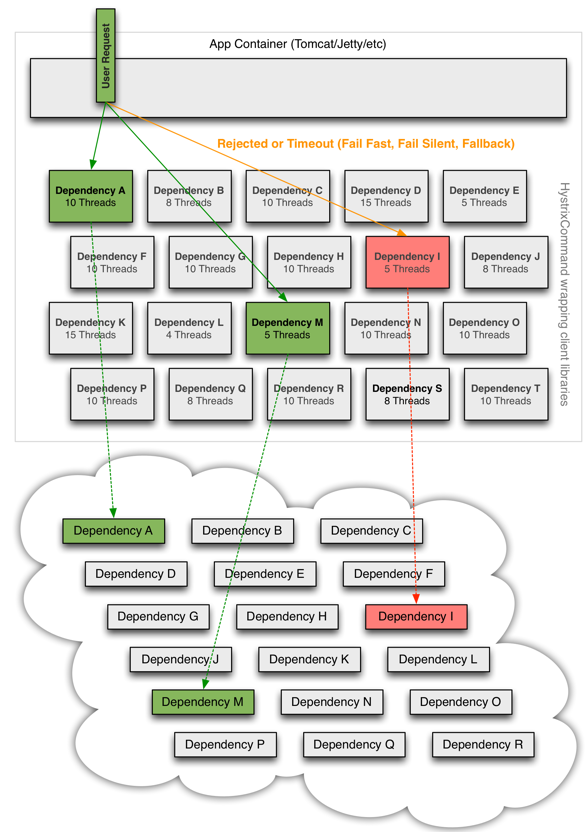
# 线程隔离，服务降级

## 原理

线程隔离示意图：



解读：

Hystrix为每个依赖服务调用分配一个小的线程池，如果线程池已满调用将被立即拒绝，默认不采用排队.加速失败判定时间。

用户的请求将不再直接访问服务，而是通过线程池中的空闲线程来访问服务，如果线程池已满，或者请求超时，则会进行降级处理，什么是服务降级？

## 服务降级



用户的请求故障时，不会被阻塞，更不会无休止的等待或者看到系统崩溃，至少可以看到一个执行结果（例如返回友好的提示信息） 。

服务降级虽然会导致请求失败，但是不会导致阻塞，而且最多会影响这个依赖服务对应的线程池中的资源，对其它服务没有响应。

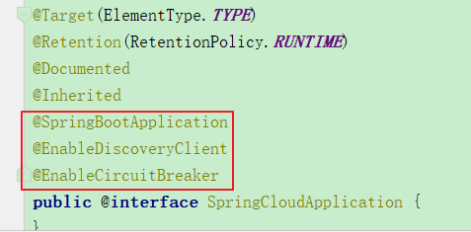
## 触发Hystix服务降级的情况：

- 线程池已满

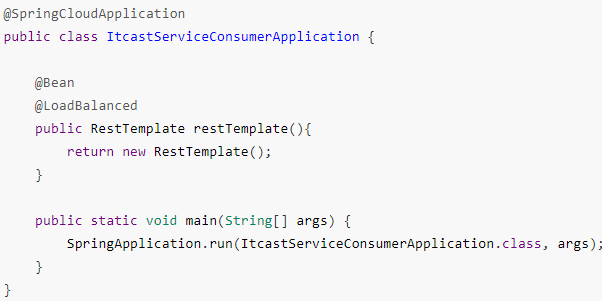
- 请求超时

## @SpringCloudApplication

可以看到，我们类上的注解越来越多，在微服务中，经常会引入上面的三个注解，于是Spring就提供了一个组合注解：@SpringCloudApplication



因此，我们可以使用这个组合注解来代替之前的3个注解。



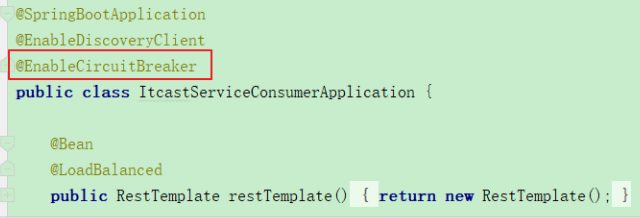
# 动手实践

## 引入依赖

首先在itcast-service-consumer的pom.xml中引入Hystrix依赖：



## 开启熔断



## 编写降级逻辑

我们改造itcast-service-consumer，当目标服务的调用出现故障，我们希望快速失败，给用户一个友好提示。因此需要提前编写好失败时的降级处理逻辑，要使用HystixCommond来完成：



要注意，因为熔断的降级逻辑方法必须跟正常逻辑方法保证：相同的参数列表和返回值声明。失败逻辑中返回User对象没有太大意义，一般会返回友好提示。所以我们把queryById的方法改造为返回String，反正也是Json数据。这样失败逻辑中返回一个错误说明，会比较方便。

说明：

- @HystrixCommand(fallbackMethod = "queryByIdFallBack")：用来声明一个降级逻辑的方法

测试：

当itcast-service-provder正常提供服务时，访问与以前一致。但是当我们将itcast-service-provider停机时，会发现页面返回了降级处理信息：

## 默认FallBack

我们刚才把fallback写在了某个业务方法上，如果这样的方法很多，那岂不是要写很多。所以我们可以把Fallback配置加在类上，实现默认fallback：

@Controller

@RequestMapping("consumer/user")

@DefaultProperties(defaultFallback = "fallBackMethod") // 指定一个类的全局熔断方法

public class UserController {

@Autowired

private RestTemplate restTemplate;

@GetMapping

@ResponseBody

@HystrixCommand // 标记该方法需要熔断

public String queryUserById(@RequestParam("id") Long id) {

String user = this.restTemplate.getForObject("http://service-provider/user/" + id, String.class);

return user;

}

/\*\*

\* 熔断方法

\* 返回值要和被熔断的方法的返回值一致

\* 熔断方法不需要参数

\* @return

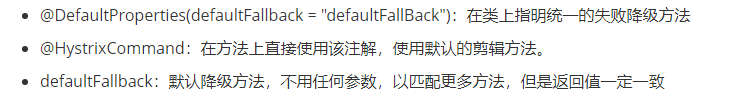
\*/

public String fallBackMethod(){

return "请求繁忙，请稍后再试！";

}

}





## 设置超时

在之前的案例中，请求在超过1秒后都会返回错误信息，这是因为Hystix的默认超时时长为1，我们可以通过配置修改这个值：

我们可以通过hystrix.command.default.execution.isolation.thread.timeoutInMilliseconds来设置Hystrix超时时间。该配置没有提示。



改造服务提供者

改造服务提供者的UserController接口，随机休眠一段时间，以触发熔断：

