关于cat的部署与使用

1. 需要的第三方工具：maven，mysql数据库
2. 部署步骤：

(1).编译cat项目

mvn clean install

(2).启动mysql数据库(让开机自启动使用chkconfig命令查看)

service mysqld start

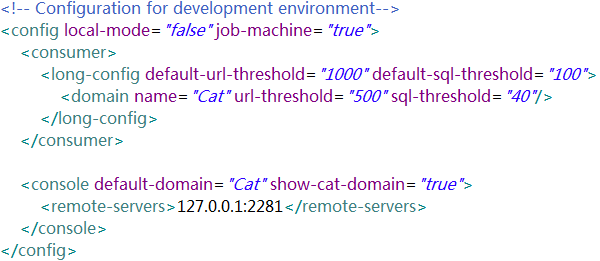
(3).安装cat环境

mvn cat:install

当中要输入mysql数据库连接，直到出现success为止

(4).修改cat环境变量(不使用hadoop不需要配置hadoop选项)

/data/appdatas/cat/server.xml



a).local-mode=”true” job-machine=”true”

这种模式将可以在cat展示中能看到依赖关系图表，大盘监控图表

b).remote-servers为指定的web监控平台浏览地址

(5).启动cat监控(监控系统与展示系统一体)

mvn jetty:run

Cat监控平台浏览地址http://ip:2281/cat

监控端口2280

1. Cat客户端api的埋点实例

埋点原则：

首先必须在所有的要监控的步骤外部包围一个大的Transaction t(后面用t表示)，t主要用来跟踪业务耗时等指标。

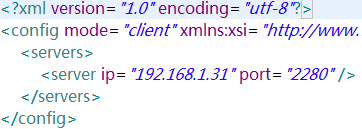
然后再在包围的t中设定具体跟踪的Event e(后面用e表示)，e主要用来跟踪业务详细指标。

如果碰到在某个t中还需要跟踪细分的业务耗时指标，则可以在外围t中嵌套一个t，t可以不断的嵌套。

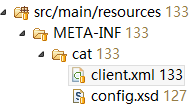
客户端常量配置

Java配置：

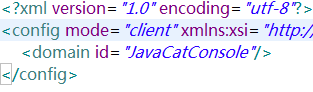
1).在/data/appdatas/cat/client.xml配置cat的服务器地址



