Capgemini

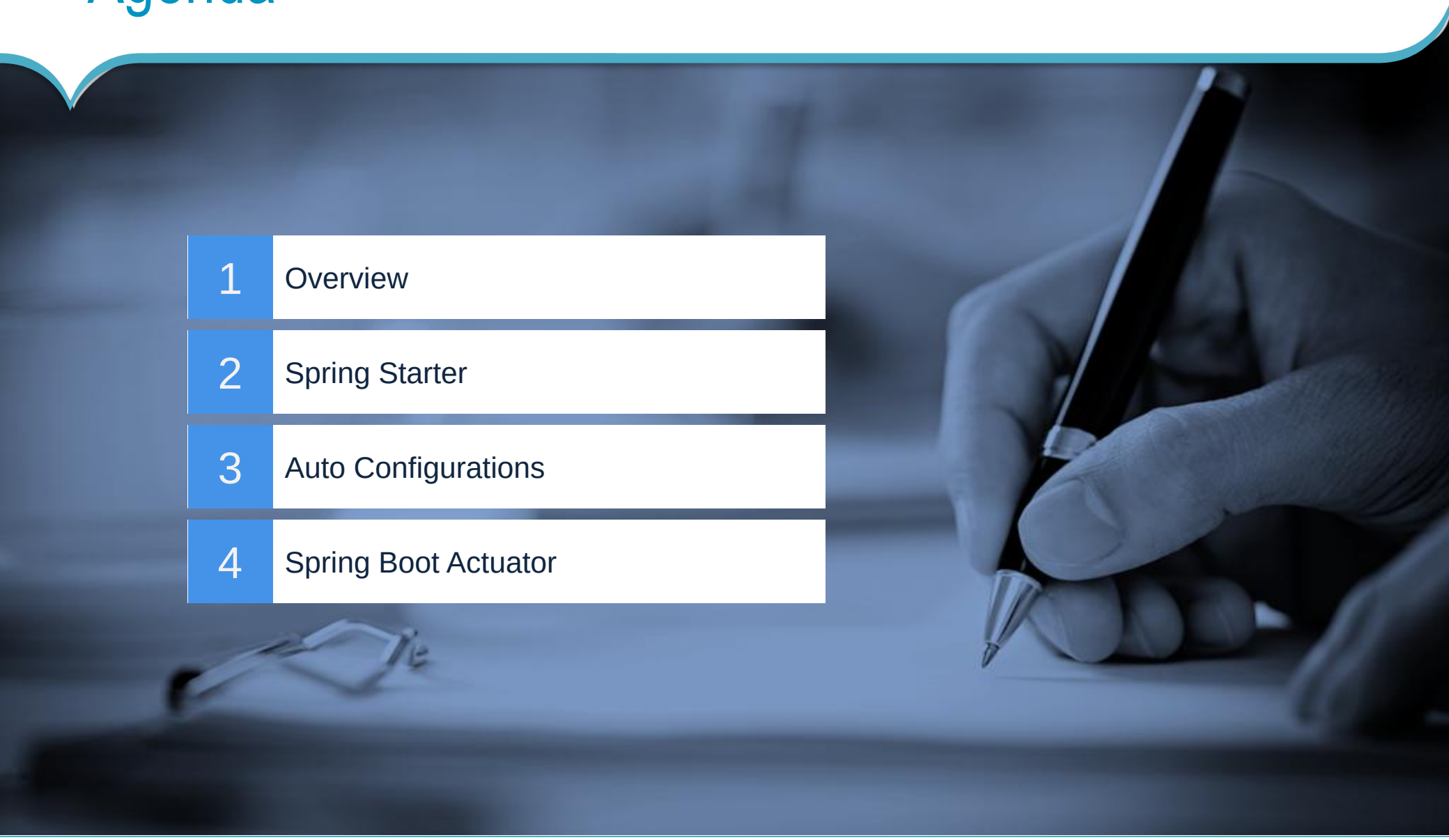
CONSULTING.TECHNOLOGY.OUTSOURCING

Spring Boot

2017

People matter, results count.

