Handler Interceptor

=========

===========

in

(It is like Filters sevlet programming)

=> This comp logic can be added to controller comp as the extension hook logic ... which can be executed as pre, post and after logics with respect to controller classes execution

=> These are like servlet filter comps.. but servlet filter comps allow us to add only pre,post request processisng logics with respect to other web comps of the web application where as HandlerInterceptor allows us to add pre,post, after logics with respect to controller classes.

=> To develop Handler Interterceptor for Controller class we need to take a class implementing

HandlerInteceptor (1) which is having 3 methods (java 8 interface having all the 3 methods as the default methods)

controller class obj

`default void`

postHandle(HttpServletRequest request, HttpServletResponse response, Object handler, ModelAndView modelAndView) Interception point after successful execution of a handler. (controller) usecase :: sending addional model data having system date, time and etc..

`default boolean`

preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) Interception point before the execution of a handler. (controller) usecase:: time checking (trading b/w 9am to 3pm), allowing requests only from certain browser and etc..

`default void`

(if this method returns true

then the controller class in the flow executes otherwise the controller class in the flow does not execute)

```
afterCompletion (HttpServletRequest request, HttpServletResponse response, Object
handler, Exception ex) Callback after completion of request processing, that is, after
rendering the view. usecase: Nullifying request attributes,session attributes and
etc..
```

=>Since all the 3 methods are default methods.. So we can overide only those methods in which we are intrested in

request

Example App

response

step1) Keep any old app ready..

(MiniProject)

step2) Develop Handler Interceptor class. TimeCheckInterceptor.java

import java.time.LocalDateTime;

Dispatcher Servlet

Interceptor1 Interdeptor2 Controller class Interceptor3

```
import javax.servlet.RequestDispatcher;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.springframework.web.servlet.HandlerInterceptor;
public class TimeCheckInterceptor implements HandlerInterceptor {
@Override
if multiple interceptors are applied on the same controller class then they trap request in certain order before going to controller.. but they trap response
in revers order while leaving the controller (3,2,1)
=> we always use handler Interceptor
to place addtional and optional logics
to execute as extention hooks to controller class)
=> With out touching the source code of the controller class
if u want add pre, post, after completion logics we need to use the support of Handler interceptors
=> The logics kept is interceptor can be enabled or disabled on controller classes handler methods as the extension hooks with out disturbing the existing source code of the Project/App
=>All Handler interceptors must be registered with Handler Interceptor registry directly on indirectly
represents the current Controller/Handler class obj
public boolean preHandle(HttpServletRequest req, HttpServletResponse res, Object handler) throws Exception {
System.out.println("TimeCheckInterceptor.preHandle()");
if(req.getServletPath().equalsIgnoreCase("/")) // Do not apply this logic for home page.
return true;
// get System date and time
LocalDateTime ldt=LocalDateTime.now();
//get current hour of the day
int hour=ldt.getHour();
if(hour<9 || hour>17) {
RequestDispatcher rd=req.getRequestDispatcher("/timeout.jsp");
rd.forward(req, res);
on which this Interceptor is applied.. VeryUseful
to call other method controller class while writing Interceptor logics..
return false;
}
return true;
}
}
```

step4) place timeout.jsp page in src/main/web app folder

timeout.jsp

```
<%@ page isELIgnored="false" %>
<h1 style="color:red;text-align:center"> Trading timing are 9am to 5pm</h1>
<a href="./">home</a>
```

using

step5) Register Interceptor with Inteceptor registry by Web MVC configurer class as spring bean

MyWebMVCConfigurer.java

```
package com.nt.config;
import org.springframework.stereotype.Component;
import org.springframework.web.servlet.config.annotation.EnableWebMvc; import
org.springframework.web.servlet.config.annotation.InterceptorRegistry; import
org.springframework.web.servlet.config.annotation.WebMvcConfigurer; import
com.nt.interceptor.TimeCheckInterceptor;
@Component
public class MyWebMVCConfigurer implements WebMvcConfigurer {
@Override
public void addInterceptors(InterceptorRegistry registry) {
}
}
System.out.println("MyWebMVCConfigurer.addInterceptors()");
[registry.addInterceptor(new TimeCheckInterceptor());
```

=>This Configurer class will execute automatically

as part of application's startup and calls all the methods

as the callback methods automatically

=>This configurer allows us to override

multiple callback methods having different

registries using which we can register different

Custom services.

=>InterceptorRegistry to register the interceptors

=>ViewResolverRegistry to register the view resolvers

=>HandlerMappingRegistry to register the HandlerMapping comps and etc...

✓MMVCBootProj13-MiniProject-HandlerInterceptor [boot]

step6) run the web application..

› Spring Elements

> JAX-WS Web Services

#src/main/java

›

>

com.nt

BootMvcProj07MiniProjectCurd OperationsApplication.java ServletInitializer.java

com.nt.configurer

> MyWebMVCConfigurer.java

com.nt.controller

com.nt.interceptor

TimeCheckingJHandlerInterceptor.java

>com.nt.model

> com.nt.repository

>com.nt.service

>com.nt.validator

#src/main/resources

static templates

application.properties

Validation.properties

>

src/test/java

>

JRE System Library [JavaSE-1.8]

>

Maven Dependencies

>. Deployment Descriptor: MVCBootProj13-MiniProject-HandlerIntercep

› C Deployed Resources

✓ src

main

> java

>

resources

✓

webapp

>

css

> images

>test

WEB-INF

timeout.jsp

› target

W HELP.md

mvnw

mvnw.cmd

pombami

Chrome

Assignment:: Develope interceptors based spring boot mvc app that restricts other than browser requests

TimeCheckingInterceptor.java

```
//TimeCheckingIntercetpro.java
package com.nt.interceptor;
import java.time.LocalTime;
import org.springframework.stereotype.Component;
import org.springframework.web.servlet.HandlerInterceptor; import
org.springframework.web.servlet.ModelAndView;
import jakarta.servlet.RequestDispatcher;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
@Component
public class TimeCheckingInterceptor implements HandlerInterceptor {
@Override
public boolean preHandle(HttpServletRequest req, HttpServletResponse res, Object handler)
throws Exception {
System.out.println("TimeCheckingInterceptor.preHandle()");
//get current reuqest request path
String path=req.getServletPath();
if(!path.equalsIgnoreCase("/")) {
//get System Time
LocalTime lt=LocalTime.now();
//get current hour of the day
int hour=lt.getHour();
if(hour<9 || hour>17) {
RequestDispatcher rd=req.getRequestDispatcher("/timeout.jsp");
rd.forward(req, res);
}
return false;
}//if
}//if
```

```
return true;
//MyWebMVCConfigurer.java
package com.nt.configurer;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
import org.springframework.web.servlet.config.annotation.InterceptorRegistry; import
org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
import com.nt.interceptor.BrowserCheckingInterceptor;
import com.nt.interceptor.TimeCheckingInterceptor;
@Component
public class MyWebMVCConfigurer implements WebMvcConfigurer {
@Autowired
private TimeCheckingInterceptor tci;
@Autowired
private BrowserCheckingInterceptor bci;
}
@Override
public void addInterceptors(Interceptor Registry registry) {
registry.addInterceptor(tci);
registry.addInterceptor(bci);
}
```

note: if we configure multiple interceptors on the handler methods of the controller class then the preHandle() methods of those interceptors execute in the order Interceptors are configured and the postHandle()and afterCompletion() methods execute in the reverse order interceptor configuration