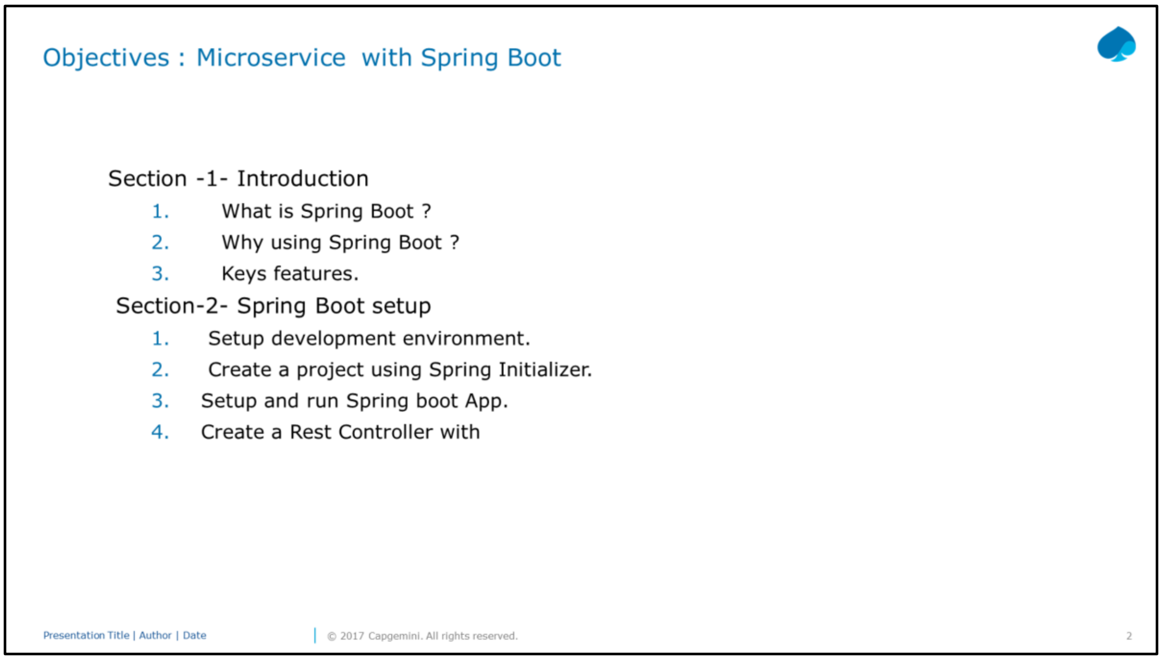
Add instructor notes here.

Microservices With Spring Boot

Capgemini



Objectives : Microservice with Spring Boot

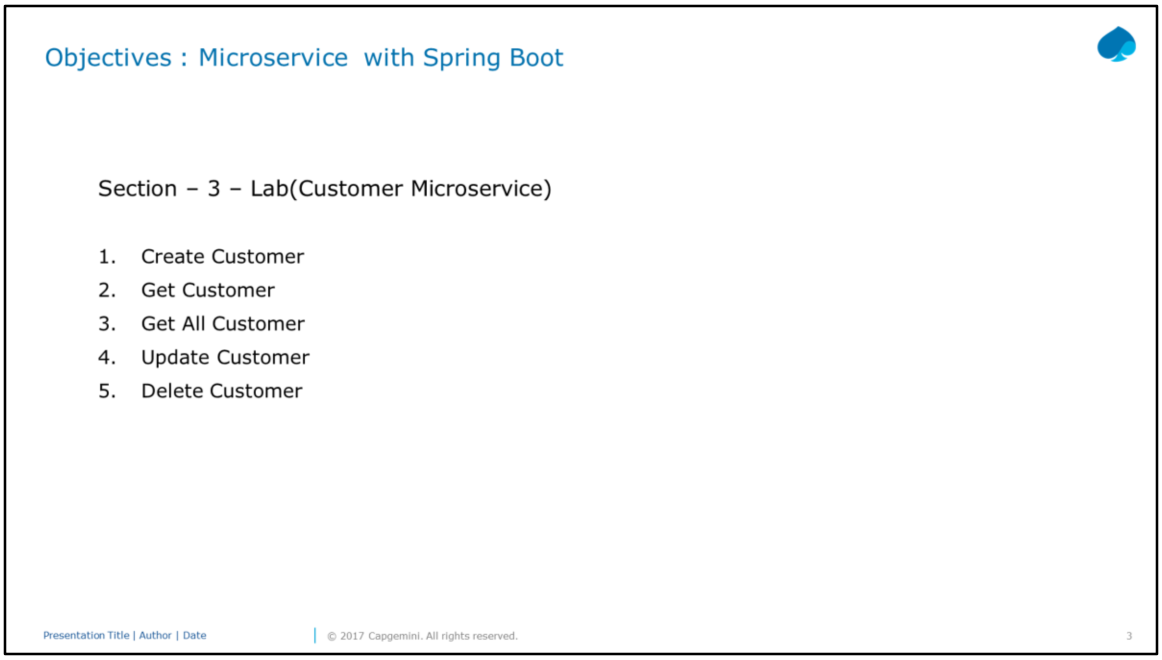
Section -1- Introduction

1. What is Spring Boot ?
2. Why using Spring Boot ?
3. Keys features.

Section-2- Spring Boot setup

1. Setup development environment.
2. Create a project using Spring Initializer.
3. Setup and run Spring boot App.
4. Create a Rest Controller with

© 2017 Capgemini. All rights reserved.



Objectives : Microservice with Spring Boot

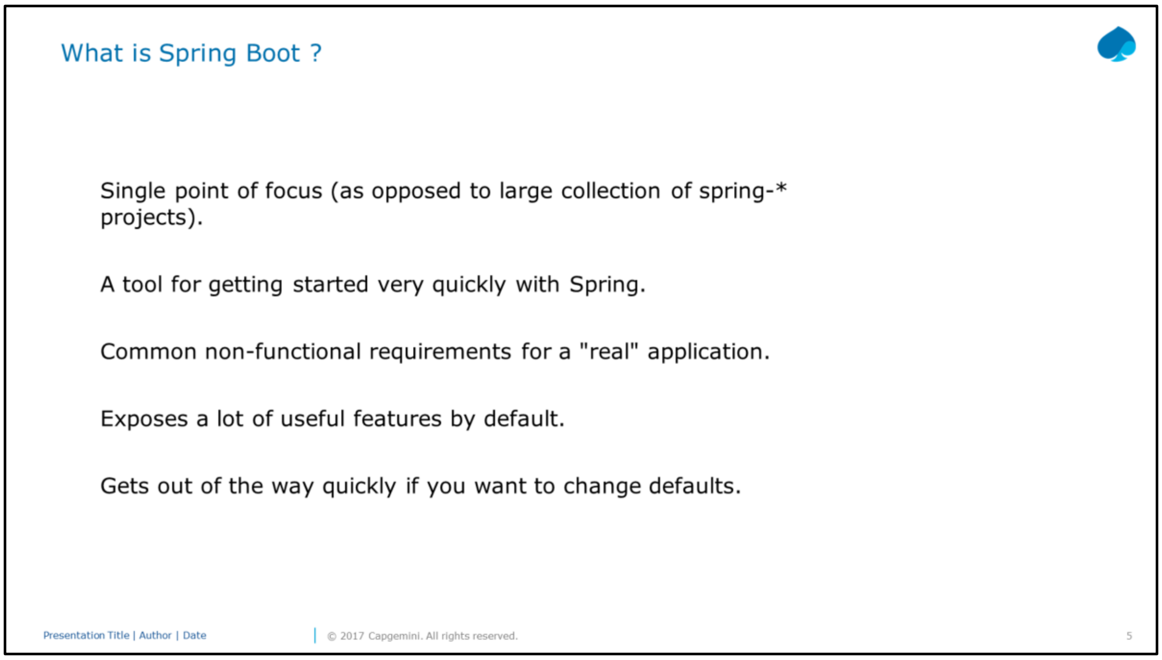
Section – 3 – Lab(Customer Microservice)

1. Create Customer
2. Get Customer
3. Get All Customer
4. Update Customer
5. Delete Customer

© 2017 Capgemini. All rights reserved.

Section -1
Introduction Spring Boot

© 2017 Capgemini. All rights reserved.



What is Spring Boot ?

Single point of focus (as opposed to large collection of spring-* projects).

