Java Initializer函数

更新时间：2020-09-15 15:45:31

本页目录

* [预定义接口示例](https://help.aliyun.com/document_detail/159074.html?spm=a2c4g.11186623.6.584.f1004316iB6Hke#title-82j-mvv-ikl)
* [简单流式示例](https://help.aliyun.com/document_detail/159074.html?spm=a2c4g.11186623.6.584.f1004316iB6Hke#title-g6n-yyt-uk1)
* [泛型示例](https://help.aliyun.com/document_detail/159074.html?spm=a2c4g.11186623.6.584.f1004316iB6Hke#title-gb7-1u7-ads)

Initializer函数是实例的初始化函数，保证同一实例成功且仅成功执行一次。本文介绍Java Initializer函数结构信息与相关示例。

**预定义接口示例**

无论您的函数使用流式输入还是通过泛型的方式自定义输入和输出，当需要在Java runtime中添加initializer接口时，都需要在原有的基础上额外实现initializer预定义的接口。

[Initializer函数](https://help.aliyun.com/document_detail/157704.html#section-fn9-a36-vik)预定义接口如下所示。

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

**import** java.io.IOException;

**public** **interface** **FunctionInitializer** {

/\*\*

\* The interface to handle a function compute initialize request

\*

\* **@param** context The function compute initialize environment context object.

\* **@throws** IOException IOException during I/O handling

\*/

**void** **initialize**(Context context) **throws** IOException;

}

**简单流式示例**

一个简单的流式输入的函数和Initializer函数结合的示例如下所示。

**package** aliyun.serverless.test.example;

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

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

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

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

**import** java.io.IOException;

**import** java.io.InputStream;

**import** java.io.OutputStream;

**public** **class** **InitializerAndStreamRequest** **implements** **StreamRequestHandler**, **FunctionInitializer** {

@Override

**public** **void** **initialize**(Context context) {

FunctionComputeLogger logger = context.getLogger();

logger.debug(String.format("RequestID is %s %n", context.getRequestId()));

}

@Override

**public** **void** **handleRequest**(InputStream input, OutputStream output, Context context) **throws** IOException {

FunctionComputeLogger logger = context.getLogger();

logger.debug(String.format("RequestID is %s %n", context.getRequestId()));

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

output.flush();

}

}

针对InitializerAndStreamRequest中新增加的initializer方法即是initializer接口，有以下特性：

* 包名/类名

initializer所属包名和类名可以是任意的。initializer的格式为{package}.{class}::{method}。根据定义可知此示例的initializer为aliyun.serverless.test.example.InitializerAndStreamRequest::initialize。

* 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。上文示例中initializer接口实现了FunctionInitializer，initializer接口只有一个context参数。

* context参数context参数中包含一些函数的运行时信息例如requestId、临时AccessKey等，其类型是com.aliyun.fc.runtime.Context。
* 返回值

实现FunctionInitializer接口的函数没有返回结果。

**泛型示例**

泛型的方式输入的函数和Initializer函数结合的示例如下所示。

**package** aliyun.serverless.test.example;

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

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

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

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

**public** **class** **InitializerAndPojoRequest** **implements** **FunctionInitializer**,**PojoRequestHandler**<**SimpleRequest**, **SimpleResponse**> {

@Override

**public** **void** **initialize**(Context context) {

FunctionComputeLogger logger = context.getLogger();

logger.debug(String.format("RequestID is %s %n", context.getRequestId()));

}

@Override

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

FunctionComputeLogger logger = context.getLogger();

logger.debug(String.format("RequestID is %s %n", context.getRequestId()));

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

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

}

}