**Spring Boot**

**Interceptor**

@Component

**public** **class** RequestResponseLoggingInterceptor **implements** **HandlerInterceptor** {

@Override

**public** **boolean** **preHandle**(**final** HttpServletRequest request, **final** HttpServletResponse response, **final** Object handler) **throws** IOException {

System.***out***.println("Pre Handle");

**return** **true**;

}

@Override

**public** **void** **postHandle**(**final** HttpServletRequest request, **final** HttpServletResponse response, **final** Object handler,

@Nullable **final** ModelAndView modelAndView) {

System.***out***.println("Post Handle");

}

@Override

**public** **void** **afterCompletion**(**final** HttpServletRequest request, **final** HttpServletResponse response, **final** Object handler,

@Nullable **final** Exception ex) **throws** Exception {

}

}

@Configuration

**public** **class** WebMvcConfig **implements** **WebMvcConfigurer** {

@Autowired

RequestResponseLoggingInterceptor requestResponseLoggingInterceptor;

@Override

**public** **void** **addInterceptors**(**final** InterceptorRegistry registry) {

registry.addInterceptor(requestResponseLoggingInterceptor);

}

}

**Security**

<dependency>

    <groupId>org.springframework.boot</groupId>

    <artifactId>spring-boot-starter-security</artifactId>

</dependency>

spring.security.user.name

spring.security.user.password

@Configuration

**@EnableWebSecurity**

public class BasicConfiguration extends **WebSecurityConfigurerAdapter** {

    @Override

    protected void **configure**(AuthenticationManagerBuilder auth)

      throws Exception {

        auth

          .inMemoryAuthentication()

          .withUser("user")

            .password("password")

            .roles("USER")

            .and()

          .withUser("admin")

            .password("admin")

            .roles("USER", "ADMIN");

    }

    @Override

    protected void configure(HttpSecurity http) throws Exception {

        http

          .authorizeRequests()

          .anyRequest()

          .authenticated()

          .and()

          .httpBasic();

    }

}

**The @PathVariable annotation is used for data passed in the URI (e.g. RESTful web services) while @RequestParam is used to extract the data found in query parameters.**

**@PathParam is a JAX-RS annotation that is equivalent to @PathVariable in Spring.**

The **@RestController** annotation in Spring MVC is nothing but a combination of @Controller and @ResponseBody annotation.

**Rest Architecture**

Rest is a **client-server** architecture

Rest is stateless.

Rest is cacheable

Uniform Resource

Layered Architecture

**Spring boot cache annotations**

**@EnableCaching**

**@Cacheable**

@Cacheable(value="books", key="#isbn")

public Book findStoryBook(ISBN isbn, boolean checkWarehouse, boolean includeUsed)

**@CachePut**

this will allow us to update the cache and will also allow the method to be executed.

**@CacheEvict**

When **CacheEvict** annotated methods will be executed, it will clear the cache.

**@Caching**

This annotation is required when we need both CachePut and CacheEvict at the same time.