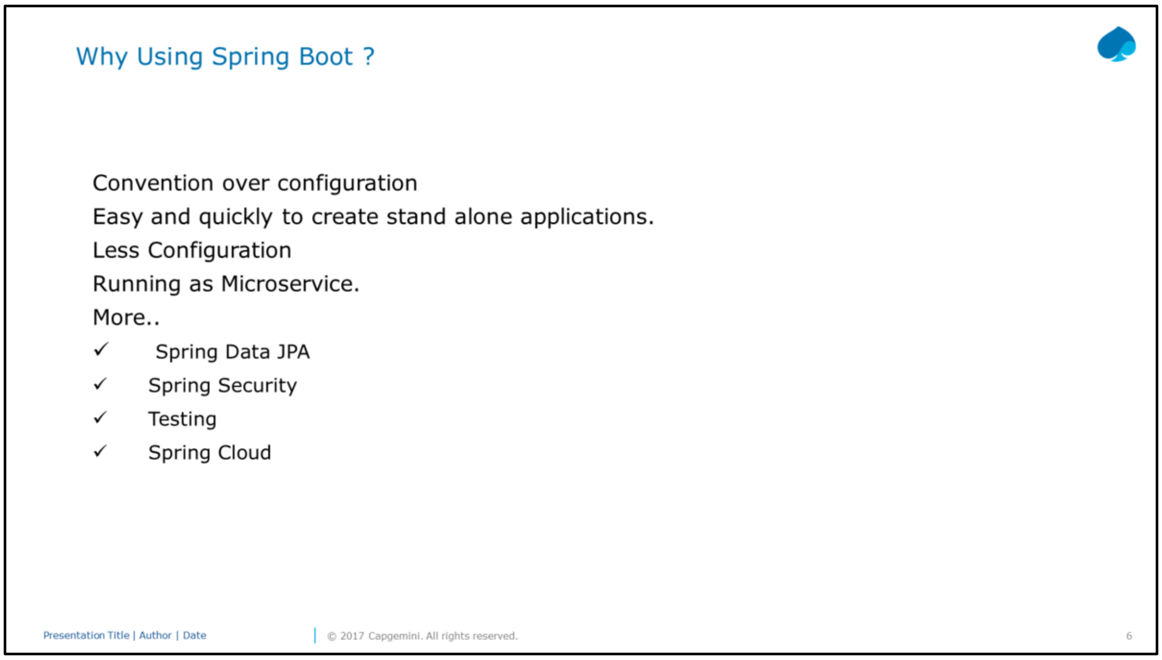
A tool for getting started very quickly with Spring.

Common non-functional requirements for a "real" application.

Exposes a lot of useful features by default.

Gets out of the way quickly if you want to change defaults.

© 2017 Capgemini. All rights reserved.

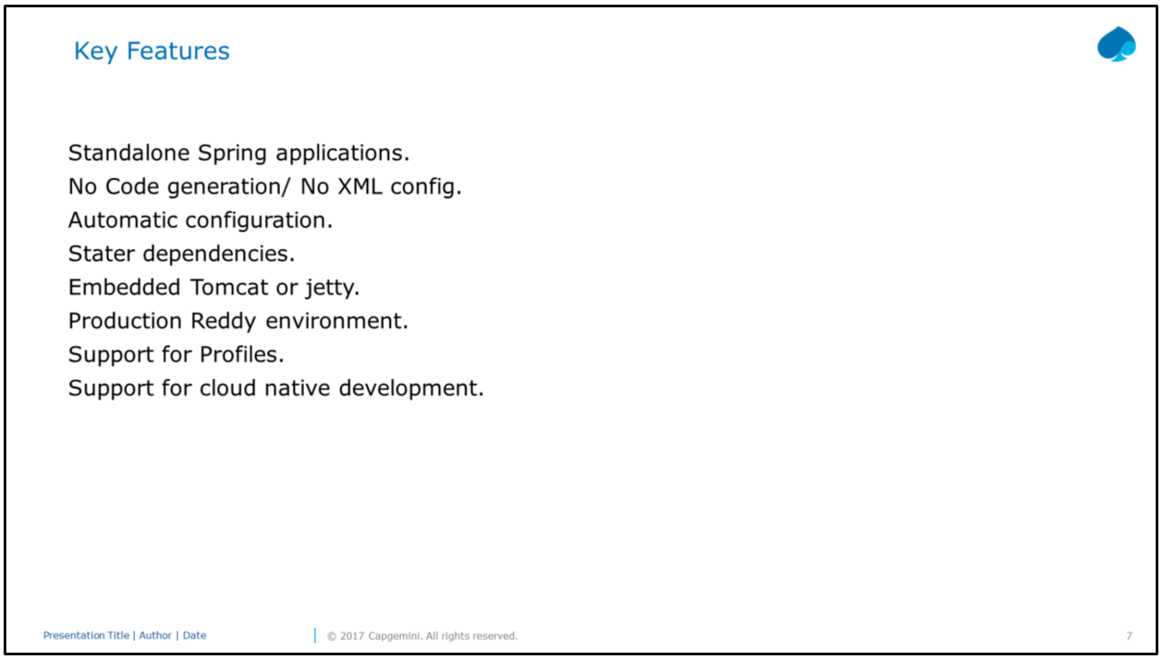


Why Using Spring Boot ?

Convention over configuration
Easy and quickly to create stand alone applications.
Less Configuration
Running as Microservice.
More..

- ✓ Spring Data JPA
- ✓ Spring Security
- ✓ Testing
- ✓ Spring Cloud

© 2017 Capgemini. All rights reserved.



Key Features

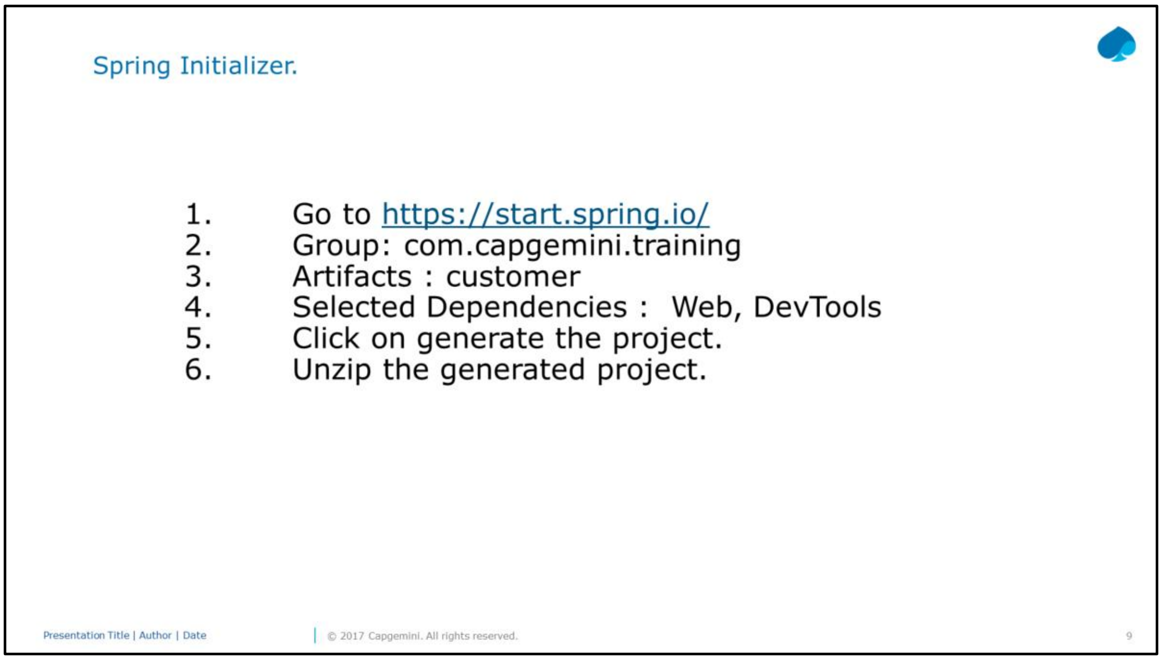
Standalone Spring applications.
No Code generation/ No XML config.
Automatic configuration.
Stater dependencies.
Embedded Tomcat or jetty.
Production Reddy environment.
Support for Profiles.
Support for cloud native development.

© 2017 Capgemini. All rights reserved.

