Think IT Think Us
RAYS

SpringBoot

www.raystec.com
www.sunilos.com

Spring Boot

- Spring Boot makes it easy to create stand-alone enterprise application
- Easy to understand and develop spring applications
- Single class will run your entire application with integrated web server.
- Reduces the development time
- Microservices can be developed in spring boot

Maven Dependency

```
<properties>
        <java.version>1.8</java.version>
</properties>

<parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.1.2.RELEASE</version>
        <relativePath /> <!-- lookup parent from repository -->
</parent>

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

Start an Application

```
@SpringBootApplication
@ComponentScan(basePackages = { "com.sunilos.springboot" })
public class Application {
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }

    @Bean
    public UserDTO userBean() {
            UserDTO dto = new UserDTO();
            dto.setFirstName("Ram");
            dto.setLastName("Sharma");
            return dto;
    }
}
```

@Configuration

- **@Configuration**
- public class AppConfig{
- @Bean
- public UserDTO userBean() {
- UserDTO dto = new UserDTO();
- dto.setFirstName("Ram");
- dto.setLastName("Sharma");
- return dto;
- }
- }

Application properties

- #Server info
- server.port = 8080
- spring.application.name = SunilOSApp

- server.servlet.context-path =/SunilOS

- ##Data connection pool
- spring.datasource.driver-class-name=com.mysql.jdbc.Driver
- spring.datasource.url = jdbc:mysql://localhost:3306/ORS_P10?useSSL=false
- spring.datasource.username = ram
- spring.datasource.password = ram

Application properties (Cont.)

- ## Hibernate/JPA Properties
- spring.jpa.show-sql=true
- spring.jpa.properties.hibernate.format_sql = true
- spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQLDialect

- #Email properties
- spring.mail.host=smtp.gmail.com
- spring.mail.port=587
- spring.mail.username=my@gmail.com
- spring.mail.password=mypass
- spring.mail.properties.mail.smtp.auth=true
- spring.mail.properties.mail.smtp.starttls.enable=true

Develop RESTful web sevice

Add/Update/Delete/Search College

College

Record is successfully saved..

Name (*)	Symbiosis
Address *	Pune
Phone No * (10 Digits)	9876543210
City *	Pune
State *	MH

Save Search

College List

Name Address Search

ID	Name	Address	Phone	City	State	Edit	Delete
73	Bajaj ji	Mumbai	7894561236	Mumbai	MH	Edit	Delete
82	IPSA	Rau	9876543210	Indore	MP	Edit	Delete
87	Symbiosis	Pune	9876543210	Pune	MH	Edit	Delete

API

- http://localhost:8080/College/get/1
- http://localhost:8080/College/delete/1
- http://localhost:8080/College/save
- http://localhost:8080/College/search

Webservice is REST controller

```
@RestController
@RequestMapping(value = "College")
public class CollegeCtl {
```

REST Methods

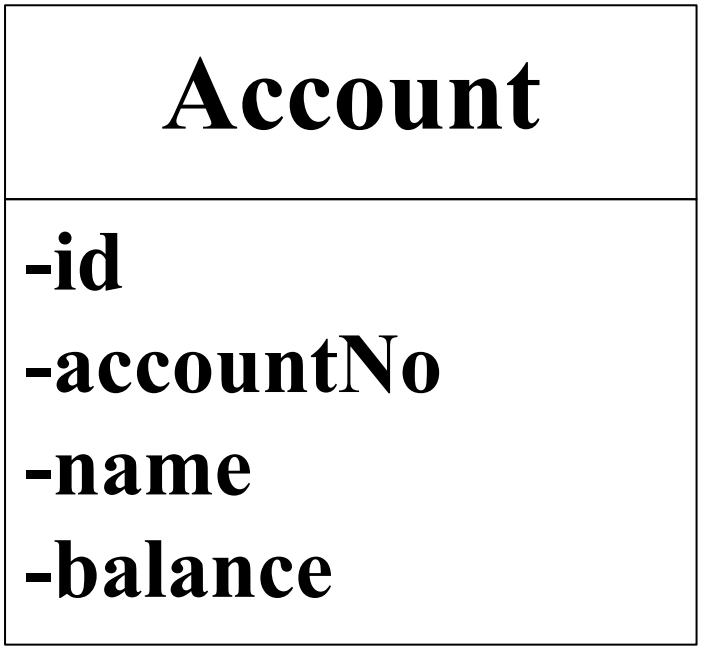
```
@GetMapping("get/{id}")
public Map get(@PathVariable Long id) {}

