文章
Hao Ma · 一月 10, 2021 阅读大约需 17 分钟

RESTful API (cn)

RESTful 应用程序编程接口 (API) 设计和文档编制初学者指南。 通过示例，您将学习一些常见的 RESTful API 模式。

在阅读之前

您需要知道

- 如何在 Ensemble 中创建 RESTful Web 服务
- 如何在 Ensemble 中使用 RESTful Web 服务
- 如何传递服务参数
- 如何返回服务结果

什么是服务 API？

什么是应用程序编程接口？ 是具体化的东西吗？ 是单一编程单元吗？ API 的作用是什么？ 在我看来，API 是由程序代码以间接方式决定的。 但完全定义的 API 是由运行可执行程序的容器（由部署设置控制）提供的。 因此，我宁愿将 API 定义为服务的公共描述。 该描述可以是人类可读的，也可以仅机器可读， 或者两者均可。 API 用于与将要使用服务的人员共享有关服务的基本信息。 API 说明了服务的作用、使用环境、功能以及管理的数据结构等。

在过去的好时光，“ 编制程序文档 ” 或多或少是一种 “ 必要之恶 ” 。
现代编程语言通过在程序源码中引入声明来强制编制文档。
虽然声明是 “ 机器人 ” 可读的文档，但通过使用工具（runoff、Java doc...），可以提取信息并将其格式化成人类可读的格式。
即使没有在源码中添加任何一行真正的文档，这些工具仍然能够生成少量文本。

现在有什么不同吗？ 并没有。 服务 API 仍然是一个抽象的概念，表示正常使用一个功能性计算机软件所需的信息集合。 有一些语言可以将 API 定义形式化，例如 Web 服务描述语言 (WSDL)。 遗憾的是，这些语言的使用受限。 原因并不在于，例如，WSDL 的能力不足以表达 RESTful API，而是因为非技术上的不匹配。 （用 XML 表达 JSON 结构会是怎样的？）
最终，没有像适用于 SOAP Web 服务的 WSDL 那样的适用于 REST 的实际标准语言。 很可惜。 是不是？

没关系。 无论如何，我们首先需要了解 API 编制的内容。

API 由什么组成？

服务的核心属性是什么？

- 服务位置。 服务的 URL 根路径。 例如 http://localhost:57774/csp/msa/person。
- 服务方法。 即服务的功能。 方法由 HTTP 标头中的动词（GET、POST、PUT…）与其他路径类型参数的组合定义。
- 接受的方法参数。 参数及其类型的列表。 类型可以是**路径**，表示 URL 路径中包含的参数；**查询**，表示编码的 URL 查询；**表单**，表示表单数据；**内容**，表示 HTTP 消息正文。

- 返回状态。 HTTP 响应标头中的状态字段。 每个服务方法可能有多个返回状态码。
  数字取决于服务方法和异常处理的粒度。
- 响应内容。 每个状态码的预期内容。 格式可能因状态码而异。 例如，成功完成某个请求后，预期获得一个 JSON 序列化对象。 如果服务器出错 (500)，将发送纯文本说明。

示例 API

首先，让我们试着描述一下我们要实现的目标。 我们要构建一个非常简单的服务。 一个注册表服务。
它将管理相同类型的资源。 例如人员。

结构非常简单：姓名、出生日期和地点、母亲的娘家姓和生成的内部唯一注册表 ID。
地点的结构如下：国家/地区、城市。

我们要向注册表插入一条新记录（更新完整的注册表条目），更新条目的各个属性（更新属性），删除条目，按注册表 ID 获取单个条目，根据属性匹配查询注册表 ID 列表。