Section -2
Spring Boot Configuration

1. Create a project using Spring Initializer.
2. Setup and run Spring boot App.
3. Create a Rest Controller with

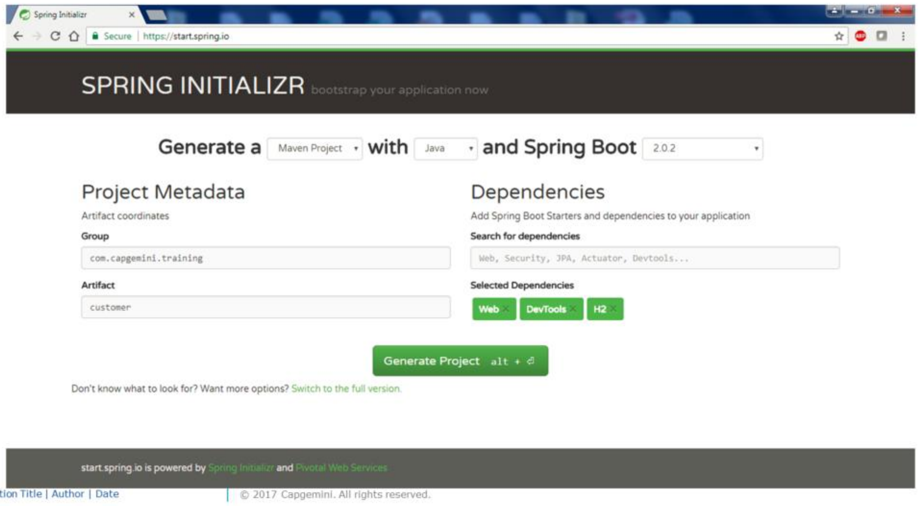
© 2017 Capgemini. All rights reserved.


Spring Initializer.

1. Go to https://start.spring.io/
2. Group: com.capgemini.training
3. Artifacts : customer
4. Selected Dependencies : Web, DevTools
5. Click on generate the project.
6. Unzip the generated project.

© 2017 Capgemini. All rights reserved.

Spring Initializer continue..


Secure | https://start.spring.io

SPRING INITIALIZR bootstrap your application now

Generate a Maven Project with Java and Spring Boot 2.0.2

Project Metadata

Artifact coordinates

Group

com.capgemini.training

Artifact

customer

Dependencies

Add Spring Boot Starters and dependencies to your application

Search for dependencies

Web, Security, JPA, Actuator, Devtools...

Selected Dependencies

Web DevTools H2

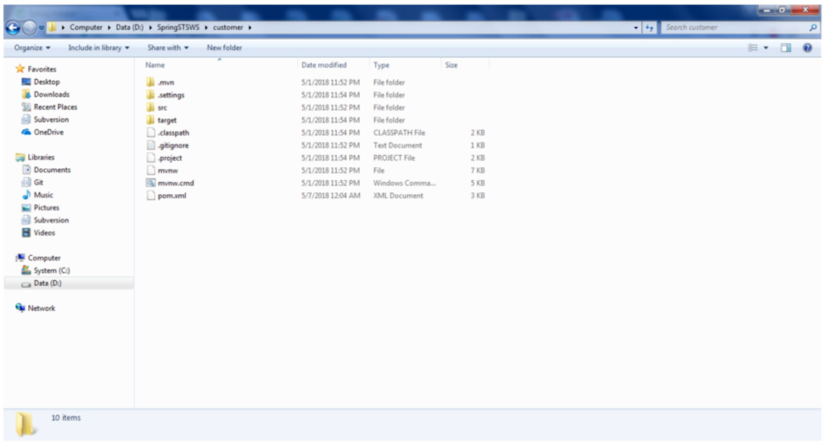
Generate Project alt + ⏎

Don't know what to look for? Want more options? Switch to the full version.

start.spring.io is powered by Spring Initializr and Pivotal Web Services

© 2017 Capgemini. All rights reserved.

Spring Initializer continue..

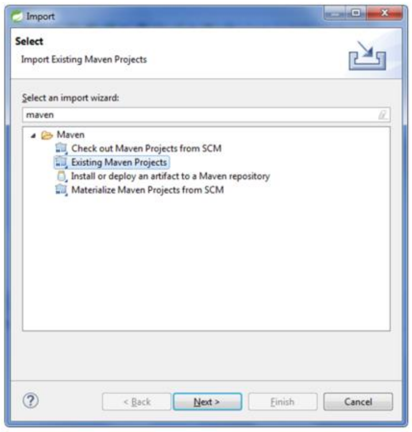


Name	Date modified	Type	Size
.mvn	5/1/2018 11:52 PM	File folder	
.settings	5/1/2018 11:54 PM	File folder	
src	5/1/2018 11:52 PM	File folder	
target	5/1/2018 11:54 PM	File folder	
.classpath	5/1/2018 11:54 PM	CLASSPATH File	2 KB
.gitignore	5/1/2018 11:52 PM	Text Document	1 KB
.project	5/1/2018 11:54 PM	PROJECT File	2 KB
mvnw	5/1/2018 11:52 PM	File	7 KB
mvnw.cmd	5/1/2018 11:52 PM	Windows Comma...	5 KB
pom.xml	5/7/2018 12:04 AM	XML Document	3 KB

10 items

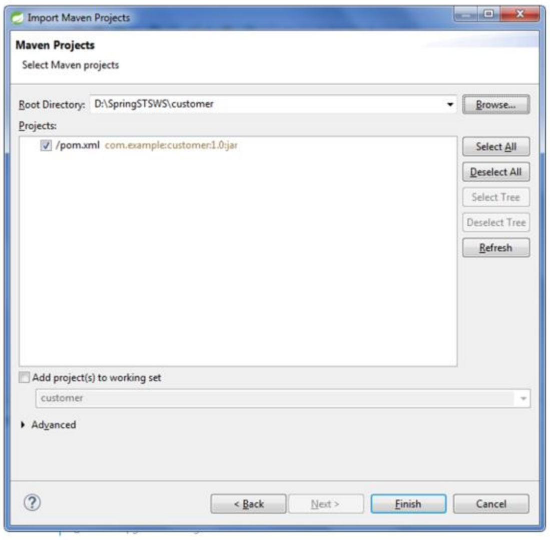
© 2017 Capgemini. All rights reserved.

Spring Initializer Importing the project


Import
Select
Import Existing Maven Projects
Select an import wizard:
maven
Maven
Check out Maven Projects from SCM
Existing Maven Projects
Install or deploy an artifact to a Maven repository
Materialize Maven Projects from SCM
< Back
Next >
Finish
Cancel

© 2017 Capgemini. All rights reserved.

Spring Initializer Importing the project continue..



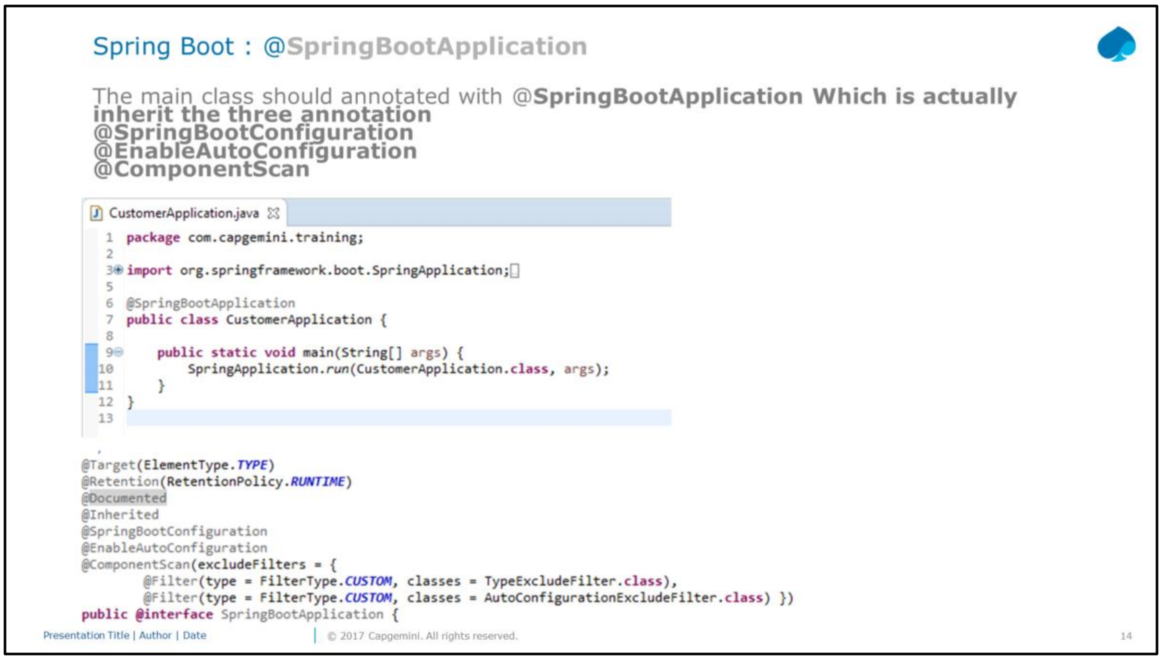
Root Directory: D:\SpringSTSWS\customer

Projects:

[✓] /pom.xml com.example:customer:1.0:jar

[ ] Add project(s) to working set

customer



Spring Boot : @SpringBootApplication

The main class should annotated with **@SpringBootApplication Which is actually inherit the three annotation**
@SpringBootConfiguration
@EnableAutoConfiguration
@ComponentScan

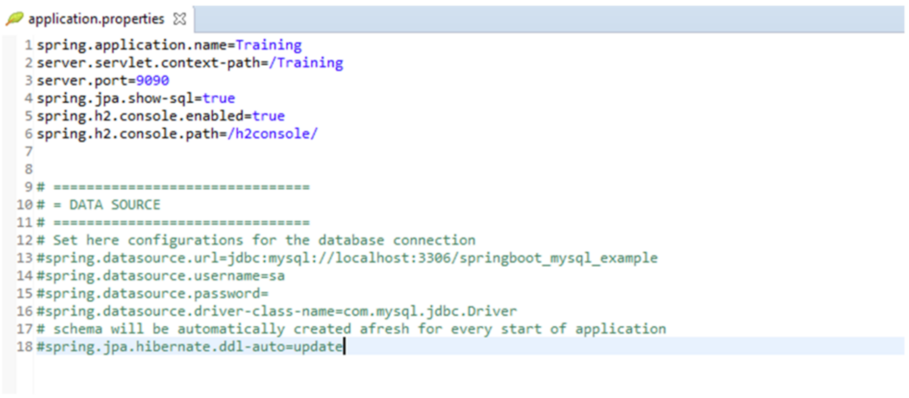
CustomerApplication.java

```
package com.capgemini.training;

import org.springframework.boot.SpringApplication;

@SpringBootApplication
public class CustomerApplication {

    public static void main(String[] args) {
        SpringApplication.run(CustomerApplication.class, args);
    }
}

```

```
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
@Documented
@Inherited
@SpringBootConfiguration
@EnableAutoConfiguration
@ComponentScan(excludeFilters = {
        @Filter(type = FilterType.CUSTOM, classes = TypeExcludeFilter.class),
        @Filter(type = FilterType.CUSTOM, classes = AutoConfigurationExcludeFilter.class) })
public @interface SpringBootApplication {
```

© 2017 Capgemini. All rights reserved.

Spring boot Application.properties

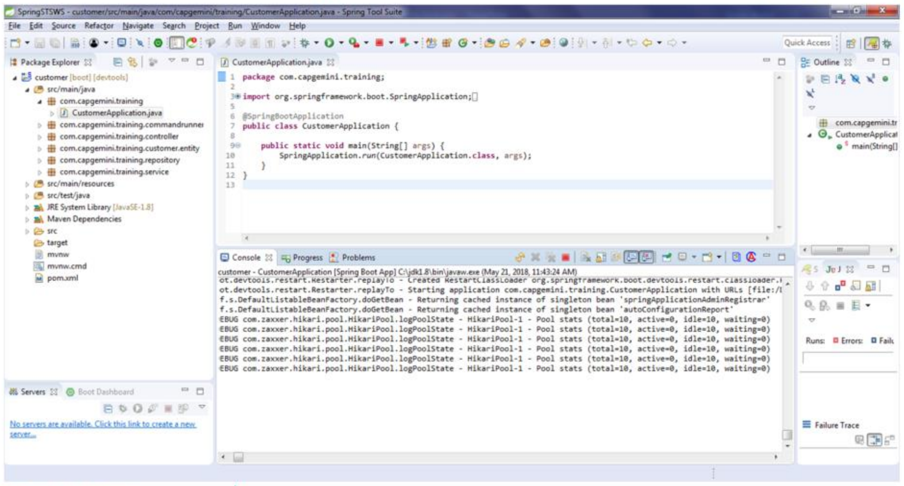

application.properties

```
spring.application.name=Training
server.servlet.context-path=/Training
server.port=9090
spring.jpa.show-sql=true
spring.h2.console.enabled=true
spring.h2.console.path=/h2console/


# ===============================
# = DATA SOURCE
# ===============================
# Set here configurations for the database connection
#spring.datasource.url=jdbc:mysql://localhost:3306/springboot_mysql_example
#spring.datasource.username=sa
#spring.datasource.password=
#spring.datasource.driver-class-name=com.mysql.jdbc.Driver
# schema will be automatically created afresh for every start of application
#spring.jpa.hibernate.ddl-auto=update
```

© 2017 Capgemini. All rights reserved.

Running Spring Boot project

Right click on CustomerApplication class and select Run As SpringBoot App.



CustomerApplication.java

```java
package com.capgemini.training;

import org.springframework.boot.SpringApplication;

@SpringBootApplication
public class CustomerApplication {

    public static void main(String[] args) {
        SpringApplication.run(CustomerApplication.class, args);
    }
}
```

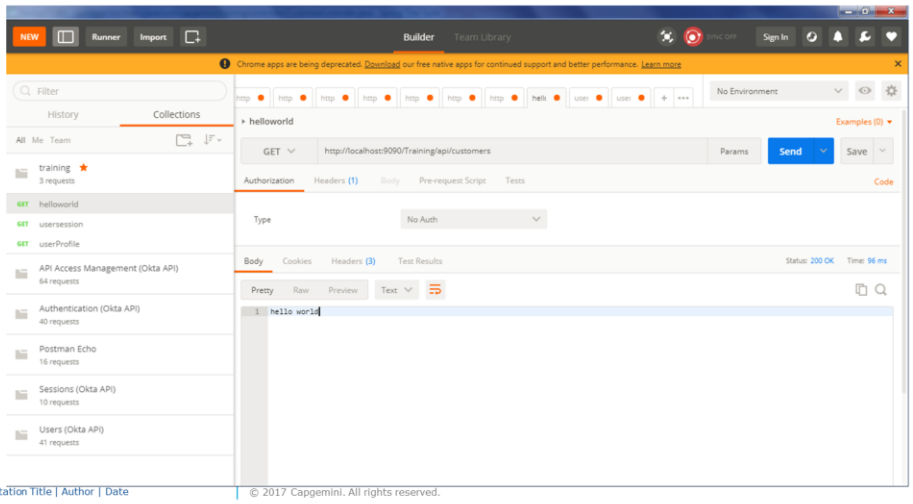
customer - CustomerApplication [Spring Boot App] C:\jdk1.8\bin\javaw.exe (May 21, 2018, 11:43:24 AM)

```
ot.devtools.restart.Restarter.replayTo - Created RestartClassLoader org.springframework.boot.devtools.restart.classloader.I
ot.devtools.restart.Restarter.replayTo - Starting application com.capgemini.training.CustomerApplication with URLs [file:/I
f.s.DefaultListableBeanFactory.doGetBean - Returning cached instance of singleton bean 'springApplicationAdminRegistrar'
f.s.DefaultListableBeanFactory.doGetBean - Returning cached instance of singleton bean 'autoConfigurationReport'
EBUG com.zaxxer.hikari.pool.HikariPool.logPoolState - HikariPool-1 - Pool stats (total=10, active=0, idle=10, waiting=0)
EBUG com.zaxxer.hikari.pool.HikariPool.logPoolState - HikariPool-1 - Pool stats (total=10, active=0, idle=10, waiting=0)
EBUG com.zaxxer.hikari.pool.HikariPool.logPoolState - HikariPool-1 - Pool stats (total=10, active=0, idle=10, waiting=0)
EBUG com.zaxxer.hikari.pool.HikariPool.logPoolState - HikariPool-1 - Pool stats (total=10, active=0, idle=10, waiting=0)
EBUG com.zaxxer.hikari.pool.HikariPool.logPoolState - HikariPool-1 - Pool stats (total=10, active=0, idle=10, waiting=0)
EBUG com.zaxxer.hikari.pool.HikariPool.logPoolState - HikariPool-1 - Pool stats (total=10, active=0, idle=10, waiting=0)
```

SpringSTSWS - customer/src/main/java/com/capgemini/training/CustomerApplication.java - Spring Tool Suite
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
customer [boot] [devtools]
src/main/java
com.capgemini.training
CustomerApplication.java
com.capgemini.training.commandrunner
com.capgemini.training.controller
com.capgemini.training.customer.entity
com.capgemini.training.repository
com.capgemini.training.service
src/main/resources
src/test/java
JRE System Library [JavaSE-1.8]
Maven Dependencies
src
target
mvnw
mvnw.cmd
pom.xml
Servers
Boot Dashboard
No servers are available. Click this link to create a new server...
Console
Progress
Problems
Outline
com.capgemini.tr
CustomerApplicat
main(String[])
Runs:
Errors:
Failu
Failure Trace
Quick Access

© 2017 Capgemini. All rights reserved.