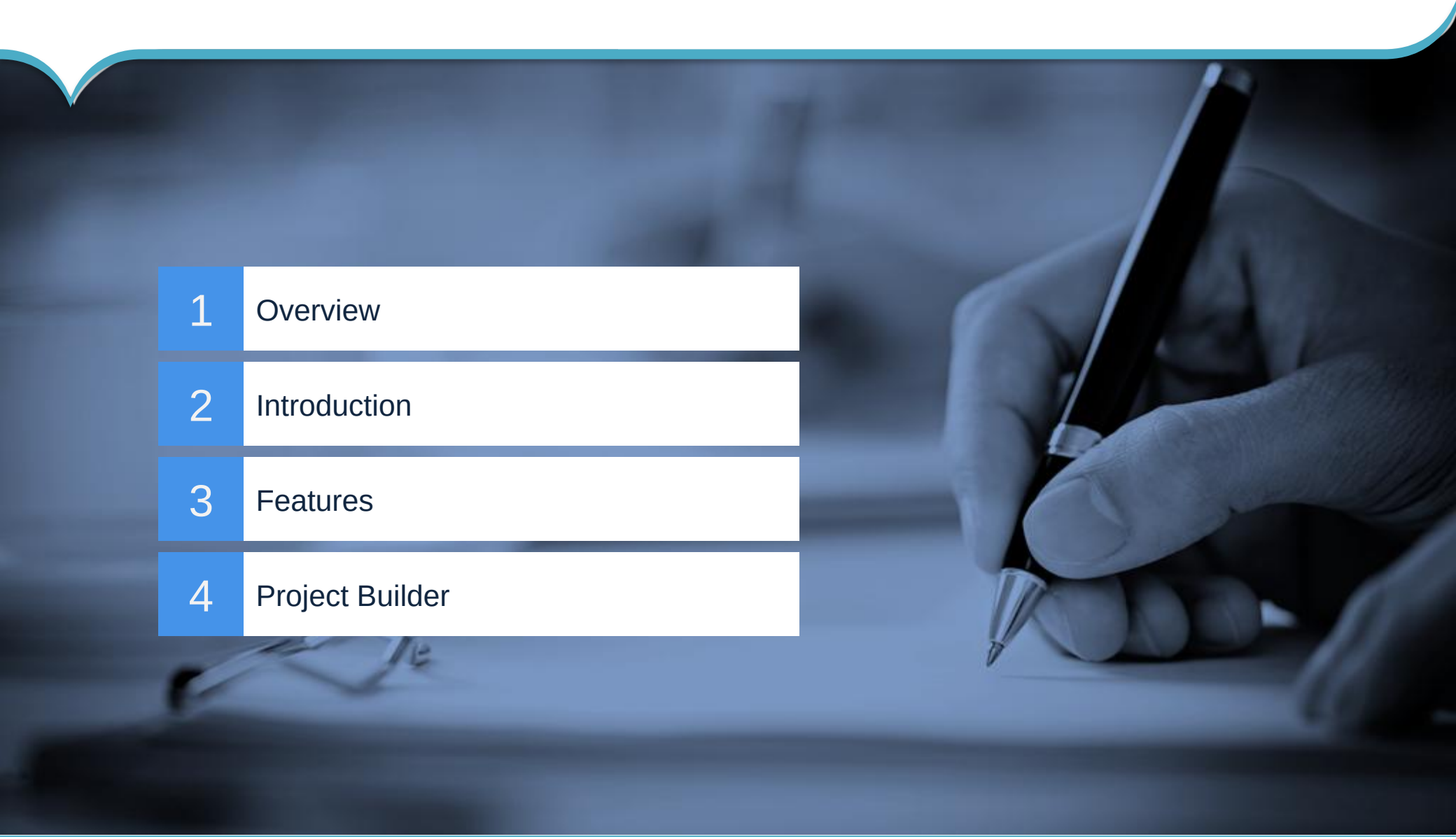
Agenda



1. Overview
2. Spring Starter
3. Auto Configurations
4. Spring Boot Actuator

Spring Boot - Overview

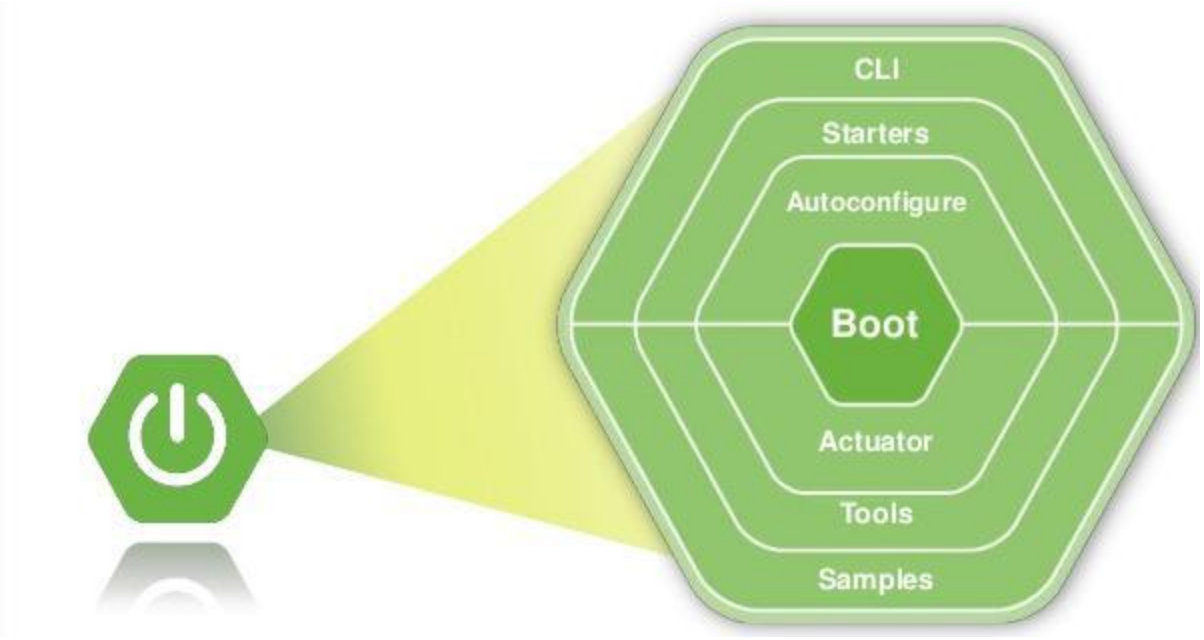
Outline



1. Overview
2. Introduction
3. Features
4. Project Builder

Overview

- Spring is a fantastic open source framework, addresses the complexity of application development. One of the chief advantages of the Spring framework is its layered architecture, which allows you to be selective about which of its components you use. Spring is a cohesive framework for J2EE application development.
