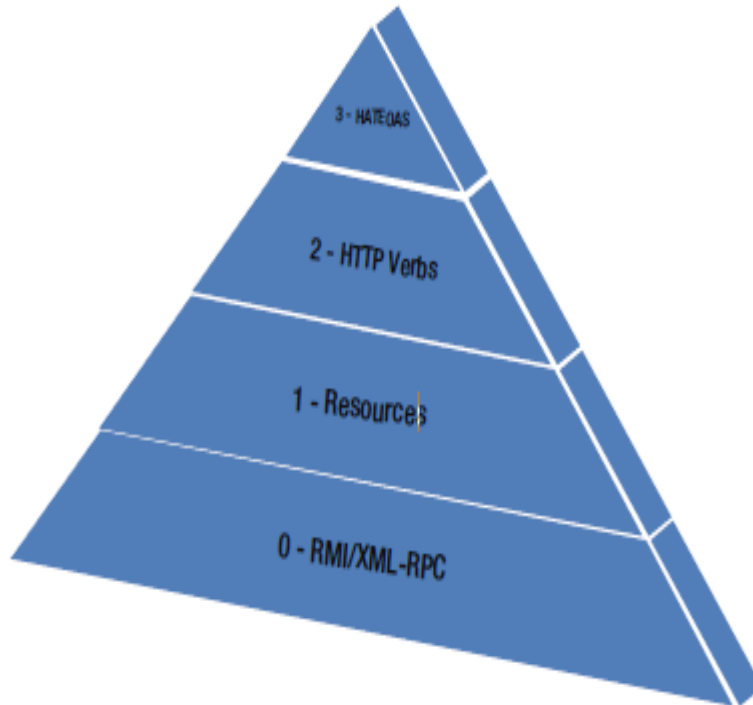
- Spring provides the `@PropertySource` annotation to specify the list of configuration files.
- Spring Boot takes it one step further by automatically registering a **PropertyPlaceholderConfigurer** bean using the **application.properties** file in the root classpath by default.
- You can also create profile specific configuration files using the filename as `application-{profile}.properties`

- Without using Spring Boot, you need to configure various beans like **DataSource**, **TransactionManager**, **LocalContainerEntityManagerFactoryBean**, etc. by yourself.
- You can use the Spring Boot JPA Starter **spring-boot-starter-data-jpa** to quickly get up and running with JPA
- Spring Data provides various repository abstractions, such as
- **CrudRepository**, **PagingAndSortingRepository**, **JpaRepository**, etc. They provide out-of-the-box support for CRUD operations, as well as pagination and sorting
- By defining a User **findByEmail(String email)** method, Spring Data will automatically generate the query with a where clause, as in "where email = ?1".
- By defining a User **findByEmailAndPassword(String email, String password)** method, Spring Data will automatically generate the query with a where clause, as in "where email = ?1 and password=?2".

Using Spring Data with Spring Boot :

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

- **Richardson's Maturity Model:** The Richardson's Maturity Model (RMM), developed by Leonard Richardson, classifies REST-based Web services on how well they adhere to REST principles.



Level Zero

This is the most rudimentary maturity level for a service. Services in this level use HTTP as the transport mechanism and perform remote procedure calls on a single URI. Typically, POST or GET HTTP methods are employed for service calls. SOAP- and XML-RPC-based Web services fall under this level.

Level One

The next level adheres to the REST principles more closely and introduces multiple URIs, one per resource. Complex functionality of a large service endpoint is broken down into multiple resources. However, services in this layer use one HTTP verb, typically POST, to perform all of the operations.

Level Two

Services in this level leverage HTTP protocol and make the right use of HTTP verbs and status codes available in the protocol. Web services implementing CRUD operations are good examples of Level 2 services.

Level Three

This is the most mature level for a service and is built around the notion of Hypermedia as the Engine of Application State, or HATEOAS. Services in this level allow discoverability by providing responses that contain links to other related resources and controls that tell the client what to do next.

- **SpringMVC** provides first-class support for building RESTful web services. As Spring's REST support is built on top of **SpringMVC**, you can leverage the knowledge of SpringMVC for building REST APIs.
- **Spring Data REST** is a spring portfolio project that can be used to expose Spring Data repositories as REST endpoints.
- You can expose Spring Data JPA, Spring Data Mongo, and Spring Data Cassandra repositories as REST endpoints without much effort.
- **Sorting and Pagination** If the Repository extends **PagingAndSortingRepository**, then Spring Data REST endpoints support pagination and sorting out-of-the-box.