Postman : testing hello world (http://localhost:8080/customers)



Chrome apps are being deprecated. Download our free native apps for continued support and better performance. Learn more

No Environment

- training ★ — 3 requests
  - GET helloworld
  - GET usersession
  - GET userProfile
- API Access Management (Okta API) — 64 requests
- Authentication (Okta API) — 40 requests
- Postman Echo — 16 requests
- Sessions (Okta API) — 10 requests
- Users (Okta API) — 41 requests

helloworld

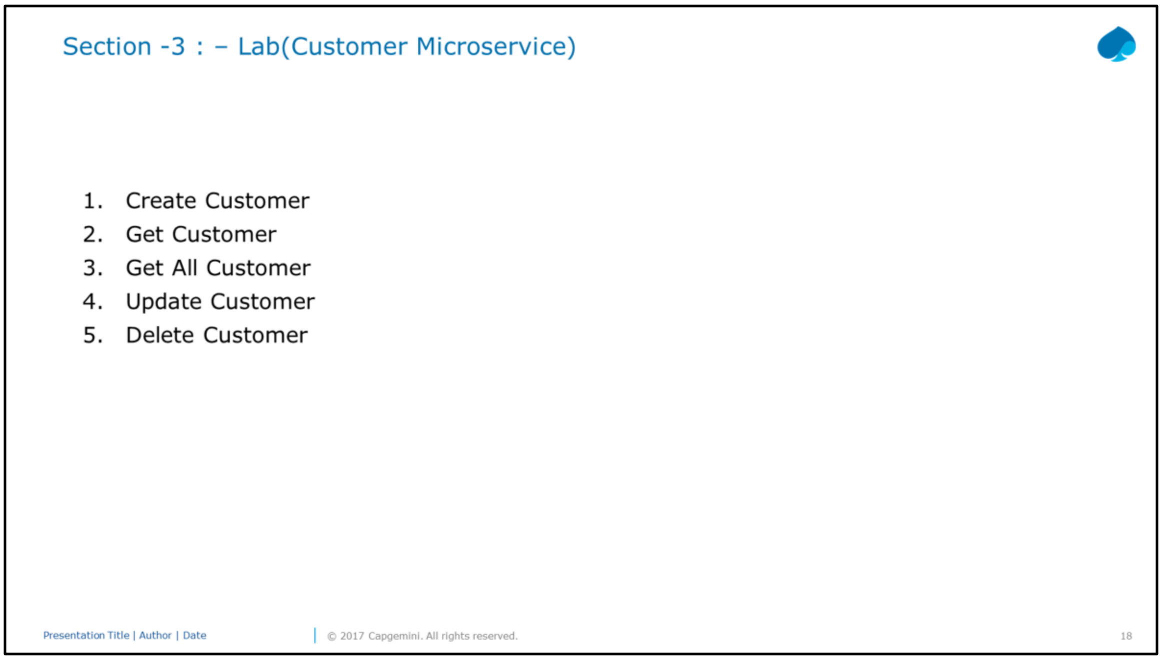
GET http://localhost:9090/Training/api/customers

Type: No Auth

Status: 200 OK Time: 96 ms

```
hello world
```

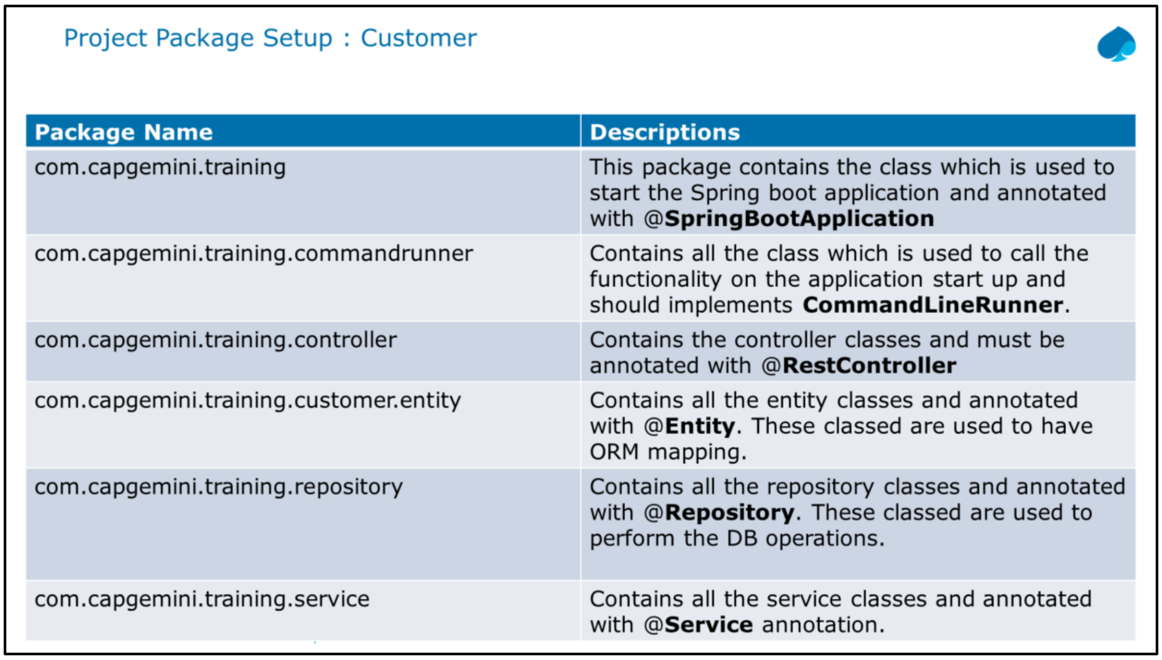
© 2017 Capgemini. All rights reserved.



Section -3 : – Lab(Customer Microservice)

1. Create Customer
2. Get Customer
3. Get All Customer
4. Update Customer
5. Delete Customer

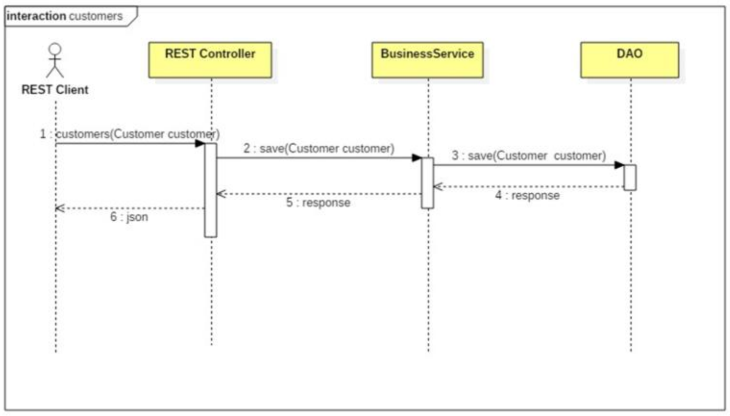
© 2017 Capgemini. All rights reserved.



Project Package Setup : Customer

Package Name	Descriptions
com.capgemini.training	This package contains the class which is used to start the Spring boot application and annotated with @**SpringBootApplication**
com.capgemini.training.commandrunner	Contains all the class which is used to call the functionality on the application start up and should implements **CommandLineRunner**.
com.capgemini.training.controller	Contains the controller classes and must be annotated with @**RestController**
com.capgemini.training.customer.entity	Contains all the entity classes and annotated with @**Entity**. These classed are used to have ORM mapping.
com.capgemini.training.repository	Contains all the repository classes and annotated with @**Repository**. These classed are used to perform the DB operations.
com.capgemini.training.service	Contains all the service classes and annotated with @**Service** annotation.

Customer Microservice: Create Customer
HTTP Verb : POST



© 2017 Capgemini. All rights reserved.