- Spring boot is developed on top of the Spring Framework; it's a modular framework which helps in building application infrastructure, taking care of CLI(command line interface), boot strapping, dependencies, framework integrations, testing, tools, auto-configuration and actuator. It favours convenience over configuration and so it's a great way to get quickly ride over spring platform.



Introduction

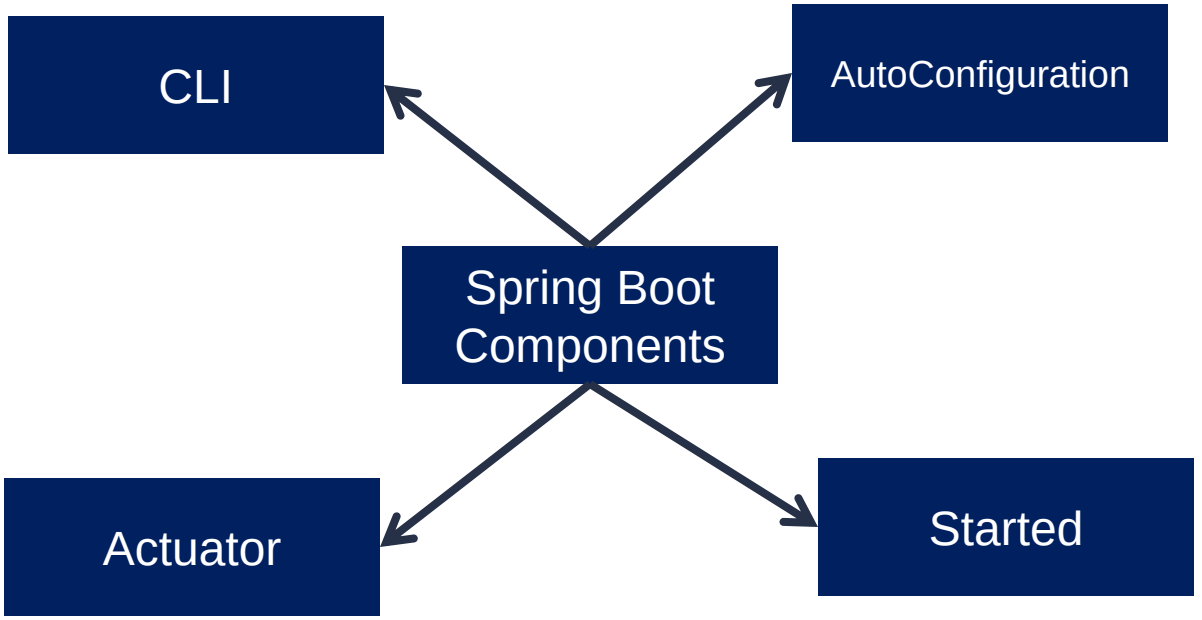
- Spring boot encapsulates spring and many other frameworks and tools like apache velocity, log4j, tomcat and provide starters to take care of these frameworks and tools integration, the figure 1.0 shows boot Starter.

