**SpringBoot中实现拦截器级别URl过快访问拦截**

新建一个拦截器类，并实现对URL访问过快的拦截，并将那些攻击者的IP加入黑名单中去。例子代码如下：

package example.Interceptor;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import example.controller.exception.RequestLimitException;

import example.controller.limit.RequestLimit;

import example.controller.limit.RequestLimitContract;

import example.dao.BlacklistDao;

import example.entity.Blacklist;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Component;

import org.springframework.stereotype.Controller;

import org.springframework.stereotype.Service;

import org.springframework.web.servlet.HandlerInterceptor;

import org.springframework.web.servlet.ModelAndView;

import java.util.\*;

/\*\*

\* 自定义拦截器1

\*/

public class URLInterceptor implements HandlerInterceptor {

@Autowired

private BlacklistDao blacklistDao;

private Map<String, Integer> redisTemplate=new HashMap<String,Integer>();

private static final Logger logger = LoggerFactory.getLogger("RequestLimitLogger");

@Override

public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler)

throws Exception {

return true;

}

@Override

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

ModelAndView modelAndView) throws Exception {

String ip = request.getLocalAddr();

List<Blacklist> blackList =blacklistDao.findByIp(ip);

if(blackList==null || blackList.size()==0){

urlHandle(request,10000,10);

}else{

modelAndView.setViewName("/errorpage/error");

}

}

@Override

public void afterCompletion(HttpServletRequest request, HttpServletResponse response, Object handler, Exception ex)

throws Exception {

}

public void urlHandle(HttpServletRequest request, long limitTime,int limitCount) throws RequestLimitException {

try{

String ip = request.getLocalAddr();

String url = request.getRequestURL().toString();

String key = "req\_limit\_".concat(url).concat(ip);

if(redisTemplate.get(key)==null || redisTemplate.get(key)==0){

redisTemplate.put(key,1);

}else{

redisTemplate.put(key,redisTemplate.get(key)+1);

}

int count = redisTemplate.get(key);

if (count > 0) {

Timer timer= new Timer();

TimerTask task = new TimerTask(){

@Override

public void run() {

redisTemplate.remove(key);

}

};

timer.schedule(task, limitTime);

}

if (count > limitCount) {

addHostHandle(ip);

throw new RequestLimitException();

}

} catch (RequestLimitException e) {

throw e;

} catch (Exception e) {

logger.error("发生异常: ", e);

}

}

public void addHostHandle(String ip){

Calendar calendar = Calendar.getInstance();

Date iptime=calendar.getTime();

Blacklist blacklist=new Blacklist(ip,iptime);

blacklistDao.save(blacklist);

}

}

在拦截器添加类中加入bean方法，否则JPA不能自动注入成功，因为容器还未加载就已经实现拦截器的功能了。例子代码如下：

package example.configuration;

import example.Interceptor.ErrorInterceptor;

import example.Interceptor.URLInterceptor;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.web.servlet.HandlerInterceptor;

import org.springframework.web.servlet.config.annotation.InterceptorRegistry;

import org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;

@Configuration

public class MyWebAppConfigurer

extends WebMvcConfigurerAdapter {

@Bean

public HandlerInterceptor getMyInterceptor(){

return new URLInterceptor();

}

@Override

public void addInterceptors(InterceptorRegistry registry) {

// 多个拦截器组成一个拦截器链

// addPathPatterns 用于添加拦截规则

// excludePathPatterns 用户排除拦截

registry.addInterceptor(getMyInterceptor()).addPathPatterns("/\*\*");

super.addInterceptors(registry);

}

}

异常处理类

package example.controller.exception;

public class RequestLimitException extends Exception {

private static final long serialVersionUID = 1364225358754654702L;

public RequestLimitException() {

super("HTTP请求超出设定的限制");

}

public RequestLimitException(String message) {

super(message);

}

}

实现一个控制类，并添加使用注解功能。下面看代码事例：

package example.controller;

import example.controller.limit.RequestLimit;

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.ResponseBody;

import javax.servlet.http.HttpServletRequest;

@Controller

public class URLController {

@RequestMapping("/urltest")

@ResponseBody

public String test(HttpServletRequest request, ModelMap modelMap) {

return "aaa";

}

}