Customer Microservice: Create Customer

Controlle
r

```
@PostMapping("/customers")
public void createCustomer(@RequestBody Customer cust) {
    customerService.save(cust);
}
```

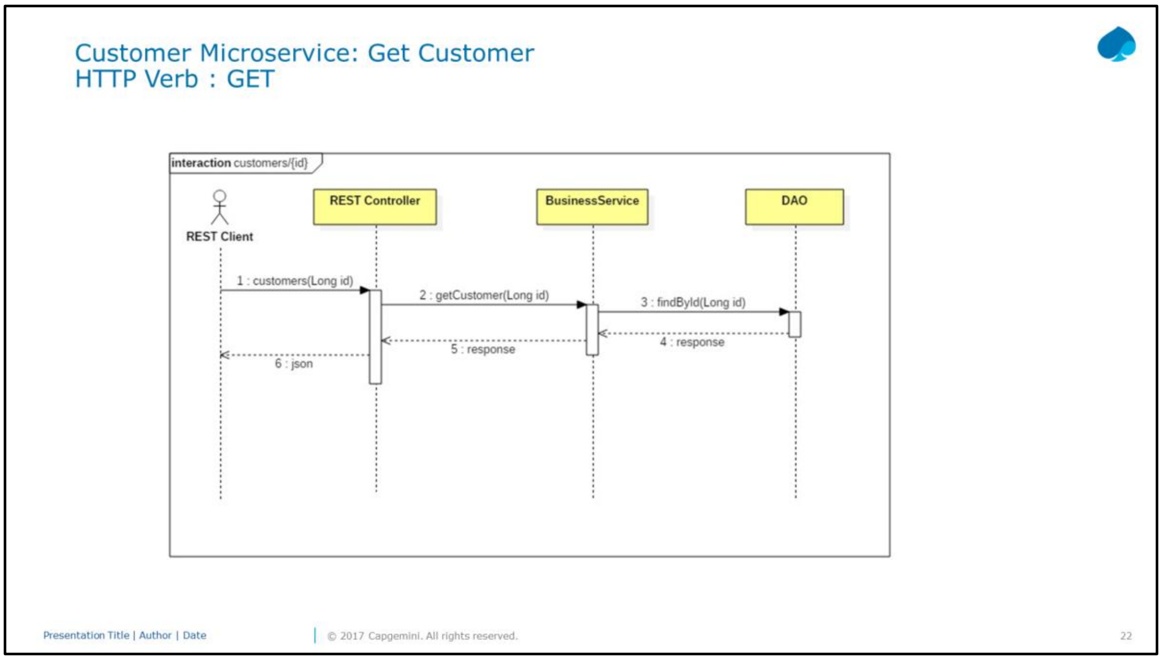
Service

```
@Transactional
public void save(Customer cust) {

    customerRepository.save(cust);
}
```

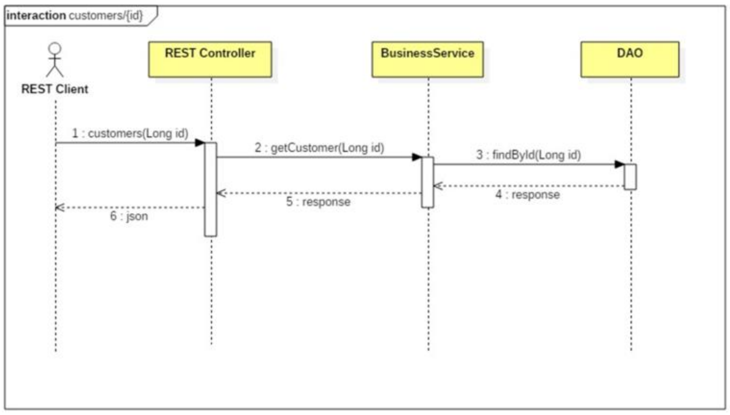
DAO

```
public interface CustomerRepository extends JpaRepository<Customer, Long> {

}
```

© 2017 Capgemini. All rights reserved.



Customer Microservice: Get Customer
HTTP Verb : GET



© 2017 Capgemini. All rights reserved.

Customer Microservice: Get Customer

Controller

```
@GetMapping("/customers/{id}")
public Customer getCustomer(@PathVariable Long id) {
    return customerService.getCustomer(id);
}
```

Service

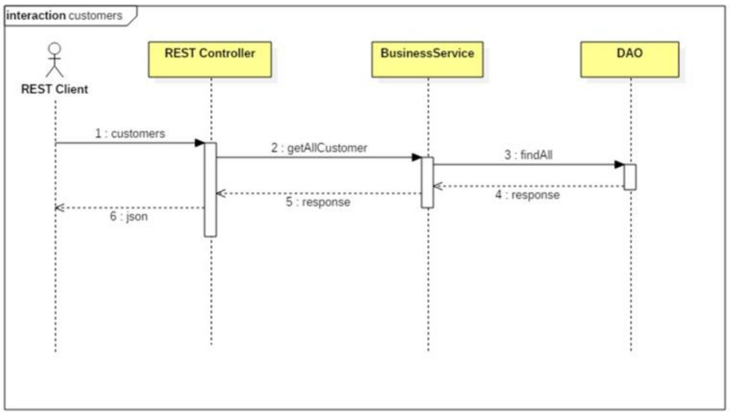
```
public Customer getCustomer(Long id) {
    return customerRepository.findById(id).get();
}
```

DAO

```
public interface CustomerRepository extends JpaRepository<Customer, Long> {

}
```

© 2017 Capgemini. All rights reserved.

Customer Microservice: Get All Customer
HTTP Verb : GET



© 2017 Capgemini. All rights reserved.

Customer Microservice: get All Customers

Controller

```
@GetMapping("/customers")
public List<Customer> getAllCustomer() {
    return customerService.getAllCustomer();
}
```

Service

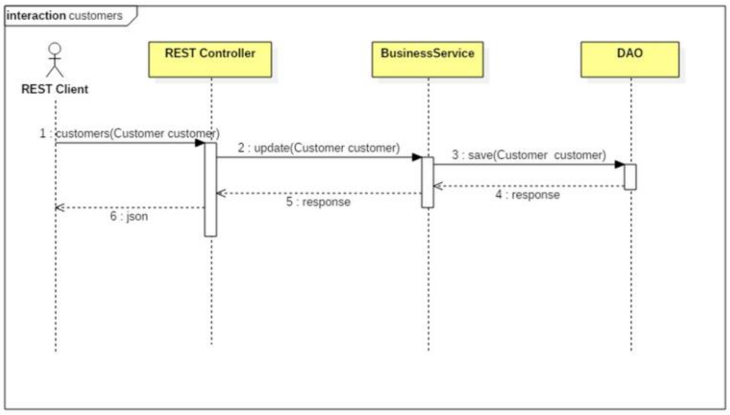
```
public List<Customer> getAllCustomer() {
    List<Customer> customer = new ArrayList<>();
    for (Customer cust : customerRepository.findAll()) {
        customer.add(cust);
    }
    return customer;
}
```

DAO

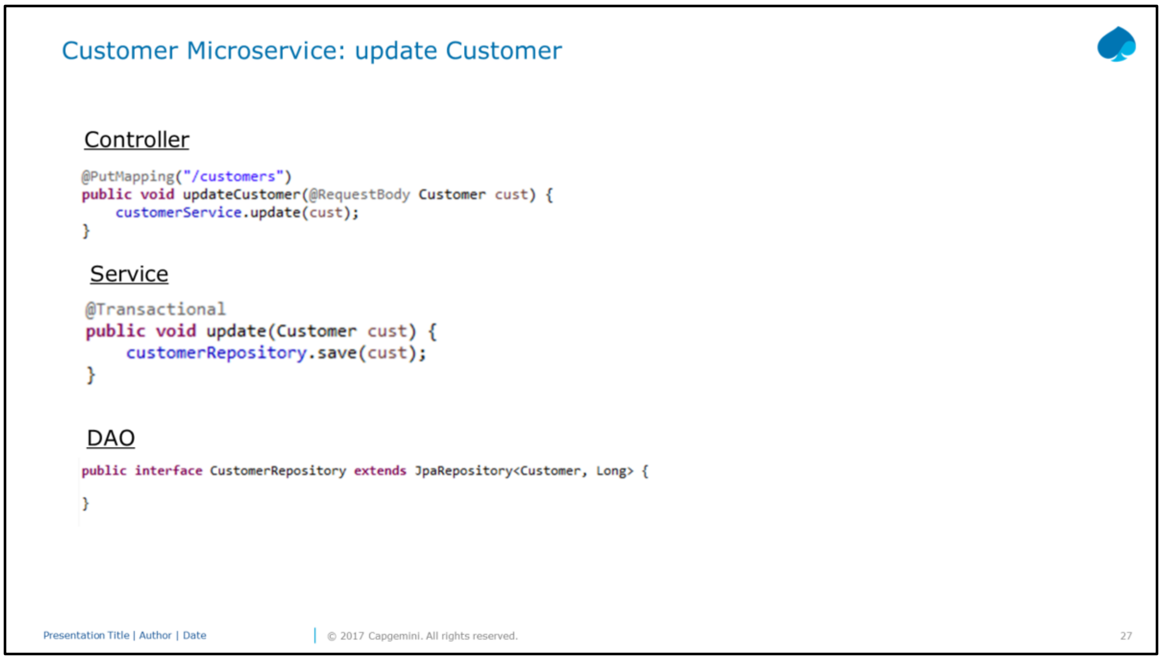
```
public interface CustomerRepository extends JpaRepository<Customer, Long> {

}
```

© 2017 Capgemini. All rights reserved.