2).然后在开发的项目包中加入如下目录结构的文件



其中，client.xml中只需要配置项目的名称即可，如下图

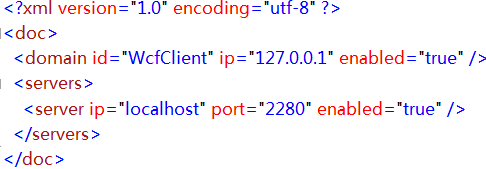


domain节点中的id就是你的项目名称。

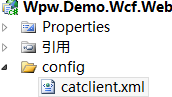
C#配置：

1).配置文件结构如下，此配置文件需要加到c#项目中

配置文件结构：



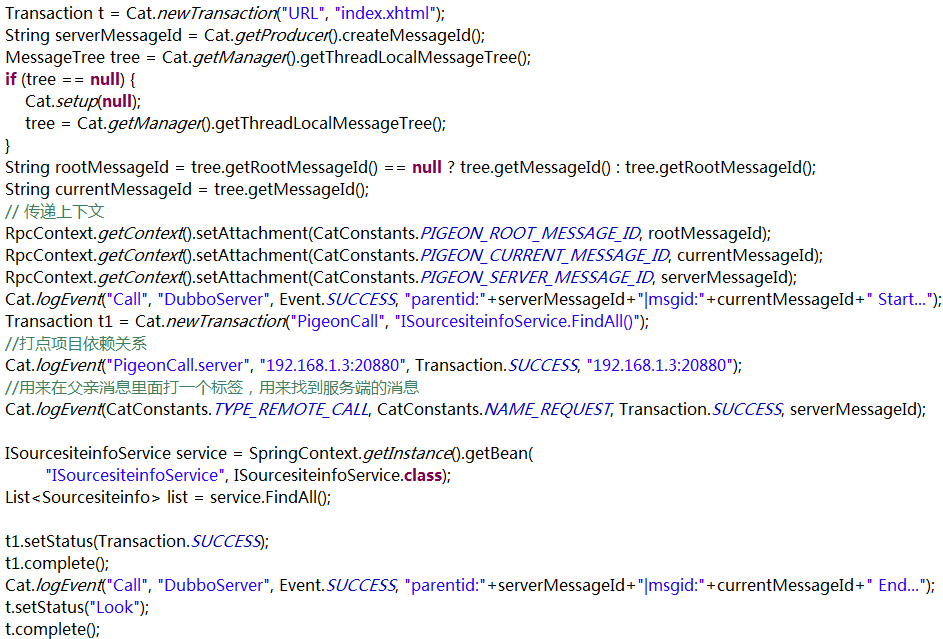
项目结构：



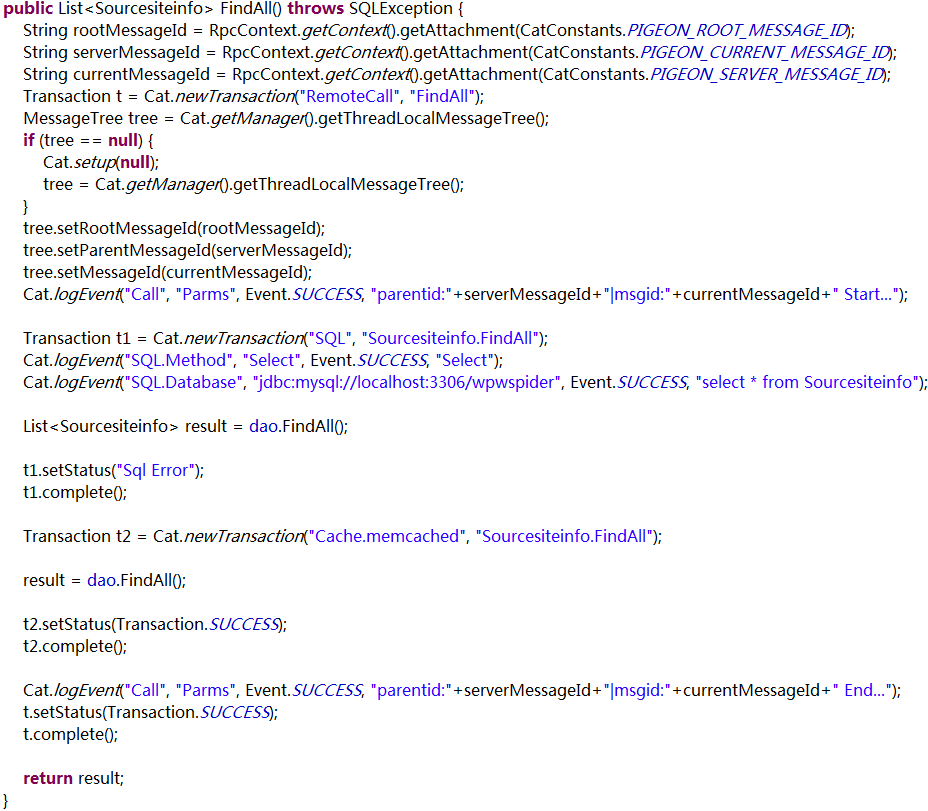
Java打点例子：以dubbo的一个远程调用为例

(1).实例

Dubbo的客户端打点：



Dubbo服务器端打点：



(2).说明

a).远程方法调用的打点，需要跟踪远程服务端的日志行为，dubbo客户端埋点规则为：

首先定义上下文的消息id，注意详细的id名称，不要搞错

String serverMessageId = Cat.getProducer().createMessageId();

MessageTree tree = Cat.getManager().getThreadLocalMessageTree();

if (tree == null) {

Cat.setup(null);

tree = Cat.getManager().getThreadLocalMessageTree();

}

String rootMessageId = tree.getRootMessageId() == null ? tree.getMessageId() : tree.getRootMessageId();

String currentMessageId = tree.getMessageId();

