在Web中最经常发生的就是利用恶性URL访问刷爆服务器之类的攻击，今天我就给大家介绍一下如何利用自定义注解实现这类攻击的防御操作。

其实这类问题一般的解决思路就是：在控制器中加入自定义注解实现访问次数限制的功能。

具体的实现过程看下面的例子：

步骤一：先定义一个注解类，下面看代码事例：

package example.controller.limit;

import org.springframework.core.Ordered;

import org.springframework.core.annotation.Order;

import java.lang.annotation.\*;

@Retention(RetentionPolicy.RUNTIME)

@Target(ElementType.METHOD)

@Documented

//最高优先级

@Order(Ordered.HIGHEST\_PRECEDENCE)

public @interface RequestLimit {

/\*\*

\*

\* 允许访问的次数，默认值MAX\_VALUE

\*/

int count() default Integer.MAX\_VALUE;

/\*\*

\*

\* 时间段，单位为毫秒，默认值一分钟

\*/

long time() default 60000;

}

步骤二：定义一个异常类，用来处理URL攻击时产生的异常问题，下面看代码事例：

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);

}

}

步骤三：定义一个注解的具体实现类，下面看代码事例：

package example.controller.limit;

import example.controller.exception.RequestLimitException;

import org.aspectj.lang.JoinPoint;

import org.aspectj.lang.annotation.Aspect;

import org.aspectj.lang.annotation.Before;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.stereotype.Component;

import javax.servlet.http.HttpServletRequest;

import java.util.HashMap;

import java.util.Map;

import java.util.Timer;

import java.util.TimerTask;

import java.util.concurrent.TimeUnit;

@Aspect

@Component

public class RequestLimitContract {

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

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

@Before("within(@org.springframework.stereotype.Controller \*) && @annotation(limit)")

public void requestLimit(final JoinPoint joinPoint, RequestLimit limit) throws RequestLimitException {

try {

Object[] args = joinPoint.getArgs();

HttpServletRequest request = null;

for (int i = 0; i < args.length; i++) {

if (args[i] instanceof HttpServletRequest) {

request = (HttpServletRequest) args[i];

break;

}

}

if (request == null) {

throw new RequestLimitException("方法中缺失HttpServletRequest参数");

}

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(){ //创建一个新的计时器任务。

@Override

public void run() {

redisTemplate.remove(key);

}

};

timer.schedule(task, limit.time());

//安排在指定延迟后执行指定的任务。task : 所要安排的任务。10000 : 执行任务前的延迟时间，单位是毫秒。

}

if (count > limit.count()) {

//logger.info("用户IP[" + ip + "]访问地址[" + url + "]超过了限定的次数[" + limit.count() + "]");

throw new RequestLimitException();

}

} catch (RequestLimitException e) {

throw e;

} catch (Exception e) {

logger.error("发生异常: ", e);

}

}

}

步骤四：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 {

@RequestLimit(count=10,time=5000)

@RequestMapping("/urltest")

@ResponseBody

public String test(HttpServletRequest request, ModelMap modelMap) {

return "aaa";

}

}

其中count指的是规定时间内的访问次数，time指的就是规定时间，单位为毫秒。

这样就实现了在控制器这个层次上面的url拦截了。不过这里有个问题，就是如果想在每一个URL页面上面都进行这样的拦截，这种方法明显是不够的。因为我们不可能在每个控制器上面都加上url拦截的注解，所以这种方法只适合在某些特定的URL拦截上面使用它们。