- The spring-boot-starter-parent root module hooks your application with spring boot; and rest of the starter modules can be included as needed.

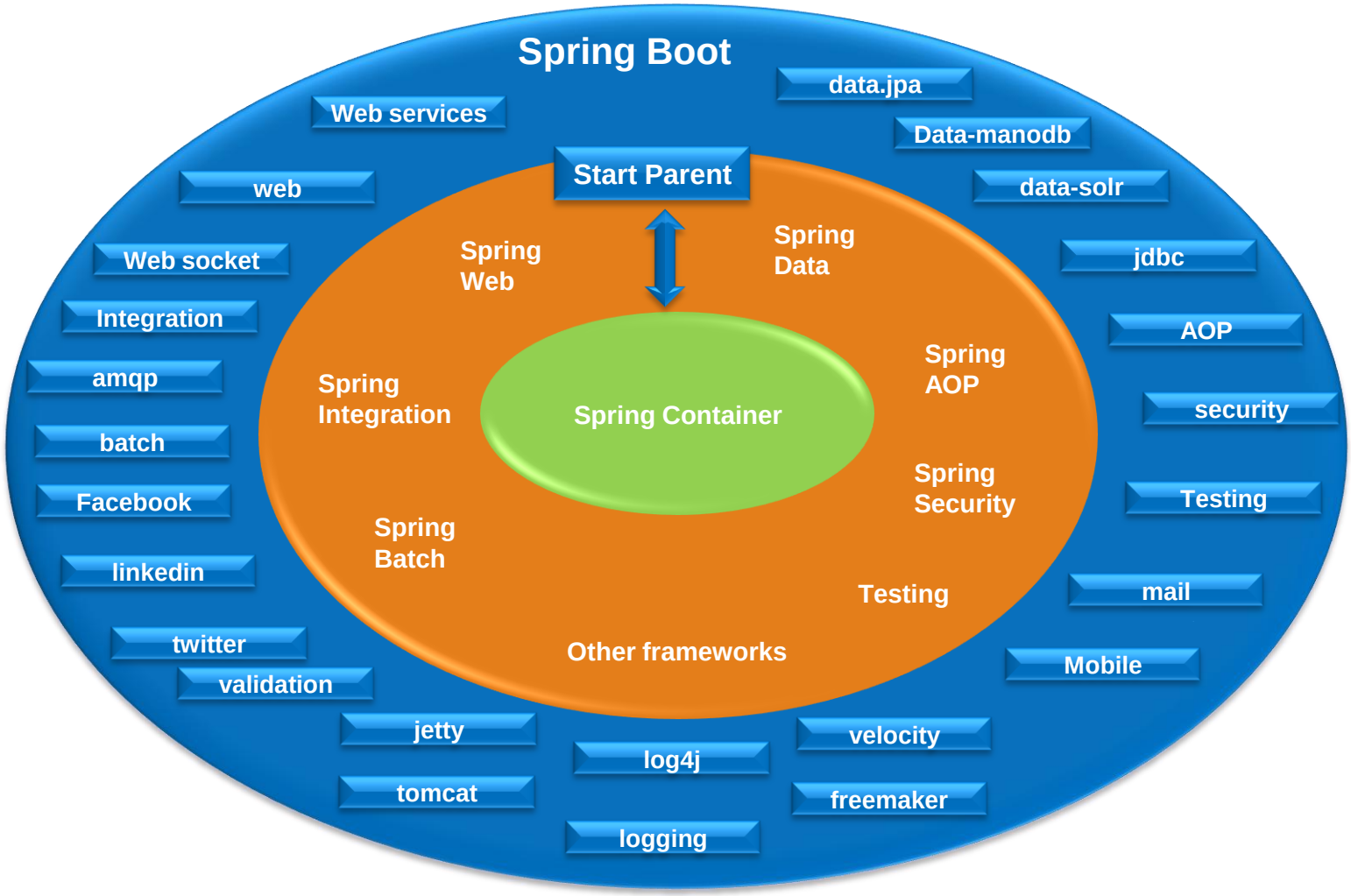
- Spring Boot support xml configuration to get up with legacy applications and favours Java-based configuration.