// 传递上下文

RpcContext.getContext().setAttachment(CatConstants.PIGEON\_ROOT\_MESSAGE\_ID, rootMessageId);

RpcContext.getContext().setAttachment(CatConstants.PIGEON\_CURRENT\_MESSAGE\_ID, currentMessageId);

RpcContext.getContext().setAttachment(CatConstants.PIGEON\_SERVER\_MESSAGE\_ID, serverMessageId);

然后声明一个远程调用的transaction

Transaction t1 = Cat.newTransaction("PigeonCall", "ISourcesiteinfoService.FindAll()");

再后声明依赖远程服务器的ip端口等，注：第一个参数必须声明为PigeonCall.server，否则cat无法定位依赖关系，ip和端口必须对应一个远程服务的地址

Cat.logEvent("PigeonCall.server","192.168.1.3:20880",Transaction.SUCCESS,"192.168.1.3:20880");

b).dubbo服务器端埋点规则为：

首先接受上下文传递过来的消息id

String rootMessageId = RpcContext.getContext().getAttachment(CatConstants.PIGEON\_ROOT\_MESSAGE\_ID);

String serverMessageId = RpcContext.getContext().getAttachment(CatConstants.PIGEON\_CURRENT\_MESSAGE\_ID);

String currentMessageId = RpcContext.getContext().getAttachment(CatConstants.PIGEON\_SERVER\_MESSAGE\_ID);

然后声明一个transaction

Transaction t = Cat.newTransaction("RemoteCall", "FindAll");

再设定当前消息树的id，注意与客户端的消息树id的对应关系

MessageTree tree = Cat.getManager().getThreadLocalMessageTree();

if (tree == null) {

Cat.setup(null);

tree = Cat.getManager().getThreadLocalMessageTree();

}

tree.setRootMessageId(rootMessageId);

tree.setParentMessageId(serverMessageId);

tree.setMessageId(currentMessageId);

c).关于数据库的操作埋点

Transaction t1 = Cat.newTransaction("SQL", "Sourcesiteinfo.FindAll");

Cat.logEvent("SQL.Method", "Select", Event.SUCCESS, "Select");

Cat.logEvent("SQL.Database", "jdbc:mysql://localhost:3306/wpwspider", Event.SUCCESS, "select \* from Sourcesiteinfo");

注：event的第一个参数必须如上规则，数据库连接字符串的规则必须是个典型的jdbc的uri

d).关于缓存的埋点

Transaction t2 = Cat.newTransaction("Cache.memcached", "Sourcesiteinfo.FindAll");

注：transation的第一个参数必须是Cache.memcached或者Cache.web

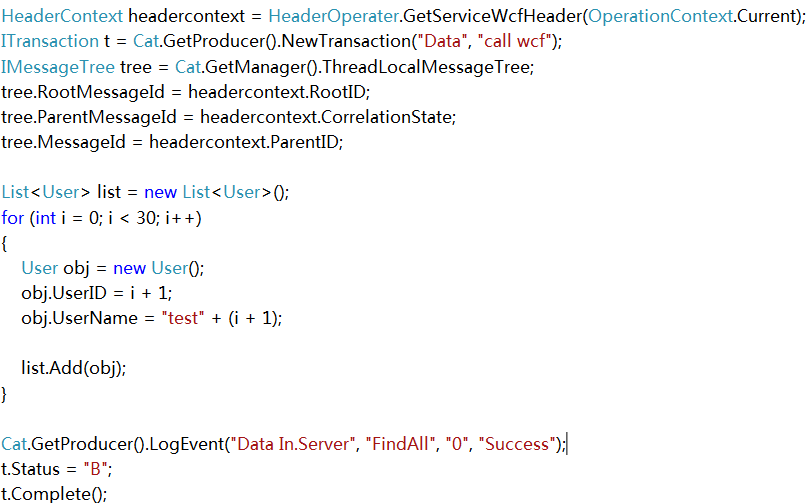
C#打点例子：以wcf的一个远程调用为例

(1).实例

Wcf客户端打点



Wcf服务器端打点



2).cat的.net客户端api的使用方法如上，打点的规则请参照java版本。