我们还需要一些服务功能：初始化注册表，填充一些记录以进行测试。

从外部看，我们希望 http://localhost:57774/csp/msa/person 为服务位置。

在服务位置通过 **PUT** 添加新条目。 将注册表记录作为内容发送。 预期返回带注册表 ID 的完整条目。

通过 **POST** 更新。 URL 包含要更新的条目的注册表 ID。 要更新的属性以表单数据的形式发送。

GET 用于检索数据。 如果 URL 路径以 ID 结尾，则返回由 ID 标识的注册表条目。 如果找不到 ID，但 URL 中有查询，则返回 ID 列表。 例如 http://localhost:57774/csp/msa/person/12A33 返回条目 12A33。
查询键值对是内部用于选择条目的属性匹配子句。
例如，http://localhost:57774/csp/msa/person?name=Hahn%20Istvan&dob=1961 返回出生于 1961 年且姓名为 Hahn Istvan 的人员名单。

DELETE 执行删除。

POST 到 http://localhost:57774/csp/msa/person/init_ 将初始化注册表。

POST 到 http://localhost:57774/csp/msa/person/populate/100_ 将加载 100 条测试条目。

API 文档

下面一节给出了一个如何对服务 API 编制文档的示例。 请记住，结构和内容均未标准化。 这只是一个示例。

我尽量让文档编制变得“ 工具不可知 ”。 市场上已有文档编制工具。 其中一些的表现已几乎达到应有的效果。
本节的目的是让您感受一下，使用文本编辑器来编制 API 文档有多复杂。

资源： 人员

一个用于管理人员类型资源的通用服务。 人员具有最少的一组属性。 基本上是人口统计特征和注册表 ID。

位置： http://localhost:57774/csp/msa/person

方法：

根据资源的唯一 ID 获取单个资源。

动词：GET

参数：

名称	类型	数据类型	注释

1	路径	资源 ID	要检索的资源的唯一 ID。

响应：

状态	返回类型	注释
200	人员	找到记录。
204	无	不存在该资源 ID 的记录。
401	无	未经授权的访问。 该资源需要在标头中提供用户凭据。
403	无	被禁止。 用户无权访问该资源。
500	错误	内部服务器错误。
501	错误	请求的方法未实现。
503	错误	服务暂时不可用。

方法：

根据非唯一查询获取匹配的资源 ID 列表。 该方法使用查询部分来构建查询字符串。
查询键/值对会转换为列名/值对。

动词：GET

参数：

名称	类型	数据类型	注释
name	查询	字符串	搜索条件。
motherMaidenName	查询	字符串	
dob	查询	日期	
birthPlaceCounty	查询	字符串	
birthPlaceCity	查询	字符串	

响应：

状态	返回类型	注释
200	人员	找到记录。
204	无	无匹配记录。
401	无	未经授权的访问。 该资源需要在标头中提供用户凭据。
403	无	被禁止。 用户无权访问该资源。
500	错误	内部服务器错误。
501	错误	请求的方法未实现。
503	错误	服务暂时不可用。

方法：

从注册表中删除条目。

动词：DELETE

参数：

名称	类型	数据类型	注释
1	路径	字符串	唯一注册表 ID。

响应：

状态	返回类型	注释
200	人员	记录已删除。
401	无	未经授权的访问。 该资源需要在标头中提供用户凭据。
403	无	被禁止。 用户无权访问该资源。
500	错误	内部服务器错误。
501	错误	请求的方法未实现。
503	错误	服务暂时不可用。

方法：

向注册表添加或更新条目。

动词：PUT

参数：

名称	类型	数据类型	注释
无	内容	JSON	序列化为 JSON 格式的对象。

响应：

状态	返回类型	注释
200	人员	注册表服务已新增或更新具有生成的资源 ID 的条目。
401	无	未经授权的访问。该资源需要在标头中提供用户凭据。
403	无	被禁止。 用户无权访问该资源。
500	错误	内部服务器错误。
501	错误	请求的方法未实现。
503	错误	服务暂时不可用。

方法：

更新注册表条目的单个属性。

动词：POST

参数：

名称	类型	数据类型	注释
1	路径	字符串	要更新的条目的资源 ID。
name	表单	字符串	属性的新值
motherMaidenName	表单	字符串	
dob	表单	日期	
birthPlaceCounty	表单	字符串	
birthPlaceCity	表单	字符串	