Overview

- Key Components of Spring Boot Framework:
  - Spring Boot Starters
  - Spring Boot AutoConfigurator
  - Spring Boot CLI
  - Spring Boot Actuator



Introduction



Features

- Faster
  - Generate project scaffolding using Initializr and your IDE
  - Spring Boot startup from 10 seconds to 2 during development
- Smarter
  - Learn how @EnableAutoConfiguration works
    - Tune or disable to your needs
    - Write your own
- Easier
  - Streamline the configuration of our application
    - Relaxed bindings
    - YAML
    - Eliminate casting using typesafe @ConfigurationProperties
    - Integrate with your IDE
- Cloudier
  - Deploy your Spring Boot apps to the cloud using Docker
  - Monitor your app in the cloud or on your servers using built in Actuators
  - Write your own custom monitoring Actuators
  - Remote debug and development with auto restart

Project Builder

- Spring projects can be build in various ways:
  1. Manual Build
  2. Spring Initializr
  3. Eclipse STS (Spring Tool Suite)
  4. Spring Boot CLI
  5. Maven Build
  6. Gradle Build

Spring Boot CLI

- Use the below link to download CLI
  - https://docs.spring.io/spring-boot/docs/current/reference/html/getting-started-installing-spring-boot.html#getting-started-installing-the-cli

- Execute through CLI
  - spring run hello.groovy
    - Takes default port called 8080
  - spring run hello.groovy -- --server.port=9000
    - If you change port

Sample Code:

```
@RestController
class hello{
    @RequestMapping("/")
    String home() {
        "Hello World!"
    }
}
```

Components

- **Spring Boot CLI**
  - Spring Boot CLI(Command Line Interface) is a Spring Boot software to run and test Spring Boot applications from command prompt.

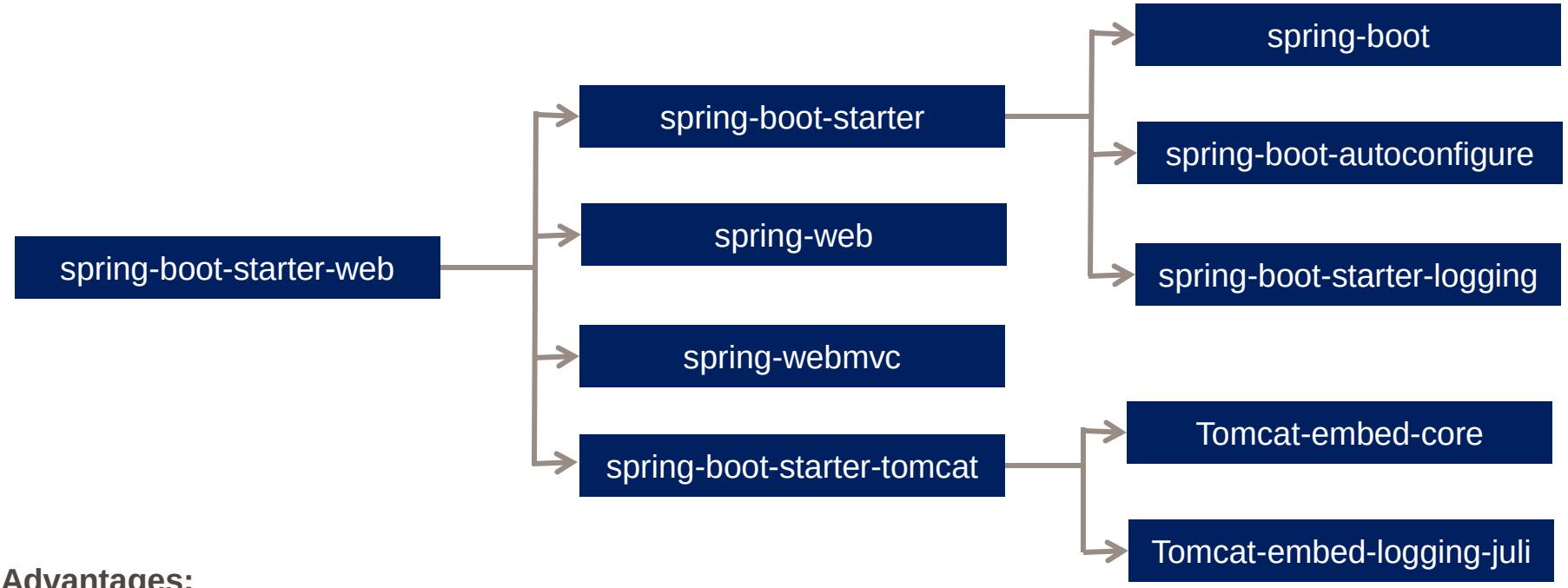
- **spring command example:**
  - spring run HelloWorld.groovy

