Java事件函数

更新时间：2020-08-31 14:20:21

本页目录

* [背景信息](https://help.aliyun.com/document_detail/113518.html?spm=a2c4g.11186631.6.582.636c330bc02Uh5#h2-url-1)
* [事件函数接口](https://help.aliyun.com/document_detail/113518.html?spm=a2c4g.11186631.6.582.636c330bc02Uh5#title-971-or2-pgr)
* [StreamRequestHandler](https://help.aliyun.com/document_detail/113518.html?spm=a2c4g.11186631.6.582.636c330bc02Uh5#title-3lm-72q-564)
* [PojoRequestHandler](https://help.aliyun.com/document_detail/113518.html?spm=a2c4g.11186631.6.582.636c330bc02Uh5#title-9o5-wmg-rfb)
* [使用context](https://help.aliyun.com/document_detail/113518.html?spm=a2c4g.11186631.6.582.636c330bc02Uh5#title-s6q-rru-l2z)

在函数计算服务使用Java编程，需要定义一个Java函数作为入口。本文介绍了Java事件函数的结构和特点。

**背景信息**

函数计算支持Java8运行环境。Java语言由于需要编译后才可以在JVM虚拟机中运行。和Python、Node.js这类脚本型语言不同，Java语言有以下限制：

* 不支持上传代码：仅支持上传已经开发完成、编译打包后的ZIP包或JAR包。函数计算不提供Java的编译能力。
* 不支持在线编辑：由于不支持上传代码，所以不支持在线编辑代码，仅能看到通过**页面上传**或**OSS上传**两种方法提交代码。

**事件函数接口**

您在使用Java编程时，必须要实现函数计算提供的接口类，对于事件入口函数目前有两个预定义接口可以选择。这两个预定义接口分别是：

* [StreamRequestHandler](https://help.aliyun.com/document_detail/113518.html?spm=a2c4g.11186631.6.582.636c330bc02Uh5#section-f71-r3j-0vu)

以流的方式接受调用输入event和返回执行结果，您需要从输入流中读取调用函数时的输入，处理完成后把函数执行结果写入到输出流中来返回。

* [PojoRequestHandler](https://help.aliyun.com/document_detail/113518.html?spm=a2c4g.11186631.6.582.636c330bc02Uh5#section-llu-srm-3k2)

通过泛型的方式，您可以自定义输入和输出的类型，但是输入和输出的类型必须是[POJO](https://en.wikipedia.org/wiki/Plain_old_Java_object)类型。

**StreamRequestHandler**

一个最简单的事件函数定义如下所示。

**package** example;

**import** com.aliyun.fc.runtime.Context;

**import** com.aliyun.fc.runtime.StreamRequestHandler;

**import** java.io.IOException;

**import** java.io.InputStream;

**import** java.io.OutputStream;

**public** **class** **HelloFC** **implements** **StreamRequestHandler** {

@Override

**public** **void** **handleRequest**(

InputStream inputStream, OutputStream outputStream, Context context) **throws** IOException {

outputStream.write(**new** String("hello world").getBytes());

}

}

* 包名和类名

由于Java有包的概念，因此执行方法和其他语言有所不同，需要带有包信息。代码例子中对应的执行方法为example.Hello::mainHandler，此处example标识为Java package，Hello标识为类，mainHandler标识为类方法。

包名和类名可以是任意的，但是需要与创建函数时的函数入口（handler）字段相对应。上述的例子包名是example，类名是HelloFC，那么创建函数时指定的handler为example.HelloFC::handleRequest，handler的格式为{package}.{class}::{method}。

* 实现的接口

您的代码中必须要实现函数计算预定义的接口。上述的代码示例中实现了StreamRequestHandler，其中的**inputStream**参数是调用函数时传入的数据，**outputStream**参数用于返回函数的执行结果。

* context参数

context参数中包含一些函数的运行时信息（例如requestId、临时AccesKey等），其类型是com.aliyun.fc.runtime.Context。

* 返回值

实现StreamRequestHandler接口的函数通过outputStream参数返回执行结果。

* 引入接口库

其中用到的 com.aliyun.fc.runtime这个包的依赖可以通过下文的 pom.xml引用。

<dependency>

<groupId>com.aliyun.fc.runtime</groupId>

<artifactId>fc-java-core</artifactId>

<version>1.3.0</version>

</dependency>

通过[maven仓库](https://mvnrepository.com/artifact/com.aliyun.fc.runtime/fc-java-core)可以获取fc-java-core最新的版本号。

在创建函数之前，您需要将代码及其依赖的fc-java-core打成JAR包。打包方式，请参见[使用自定义模块](https://help.aliyun.com/document_detail/113519.html#section-1qe-h7z-791)。示例代码包是示例中的hello world代码打包成的JAR包，您可以直接使用[示例代码包](http://docs-aliyun.cn-hangzhou.oss.aliyun-inc.com/assets/attach/58887/cn_zh/1533288706834/example.jar)进行测试。

使用 fun deploy 命令进行部署。执行成功时，会看到以下相关日志。

**using** region: cn-hangzhou

**using** accountId: \*\*\*\*\*\*\*\*\*\*\*3557

**using** accessKeyId: \*\*\*\*\*\*\*\*\*\*\*r3Ra

**using** timeout: 300

Waiting **for** service FunDemo to be deployed...