Customer Microservice: update Customer
HTTP Verb : PUT



© 2017 Capgemini. All rights reserved.



Customer Microservice: update Customer

Controller

```
@PutMapping("/customers")
public void updateCustomer(@RequestBody Customer cust) {
    customerService.update(cust);
}
```

Service

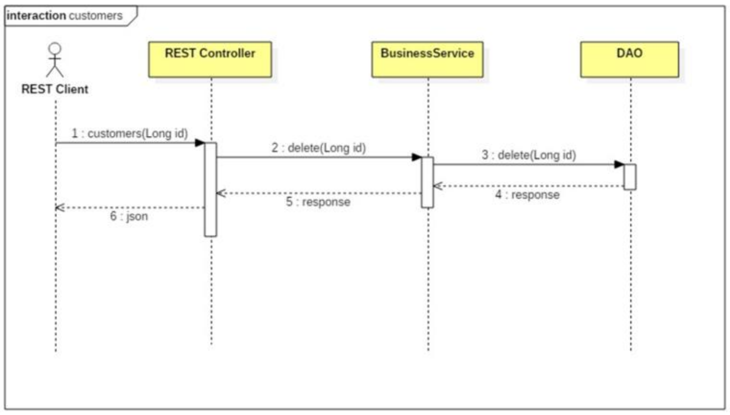
```
@Transactional
public void update(Customer cust) {
    customerRepository.save(cust);
}
```

DAO

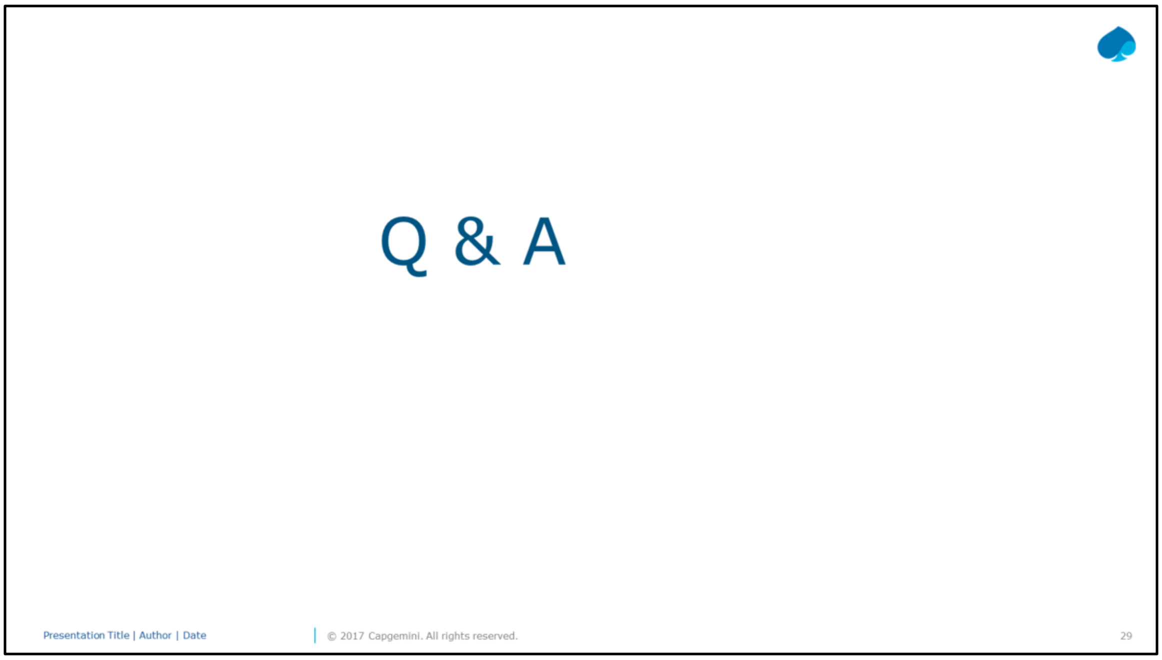
```
public interface CustomerRepository extends JpaRepository<Customer, Long> {

}
```

© 2017 Capgemini. All rights reserved.

Customer Microservice: delete Customer
HTTP Verb : DELETE



© 2017 Capgemini. All rights reserved.



Q & A

© 2017 Capgemini. All rights reserved.

Summary

In this lesson, you have learnt:

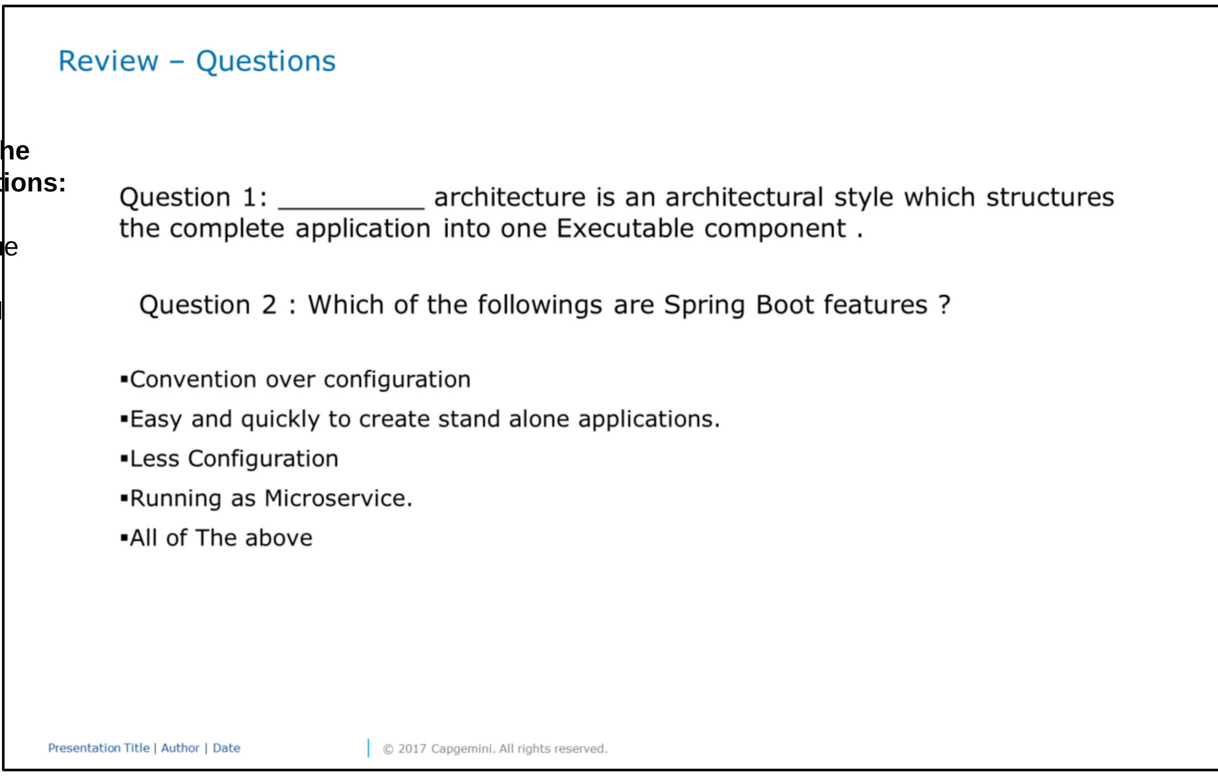
- Introduction to Spring Boot
- Implementing Spring Boot Setup
- How to implement Microservice using Spring Boot

© 2017 Capgemini. All rights reserved.

Answers for the Review Questions:

Answer 1: True

Answer 2: VM



Review – Questions

Question 1: __________ architecture is an architectural style which structures the complete application into one Executable component .

Question 2 : Which of the followings are Spring Boot features ?