Project Builder

- **Spring Boot Initilizr With ThirdParty Tools**
  - CURL Tool
  - HTTPie Tool

Components

- **Spring Boot Starter**



- **Advantages:**
  - Spring Boot Starter reduces defining many dependencies simplify project build dependencies.
  - Spring Boot Starter simplifies project build dependencies.

Components

- **Spring Boot AutoConfigurator**
  - AutoConfigurator is to reduce the Spring Configuration
  - No need to define single XML configuration and almost no or minimal Annotation configuration.

@SpringBootApplication = @Configuration + @ComponentScan + @EnableAutoConfiguration

```
@Target(value=TYPE)
@Retention(value=RUNTIME)
@Documented
@Inherited
@Configuration
@EnableAutoConfiguration
@ComponentScan
public @interface SpringBootApplication
```

Components

```
package com.capgemini;
@EnableAutoConfiguration
public class Foo {
...
}
```

Package has significance

Intelligent and seemingly "magical" annotation that enables features and configures functionality

```
package com.capgemini;
@Configuration
@ComponentScan
@EnableAutoConfiguration
public class Foo {
...
}
```

3 very common annotations in Spring Boot apps

Components

```
package com.capgemini;
@SpringBootApplication
public class Foo {
...
}
```

Package has significance

Introduced in Spring Boot 1.2.
Really three annotations in one.

Intelligent Decision Making Based on Conditions

- Presence / Absence of Jars
- Presence / Absence of Beans
- Presence / Absence of Property

Components

- **Externalized Configurations:**
  - Enhanced configuration
    - YAML
    - Typesafe configuration
    - Resolving configuration

YAML

✓Pronounced YAM-EL, rhymes with Camel

✓Data serialization language / format

✓Since 2001

✓Ruby, Python, ElasticSearch, MongoDb

YAML vs Properties

YAML

- Defined spec: http://yaml.org/spec/
- Human readable
- key/value (Map), Lists, and Scalar types
- Used in many languages
- Hierarchical
- Doesn't work with @PropertySource
- Multiple Spring Profiles in default
- Config

.properties

- java.util.Properties Javadoc is spec
- Human readable
- key/value (Map) and String types
- Used primarily in Java
- Non-hierarchical
- Works with @PropertySource
- One Spring Profile per config