Waiting **for** **function** **javademo** **to** **be** **deployed**...

Waiting **for** packaging **function** **javademo** **code**...

package **function** **javademo** **code** **done**

**function** **javademo** **deploy** **success**

service FunDemo deploy success

登录[函数计算控制台](http://https:/fc.console.aliyun.com)，您就可以查看函数的状态并调用函数了。

**PojoRequestHandler**

一个最简单的处理函数定义如下所示。

// HelloFC.java

**package** example;

**import** com.aliyun.fc.runtime.Context;

**import** com.aliyun.fc.runtime.PojoRequestHandler;

**public** **class** **HelloFC** **implements** **PojoRequestHandler**<**SimpleRequest**, **SimpleResponse**> {

@Override

**public** SimpleResponse **handleRequest**(SimpleRequest request, Context context) {

String message = "Hello, " + request.getFirstName() + " " + request.getLastName();

**return** **new** SimpleResponse(message);

}

}

// SimpleRequest.java

**package** example;

**public** **class** **SimpleRequest** {

String firstName;

String lastName;

**public** String **getFirstName**() {

**return** firstName;

}

**public** **void** **setFirstName**(String firstName) {

**this**.firstName = firstName;

}

**public** String **getLastName**() {

**return** lastName;

}

**public** **void** **setLastName**(String lastName) {

**this**.lastName = lastName;

}

**public** **SimpleRequest**() {}

**public** **SimpleRequest**(String firstName, String lastName) {

**this**.firstName = firstName;

**this**.lastName = lastName;

}

}

// SimpleResponse.java

**package** example;

**public** **class** **SimpleResponse** {

String message;

**public** String **getMessage**() {

**return** message;

}

**public** **void** **setMessage**(String message) {

**this**.message = message;

}

**public** **SimpleResponse**() {}

**public** **SimpleResponse**(String message) {

**this**.message = message;

}

}

准备调用的输入文件如下。

{

"firstName": "FC",

"lastName": "aliyun"

}

**使用context**

context是函数计算在运行时生成的一个对象，包含一些运行时的信息，您可以在代码中可以使用这些信息。context的具体实现请参见[fc-java-libs](https://github.com/aliyun/fc-java-libs/blob/master/fc-java-core/src/main/java/com/aliyun/fc/runtime/Context.java)，其定义如下所示。

**package** com.aliyun.fc.runtime;

**public** **interface** **Context** {

**public** String **getRequestId**();

**public** Credentials **getExecutionCredentials**();

**public** FunctionParam **getFunctionParam**();

**public** FunctionComputeLogger **getLogger**();

}

您可以看到context中包含了以下信息。

| **信息类型** | **信息类型说明** |
| --- | --- |
| RequestId | 本次调用请求的唯一ID，您可以把它记录下来在出现问题的时候方便查询。 |
| Function | 当前调用的函数的一些基本信息，例如函数名、函数入口、函数内存和超时时间。 |
| Credentials | 函数计算服务通过扮演您提供服务角色的获得的一组临时密钥，其有效时间是5分钟。您可以在代码中使用Credentials去访问相应的服务（ 例如OSS ），这就避免了您把自己的AccessKey信息硬编码在函数代码里。详情请参见[服务角色](https://help.aliyun.com/document_detail/52885.html#section-hpk-d6y-hla)。 |
| Logger | 函数计算封装过的logger。 |

下文的代码演示了如何使用临时密钥向OSS中上传一个文件。

**package** example;

**import** com.aliyun.fc.runtime.Context;

**import** com.aliyun.fc.runtime.Credentials;

**import** com.aliyun.fc.runtime.StreamRequestHandler;

**import** com.aliyun.oss.OSSClient;

**import** java.io.ByteArrayInputStream;

**import** java.io.IOException;

**import** java.io.InputStream;

**import** java.io.OutputStream;

**public** **class** **HelloFC** **implements** **StreamRequestHandler** {

@Override

**public** **void** **handleRequest**(

InputStream inputStream, OutputStream outputStream, Context context) **throws** IOException {

String endpoint = "oss-cn-shanghai.aliyuncs.com";

String bucketName = "my-bucket";

Credentials creds = context.getExecutionCredentials();

OSSClient client = **new** OSSClient(

endpoint, creds.getAccessKeyId(), creds.getAccessKeySecret(), creds.getSecurityToken());

client.putObject(bucketName, "my-object", **new** ByteArrayInputStream(**new** String("hello").getBytes()));

outputStream.write(**new** String("done").getBytes());

}

}