响应：

状态	返回类型	注释
200	人员	记录已更新。
204	无	不存在该资源 ID 的记录。
401	无	未经授权的访问。该资源需要在标头中提供用户凭据。
403	无	被禁止。 用户无权访问该资源。
500	错误	内部服务器错误。
501	错误	请求的方法未实现。
503	错误	服务暂时不可用。

方法：

初始化注册表。

动词：POST

参数：

名称	类型	数据类型	注释
init	路径		

响应：

状态	返回类型	注释
200	无	已初始化。
401	无	未经授权的访问。 该资源需要在标头中提供用户凭据。
403	无	被禁止。 用户无权访问该资源。
500	错误	内部服务器错误。
501	错误	请求的方法未实现。
503	错误	服务暂时不可用。

方法：

填充测试数据。

动词：POST

参数：

名称	类型	数据类型	注释
populate	路径		
2	路径	数值	要填充的条目数。

响应：

状态	返回类型	注释
200	无	已初始化。
401	无	未经授权的访问。 该资源需要在标头中提供用户凭据。
403	无	被禁止。 用户无权访问该资源。
500	错误	内部服务器错误。
501	错误	请求的方法未实现。
503	错误	服务暂时不可用。

数据结构：

人员

名称	类型	标记	注释
ID	注册表 ID	R	生成的注册表 ID。
Name	字符串	R	该人员的母语形式的姓名。
DOB	日期	R	出生日期。
BirthPlace	出生地	O	出生地。
MotherMaidenName	字符串	O	母亲的娘家姓。

出生地

名称	类型	标记	注释
Country	字符串	O	国家/地区代码。
City	字符串	R	城市名称

错误

名称	类型	标记	注释
Code	字符串	R	错误代码
Text	字符串	O	错误文本
InnerError	错误	O	报告组件的子组件报告内部错误。

实现

以下部分给出了我们前面讨论的资源注册表的示例。 这同样只是一个示例。

为了让您更容易理解，我人为地将源码进行了分组。

n 属于 API 的所有内容都放到资源映射类中。

n 完整的 UrlMap XData 块分成单个 **Route** 条目。

n 每个条目都粘附到实际实现功能的静态方法。

所以，要恢复真实的类，需要进行一些（重新）设计。 请愉快地（重新）设计！

第一个服务方法是查询...

```
<

Route Url="/:service" Method="GET" Call="QueryRegistry"/>

classmethod QueryRegistry(service) as %Status {

        try {

                set serviceInstance = ..getServiceInstance(service)

                do
 ..dumpResponse(serviceInstance.runQuery(..getQueryParameters($listbuild("name","dob"
,"motherMaidenName","birthPlaceCountry","birthPlaceCity"))))

        }

        catch ex {

                do ..ReportHttpStatusCode(..getHTTPStatusCode(ex),ex.AsStatus())

        }

        quit $$$OK

}


<

Route Url="/:service/:registryID" Method="GET" Call="GetEntry"/>

classmethod GetEntry(service,registryID) as %Status {

        try {

                set serviceInstance = ..getServiceInstance(service)

                do ..dumpResponse(serviceInstance.get(registryID))

        }

        catch ex {
```

```
                do ..ReportHttpStatusCode(..getHTTPStatusCode(ex),ex.AsStatus())
        }
        quit $$$OK
}

<
Route Url="/:service/:registryID" Method="DELETE" Call="DeleteEntry"/>
classmethod DeleteEntry(service,registryID) as %Status {
        try {
                set serviceInstance = ..getServiceInstance(service)
                do ..dumpResponse(serviceInstance.delete(registryID))
        }
        catch ex {
                do ..ReportHttpStatusCode(..getHTTPStatusCode(ex),ex.AsStatus())
        }
        quit $$$OK
}

<
Route Url="/:service/_init" Method="POST" Call="InitializeRegistry"/>
classmethod InitializeRegistry(service) as %Status {
        try {
                set serviceInstance = ..getServiceInstance(service)
                do ..dumpResponse(serviceInstance.init())
        }
        catch ex {
                do ..ReportHttpStatusCode(..getHTTPStatusCode(ex),ex.AsStatus())
        }
        quit $$$OK
}
```

```
<

Route Url="/:service/_populate/:numberOfRecords" Method="POST" Call="Populate"/>

classmethod Populate(service,numberOfRecords) as %Status {

        try {

                set serviceInstance = ..getServiceInstance(service)

                do ..dumpResponse(serviceInstance.populate(numberOfRecords))

        }

        catch ex {

                do ..ReportHttpStatusCode(..getHTTPStatusCode(ex),ex.AsStatus())

        }

        quit $$$OK

}


<

Route Url="/:service/:registryID" Method="POST" Call="UpdateAttribute"/>

classmethod UpdateAttribute(service,registryID) as %Status {

        try {

                set serviceInstance = ..getServiceInstance(service)

                do
 ..dumpResponse(serviceInstance.updateAttribute(registryID, ..getFormParameters($list
build("name","dob","motherMaidenName","birthPlaceCountry","birthPlaceCity"))))

        }

        catch ex {

                do ..ReportHttpStatusCode(..getHTTPStatusCode(ex),ex.AsStatus())

        }

        quit $$$OK

}


<

Route Url="/:service" Method="PUT" Call="AddOrUpdate"/>
```

```
classmethod AddOrUpdate(service) as %Status {

        try {

                set serviceInstance = ..getServiceInstance(service)

                do
 ..dumpResponse(serviceInstance.addOrUpdate(..getContentParameter()))

        }

        catch ex {

                do ..ReportHttpStatusCode(..getHTTPStatusCode(ex),ex.AsStatus())

        }

        quit $$$OK

}
```