Components

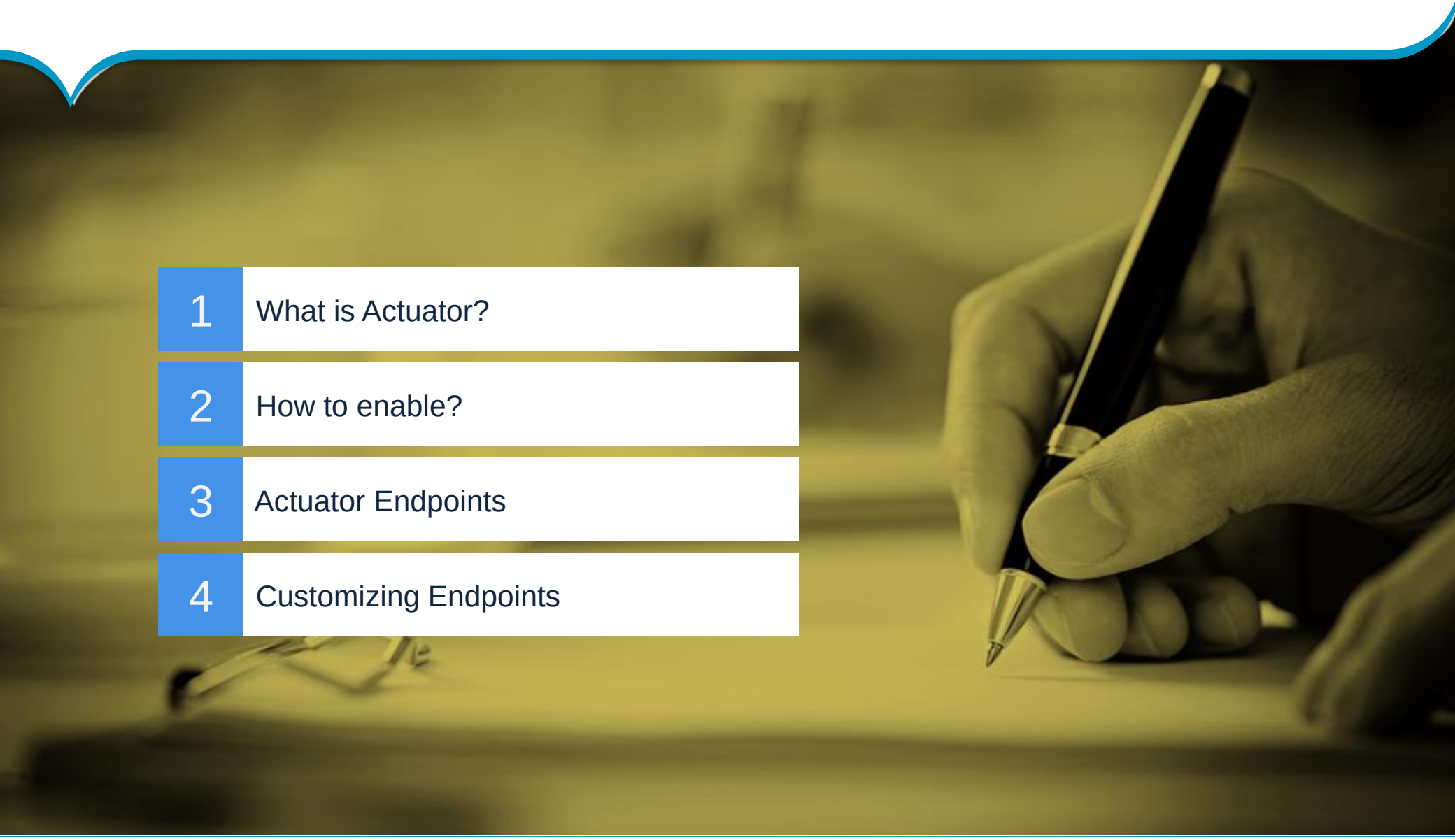
- **Spring Boot Actuator**
  - Spring Boot Actuator components gives many features, but two major features are
    - Providing Management EndPoints to Spring Boot Applications.
    - Spring Boot Applications Metrics.

Summary

- **Overview**
- **Features**
- **Key Components**
- **CLI**

Spring Boot Actuator

Outline



1. What is Actuator?
2. How to enable?
3. Actuator Endpoints
4. Customizing Endpoints

What is Actuator?

- Spring Boot Actuator is a Sub-Project of Spring Boot.
- Spring Boot Actuator includes a number of additional features to help you monitor and manage your application when it's pushed to production.
- You can choose to manage and monitor your application using HTTP Endpoints, with JMX or even by remote shell (SSH or TELNET). Auditing, Health and Metrics gathering can be automatically applied to your application.
- Actuator is supported out of the box within spring boot applications. You just have to add the dependency to enable the actuator. The default configurations are enabled if you are not providing any application specific configurations.
- Main purpose of this feature is to provide various useful metrics about the applications. It is very helpful in the production environment to check the various metrics like **health of your application, configurations, error page, version details, etc.**

How to enable?