Following the REST principles, you can use the following HTTP verbs:

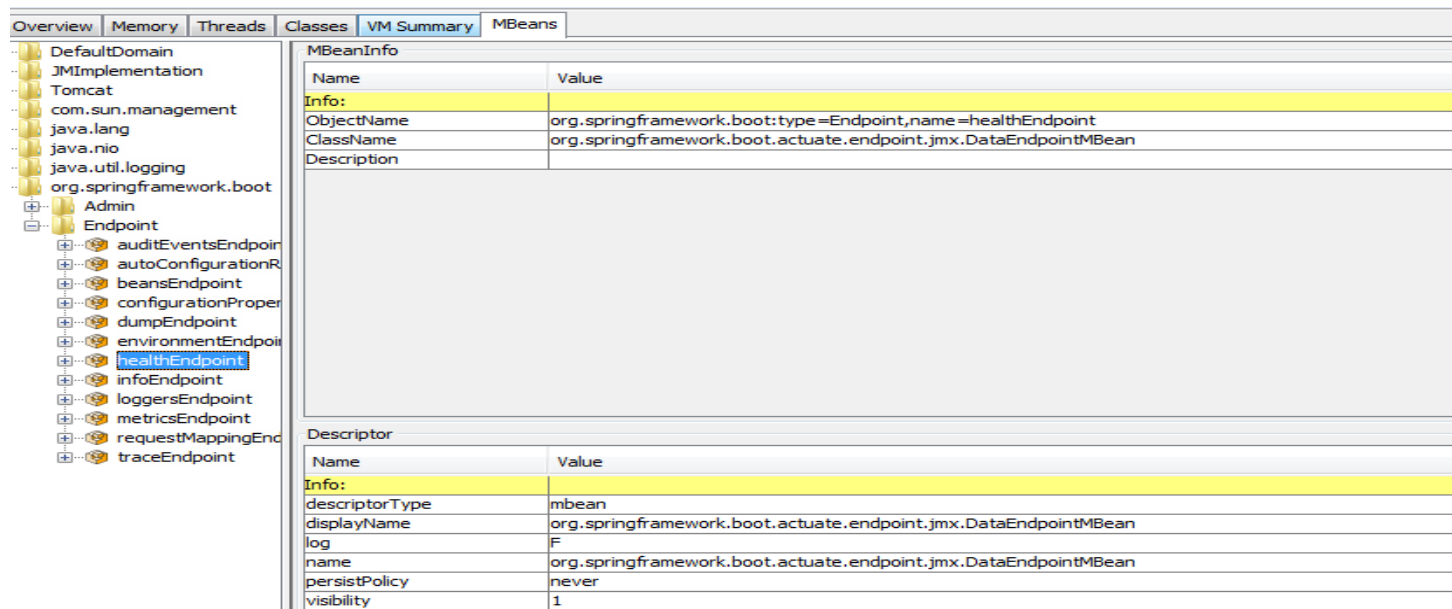
- GET—To get a collection or a single resource
- POST—To create a new resource
- PUT—To update an existing resource
- DELETE—To delete a collection or a single resource

Now consider how you can define URIs for a blog system's resources:

- GET—<http://localhost:8080/users/>: Returns a list of all users
- GET—<http://localhost:8080/users/2>: Returns a user whose ID is 2
- POST—<http://localhost:8080/users/>: Creates a new User resource
- PUT—<http://localhost:8080/users/2>: Updates a POST resource whose ID is 2
- DELETE—<http://localhost:8080/users/2>: Deletes a POST resource whose ID is 2

Spring Boot actuators

- Spring Boot actuators provide an excellent out-of-the-box mechanism to monitor and manage Spring Boot applications in production.
 - Create another **Spring Starter Project**
 - This time, select **Web** and **Actuators** under **Ops**.
 - Point the browser to localhost:8080/actuator. This will open the HAL browser.
 - [management.security.enabled=false](#) (update /add in application.properties)
- Monitoring using Jconsole:
 - Start **jconsole** and browse under mbean



| Name | Value |
|--------------|---|
| Info: | |
| ObjectName | org.springframework.boot:type=Endpoint,name=healthEndpoint |
| ClassName | org.springframework.boot.actuate.endpoint.jmx.DataEndpointMBean |
| Description | |

| Name | Value |
|----------------|---|
| Info: | |
| descriptorType | mbean |
| displayName | org.springframework.boot.actuate.endpoint.jmx.DataEndpointMBean |
| log | F |
| name | org.springframework.boot.actuate.endpoint.jmx.DataEndpointMBean |
| persistPolicy | never |
| visibility | 1 |



Thank You



Infogain Corporation, HQ

485 Alberto Way Los Gatos,
CA 95032 USA
Phone: 408-355-6000
Fax: 408-355-7000

Pune

7th Floor, Bhalerao Towers, CTS No.1669 -
1670, Behind Hotel Pride,
Shivaji Nagar, Pune - 411005
Phone : +91-20-66236700

Infogain Irvine

41 Corporate Park,
Suite 390 Irvine, CA 2606 USA
Phone: 949-223-5100
Fax: 949-223-5110

Infogain Austin

Stratum Executive Center Building D
11044 Research Boulevard Suite 200
Austin, Texas 78759

Noida

A-16, Sector 60, Noida Gautam Budh agar,
201301 (U.P.) India
Phone: +91-120-2445144
Fax: +91-120-2580406

Dubai

P O Box 500588 Office No.105,
Building No. 4, Dubai Outsource Zone,
Dubai, United Arab Emirates
Tel: +971-4-458-7336