现在是时候分享您的实用方法了。

```
classmethod getServiceInstance(serviceName) as Ens.BusinessService {

        set
 status = ##class(Ens.Director).CreateBusinessService(serviceName, .instance)

        throw:$$$ISERR(status) ##class(NoProduction).%New(status)

        quit instance

}



classmethod getHTTPStatusCode(ex) {

        quit $case(ex.%ClassName(1),

                                ##class(NoProduction).%ClassName(1)                  :5
03,

                                ##class(NotImplemented).%ClassName(1)        :501,


                               :500)

}



classmethod dumpResponse(responseObject) {

        if $isObject(responseObject) {

                if
```

```
 responseObject.%Ex
tends(##class(%DynamicObject).%ClassName(1)) { write responseObject.%ToJSON() }

               elseif
 responseObject.%Extends(##class(%ZEN.proxyObject).%ClassName(1)) {

                       do
 ##class(%ZEN.Auxiliary.jsonProvider).%ObjectToJSON(responseObject)

               }

               elseif responseObject.%Extends(##class(%XML.Adaptor).%ClassName(1)) {

                       do responseObject.XMLExportToString(.ret)

                       write ret

               }

               else { throw ##class(Serialization).%New() }

        }

        else {

               write responseObject

        }

}


classmethod getQueryParameters(parameterList) as %DynamicObject {

        set parameterObject = {}

        for i=1:1:$listlength(parameterList) {

               set parameterName=$listget(parameterList,i)

               set
 $property(parameterObject, parameterName) = %request.Get(parameterName)

        }

        quit parameterObject

}


classmethod getFormParameters(parameterList,queryObject) as %DynamicObject {

        if $data(queryObject) { set parameterObject = queryObject }

        else { set parameterObject = {} }

        for i=1:1:$listlength(parameterList) {
```

```
                set parameterName=$listget(parameterList,i)

                set
 $property(parameterObject, parameterName) = %request.Get(parameterName)

        }

        quit parameterObject

}



classmethod getContentParameter() as %DynamicObject {

        quit {}.%FromJSON(%request.Content)

}
```

到此为止。 我们完成了一个 RESTful Web 服务 API 的设计(?)、实现和文档编制。

敬请关注，我很快会回来进一步解读 Ensemble RESTful Web 服务。 下一个主题是 “ 使用 RESTful Web 服务创建 Ensemble 微服务 ” 。

#REST API #新手 #Ensemble

源 URL:https://cn.community.intersystems.com/post/restful-api-cn