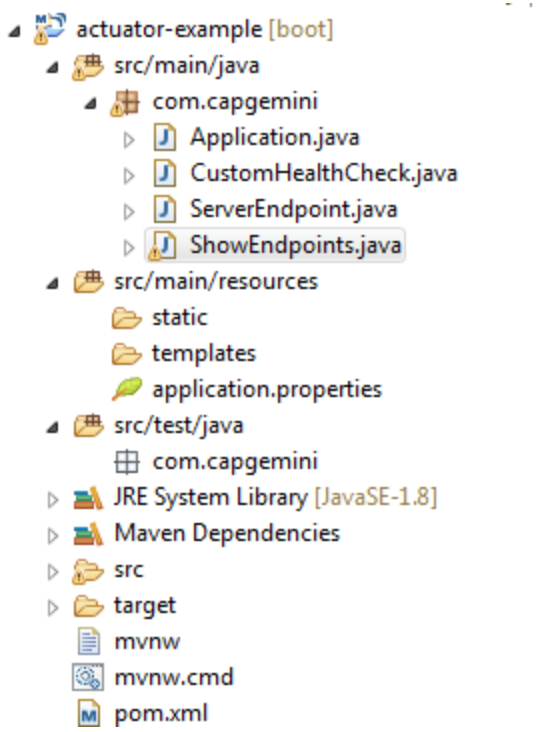
```
<dependency>
 <groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```

Actuator Endpoints

- HTTP Endpoints are enabled for accessing the various information about your application. List of 15 endpoints that are currently supported by spring boot actuator module is:
- ***actuator*:** It is the default endpoint to list the available endpoints exposed to the user. It is enabled only when HATEOAS available in your classpath.
- ***autoconfig* :** Report on auto-configuration details.
- **beans :** This endpoint lists all the beans loaded by the application.
- ***configprops* :** This endpoint shows configuration properties used by your application.
- ***dump* :** Performs a thread dump.
- ***env* :** Exposes spring's properties from the configurations.
- ***health* :** Health of the application.
- ***info* :** Displays application information like version, description, etc.
- ***metrics* :** Metrics about memory, heap, etc. for the currently running application
- ***mappings* :** Displays a list of all @RequestMapping paths.
- ***shutdown* :** This endpoint allows to shutdown the application. This is not enabled by default.
- ***trace* :** Displays trace information for the current application.
- ***logfile* :** Provides access to the configured log files (This feature supported since Spring Boot 1.3.0).
- ***flyway* :** This endpoint provides the details of any flyway database migrations have been applied (This feature supported since Spring Boot 1.3.0).
- ***liquibase* :** This endpoint provides the details of any liquibase database migrations have been applied (This feature supported since Spring Boot 1.3.0).

Customizing Endpoints

- Just enabling the endpoints may not be sufficient in most of the real time applications. You would like to update application specific configurations.
  - Spring Boot configuration File **application.properties**



Customizing Endpoints

- The most common settings are:
  - management.port=8081 – change the port of the actuator endpoints in your server
  - management.address=127.0.0.1 – restrict to run only on localhost
  - management.context-path=/details – change the context path for actuator
  - endpoints.health.enabled=false – enable or disable the endpoints.

Summary

- **Endpoints**
- **Server**
- **HealthCheckUp**
- **Info**
- **Beans**

Capgemini

CONSULTING.TECHNOLOGY.OUTSOURCING

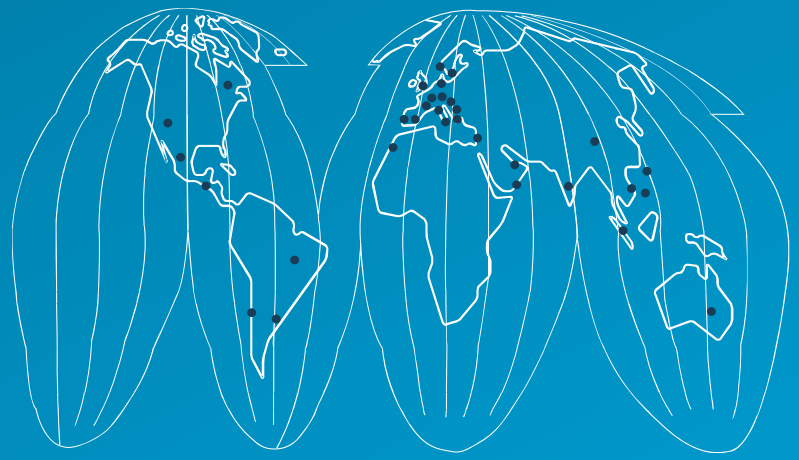
People matter, results count.

About Capgemini

With more than 145,000 people in 40 countries, Capgemini is one of the world's foremost providers of consulting, technology and outsourcing services. The Group reported 2014 global revenues of EUR 10.5 billion.

Together with its clients, Capgemini creates and delivers business and technology solutions that fit their needs and drive the results they want. A deeply multicultural organization, Capgemini has developed its own way of working, the Collaborative Business Experience™, and draws on Rightshore®, its worldwide delivery model.

Rightshore® is a trademark belonging to Capgemini



www.capgemini.com