

Springboot: Custom Interceptors (Concept & Coding)

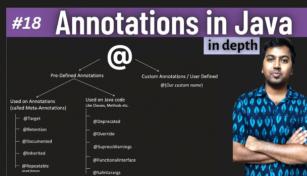
Interceptor:

it's a mediator, which get invoked before or after your actual code.

In future topics below, we might need to write our custom interceptors:

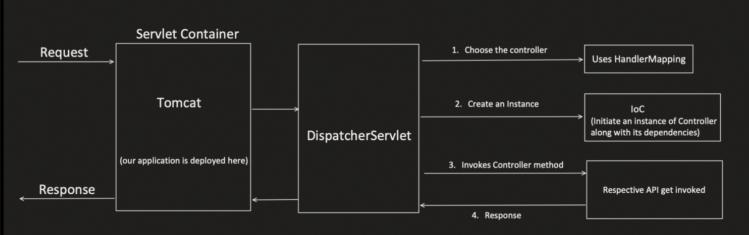
- *Springboot Caching,*
- *Springboot logging,*
- *Springboot Authentication etc..*

Pre-requisite to understand custom interceptor better:



[Report Abuse](#)

Custom Interceptor for Requests before even reaching to specific Controller class



```

@.RestController
@RequestMapping(value = "/api/")
public class UserController {
    @Autowired
    User user;

    @GetMapping(path = "/getuser")
    public String getUser() {
        user.getUser();
        return "success";
    }
}

@Configuration
public class AppConfig implements WebMvcConfigurer {

    @Autowired
    MyCustomInterceptor myCustomInterceptor;

    @Override
    public void addInterceptors(InterceptorRegistry registry) {
        registry.addInterceptor(myCustomInterceptor)
            .addPathPatterns("/*api/*") // Apply to these URL patterns
            .excludePathPatterns("/api/updateUser", "/api/deleteUser"); // Exclude for these URL patterns
    }
}

#Component
public class MyCustomInterceptor implements HandlerInterceptor {
    @Override
    public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws Exception {
        System.out.println("inside pre handle Method");
        return true;
    }

    @Override
    public void postHandle(HttpServletRequest request, HttpServletResponse response, Object handler, ModelAndView modelAndView) throws Exception {
        System.out.println("inside post handle method");
    }

    @Override
    public void afterCompletion(HttpServletRequest request, HttpServletResponse response, Object handler, @Nullable Exception ex) throws Exception {
        System.out.println("inside after completion method");
    }
}

http://localhost:8080/api/getUser

2024-08-31T23:01:49.217+05:30  INFO 4417 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet      : Completed initialization in 0 ms
inside pre handle Method
hitting db to get the userdata
inside post handle method
inside after completion method

```

Custom Interceptor for Requests after reaching to specific Controller class

Step1: Creation of custom annotation

We can create Custom Annotation using keyword "**@interface**" java Annotation.

```
public @interface MyCustomAnnotation {  
}
```

```
public class User {  
  
    @MyCustomAnnotation  
    public void updateUser(){  
        //some business logic  
    }  
}
```

2 Important Meta Annotation properties are:

- **@Target :**
this meta annotation, tells where we can apply the particular annotation on method or class or constructor etc.

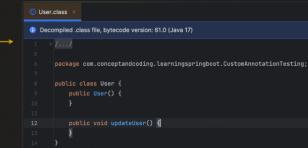
```
@Target(ElementType.METHOD)  
public @interface MyCustomAnnotation {  
}
```

```
@Target({ElementType.CONSTRUCTOR, ElementType.METHOD, ElementType.PARAMETER, ElementType.FIELD})  
public @interface MyCustomAnnotation {  
}
```

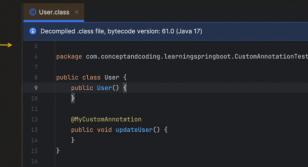
@Retention:
this meta annotation tell, how the particular annotation will be stored in java.

- **RetentionPolicy.SOURCE :**
Annotation will be discarded by compiler itself and its not even recorded in .class file.
- **RetentionPolicy.CLASS :**
Annotation will be recorded in .class file but ignored by JVM during run time.
- **RetentionPolicy.RUNTIME :**
Annotation will be recorded in .class file and also available during run time.

```
@Target(ElementType.METHOD)  
@Retention(RetentionPolicy.SOURCE)  
public @interface MyCustomAnnotation {  
}  
  
public class User {  
  
    @MyCustomAnnotation  
    public void updateUser(){  
        //some business logic  
    }  
}
```



```
@Target(ElementType.METHOD)  
@Retention(RetentionPolicy.RUNTIME)  
public @interface MyCustomAnnotation {  
}  
  
public class User {  
  
    @MyCustomAnnotation  
    public void updateUser(){  
        //some business logic  
    }  
}
```



How to create Custom Annotation with methods (more like a fields):

- No parameter, no body
- Return type is restricted to:
 - Primitive type (int, boolean, double etc.)
 - String
 - Enum
 - Class<?>
 - Annotations

```

Annotations
▪ Array of above types

public class User {

    @Target(ElementType.METHOD)
    @Retention(RetentionPolicy.RUNTIME)
    public @interface MyCustomAnnotation {
        String key() default "defaultKeyName";
    }
}

public class User {

    @MyCustomAnnotation(key = "userKey")
    public void updateUser(){
        //some business logic
    }
}

@Target(ElementType.METHOD)
@Retention(RetentionPolicy.RUNTIME)
public @interface MyCustomAnnotation {

    int intKey() default 0;
    String stringKey() default "defaultString";
    Class<?> classTypeKey() default String.class;
    MyCustomEnum enumKey() default MyCustomEnum.ENUM_VAL1;
    String[] stringArrayKey() default {"default1", "default2"};
    int[] intArrayKey() default {1, 2};
}

```

Step2: Creation of Custom Interceptor

```

@Target({ElementType.METHOD})
@Retention(RetentionPolicy.RUNTIME)
public @interface MyCustomAnnotation {

    String name() default "";
}

@Component
public class User {
    @MyCustomAnnotation(name = "user")
    public void getUser() {
        System.out.println("get the user details");
    }
}

@.RestController
@RequestMapping(value = "/api/")
public class UserController {

    @Autowired
    User user;

    @GetMapping(path = "/getUser")
    public String getUser(){
        user.getUser();
        return "SUCCESS";
    }
}

public class MyCustomInterceptor {

    @Around("@annotation(com.conceptandcoding.learningspringboot.CustomInterceptor.MyCustomAnnotation)") //pointcut expression
    public void invoke(ProceedingJoinPoint joinPoint) throws Throwable{ //advice

        System.out.println("do something before actual method");

        Method method = ((MethodSignature)joinPoint.getSignature()).getMethod();
        if(method.isAnnotationPresent(MyCustomAnnotation.class)){
            MyCustomAnnotation annotation = method.getAnnotation(MyCustomAnnotation.class);
            System.out.println("name from annotation: " + annotation.name());
        }

        joinPoint.proceed();
        System.out.println("do something after actual method");
    }
}

```

Output:

```

2024-08-31T17:51:51.344+05:30 INFO 3464 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet      : Completed initialization in 0 ms
do something before actual method
name from annotation: user
get the user details
do something after actual method

```