@GetMapping("delete/{id}")
public Map delete(@PathVariable Long id){}

@PostMapping("save")
public Map save(@RequestBody College c){}

@PostMapping("search")
public Map search(@RequestBody College c){}
```

Add/Update/Delete/Search Account



REST API

- http://localhost:8080/Account/get/1
- http://localhost:8080/Account/delete/1
- http://localhost:8080/Account/save
- http://localhost:8080/Account/search

application.properties

- **##Datasource**
- spring.datasource.driver-class-name=**com.mysql.jdbc.Driver**
- spring.datasource.url = **jdbc:mysql://localhost:3306/TEST**
- spring.datasource.username = **root**
- spring.datasource.password = **password**

- **##Hibernate/JPA Properties**
- spring.jpa.show-sql=true
- spring.jpa.properties.hibernate.format_sql = true
- spring.jpa.hibernate.ddl-auto = update

- spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MariaDB53Dialect

Maven Dependency

- <dependency>
- <groupId>org.springframework.boot</groupId>
- <artifactId>**spring-boot-starter-data-jpa**</artifactId>
- </dependency>

- <dependency>
- <groupId>**mysql**</groupId>
- <artifactId>**mysql-connector-java**</artifactId>
- <version>8.0.12</version>
- </dependency>

Spring DAO

- Data Access Object
- Contains data access logics
- Performs database operation
- Created using @Repository annotation

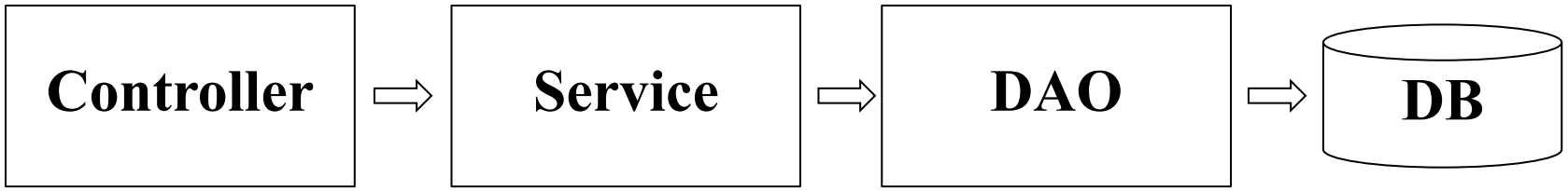
```
@Repository
public class AccountDAO {
  @PersistenceContext
  protected EntityManager entityManager;
}
```

Spring Service

- Performs business operations using DAO classes
- Does transaction handling
- Created using @Service annotation
- A method is made transactional using @Transaction annotation

```
@Service
public class AccountService {
	@Autowired
	private AccountDAO dao = null;
}
```

Spring Controller




```
@RestController
public class AccountCtl {
  @Autowired
  private AccountService service = null;
}
```

Interceptors

```
@Component
@Order(1)
public class FrontCtl extends HandlerInterceptorAdapter {
        public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler)
                        throws Exception {
                System.out.println("Front controller : Pre operation");
                return true;
        }
}
```

Interceptor Configuration

```
@Autowired
private FrontCtl frontCtl;
@Bean
public WebMvcConfigurer InterConfigurer() {
  return new WebMvcConfigurer() {
        @Override
        public void addInterceptors(InterceptorRegistry registry) {
                registry.addInterceptor(frontCtl);
        }
  };
}
```

Deploy App on external web server

```
@SpringBootApplication
public class Application extends SpringBootServletInitializer{
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

CORS

- Cross-Origin Resource Sharing (CORS) is a security policy that uses HTTP concept that allows restricting the resources implemented in web browsers.
- It prevents the JavaScript code producing or consuming the requests against different origin.
- API can be accessed by different front end frameworks like Angular, Reactive, Ext JS, Android, IOS, etc from different origins
- You can restrict origin for the API.
-

Enable CORS

```
@Bean
public WebMvcConfigurer corsConfigurer() {
  return new WebMvcConfigurerAdapter(){
    @Override
    public void addCorsMappings(CorsRegistry registry) {
      CorsRegistration cors = registry.addMapping("/**");
      cors.allowedOrigins("http://localhost:4200");
      cors.allowedHeaders("*");
      cors.allowCredentials(true);
    }
  };
}
```

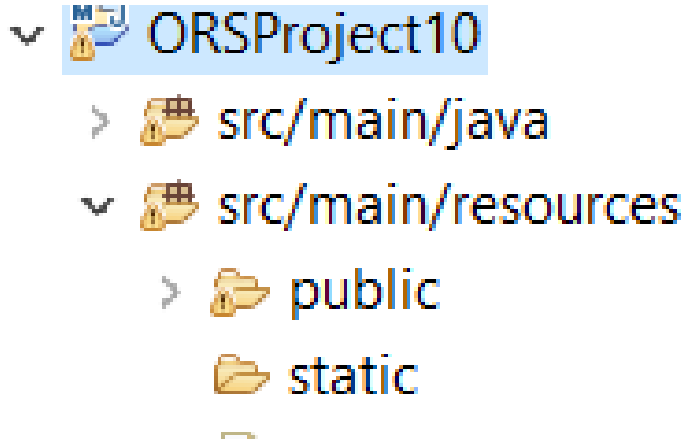
Static Resources

```
@Bean
public WebMvcConfigurer staticResource() {
  WebMvcConfigurer w = new WebMvcConfigurer() {
    @Override
    public void addResourceHandlers(ResourceHandlerRegistry reg){
ResourceHandlerRegistration hander = reg.addResourceHandler("/**");
          hander.addResourceLocations("classpath:/public/");
    }
  };
  return w;
}
```

Example

http/localhost:8080/a.jpg

http/localhost:8080/m.html



Scheduling

- To schedule job in spring boot application to run periodically, spring boot provides *@EnableScheduling* and *@Scheduled* annotations

- @SpringBootApplication
- **@EnableScheduling**
- public class Application {
- }

Scheduler

```
@Component
public class ScheduledTasks {

  @Scheduled(fixedDelay = 2000)
  public void scheduleTaskWithFixedRate() {
    System.out.println("Fixed Delay Task :: Execution Time - {}" + new Date());
    try {
      TimeUnit.SECONDS.sleep(5);
    } catch (InterruptedException ex) {
      ex.printStackTrace();
    }
  }
}
```

Get Source code

https://github.com/sunilos/SpringBoot
https://github.com/sunilos/ORSProject10