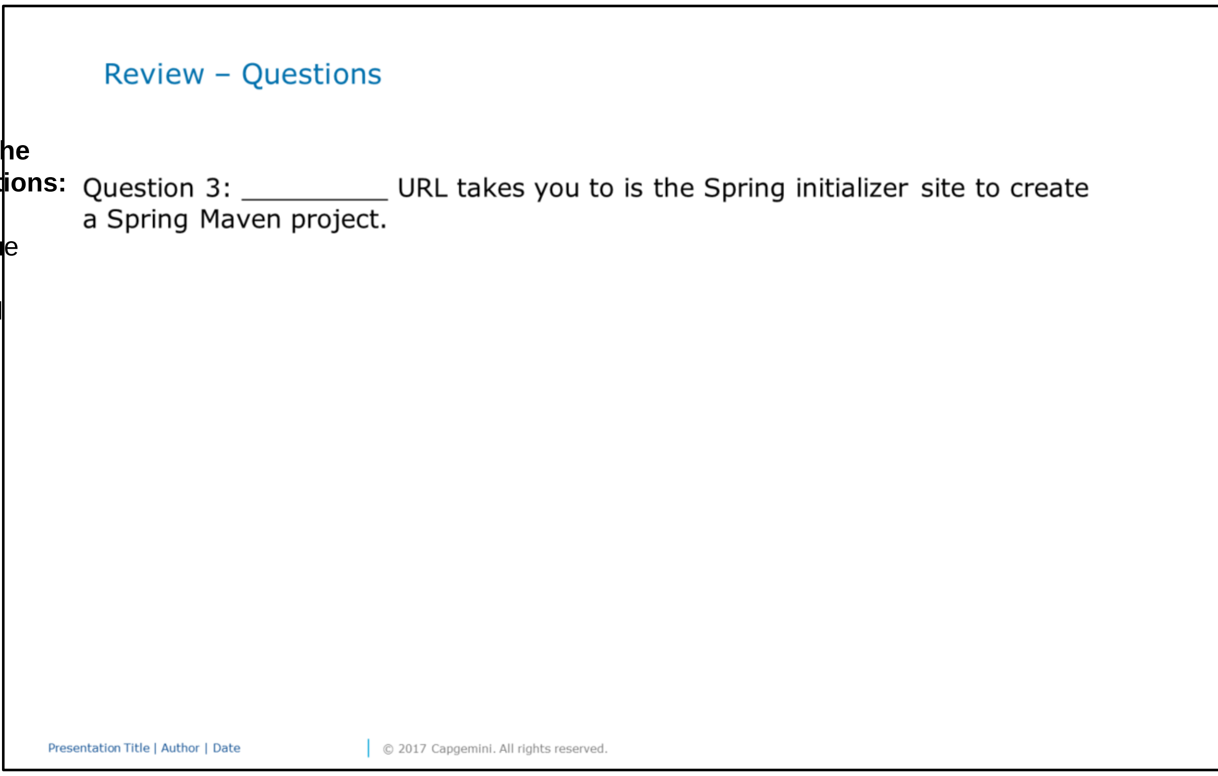
- Convention over configuration
- Easy and quickly to create stand alone applications.
- Less Configuration
- Running as Microservice.
- All of The above

© 2017 Capgemini. All rights reserved.

Answers for the Review Questions:

Answer 1: True

Answer 2: VM

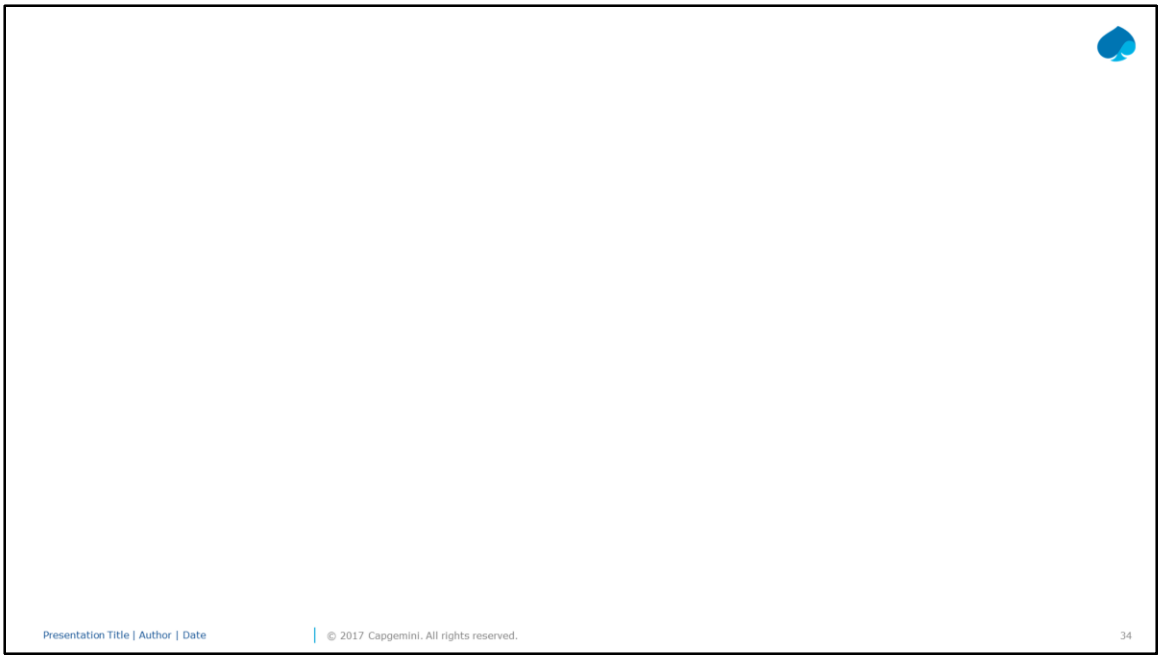


Review – Questions

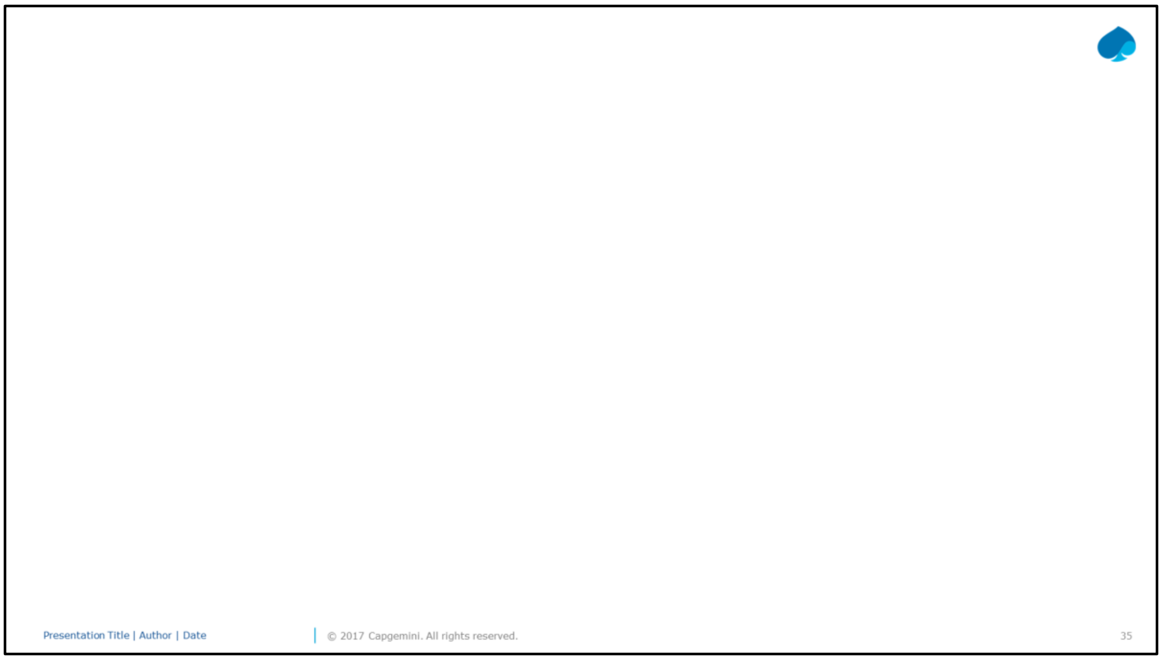
Question 3: __________ URL takes you to is the Spring initializer site to create a Spring Maven project.

© 2017 Capgemini. All rights reserved.

© 2017 Capgemini. All rights reserved.



© 2017 Capgemini. All rights reserved.



© 2017 Capgemini. All rights reserved.

© 2017 Capgemini. All rights reserved.