



技术概要: 在 InterSystems 产品中开发 REST 接口

版本 2021.1
2021-07-21

技术概要：在 InterSystems 产品中开发 REST 接口

InterSystems IRIS 数据平台 版本 2021.1 2021-07-21

版权所有 © 2021 InterSystems 公司
保留所有权利。

InterSystems、InterSystems IRIS、InterSystems Caché、InterSystems Ensemble 以及 InterSystems HealthShare 均为 InterSystems 公司的注册商标。

在此使用或涉及到的所有其他品牌或产品名称均为各公司或机构所有的商标或注册商标。

本文件所含商业机密和机密信息，属 InterSystems 公司（马萨诸塞州剑桥纪念大道 1 号，邮编 02142）或其关联公司财产，仅出于 InterSystems 公司产品运营及维护目的而提供。未经 InterSystems 公司事先书面同意，该文件任何部分均不得用于其他目的，亦不可以任何形式、任何方式全部或部分地对该文件进行重制、复制、披露、传输、存储在检索系统中或翻译为任何其他人类或计算机语言。

禁止复制、使用和处置本文件和本文中描述的软件程序，除非在 InterSystems 公司涵盖该等程序和相关文档的标准软件许可协议中所规定的有限范围内。除了标准软件许可协议中规定的声明和保证外，InterSystems 公司对此类软件程序不作任何声明和保证。此外，InterSystems 公司对与使用该等软件程序有关的或因使用该等软件程序而产生的任何损失或损害的责任，按照该等标准软件许可协议所规定的方式加以限制。

以上概括描述了 InterSystems 公司对其计算机软件的使用和责任所施加的限制。完整的信息应参考 InterSystems 公司的标准软件许可协议，该协议的副本将根据要求提供。

InterSystems 公司对本文中可能出现的错误不承担责任，并保留在不另行通知的情况下自行决定对本文中描述的产品和实践进行替换和修改的权利。

有关 InterSystems 产品的技术支持问题，请联系：

InterSystems 全球响应中心 (WRC)

Tel: +1-617-621-0700

Tel: +44 (0) 844 854 2917

Email: support@InterSystems.com

目录

技术概要：在 InterSystems 产品中开发 REST 接口	1
1 为什么提供 REST 接口	1
2 对 InterSystems IRIS 的 REST 调用.....	1
3 如何在 InterSystems IRIS 中定义 REST 接口	2
4 手动编码 REST 接口.....	2
4.1 创建 %CSPREST 的子类并定义 URLMap.....	3
4.2 编码方法.....	3
5 为自己定义 REST 接口	5
5.1 用前须知.....	5
5.2 下载示例应用程序.....	5
5.3 创建一个支持互操作性的命名空间	6
5.4 构建示例代码.....	7
5.5 定义一个 web 应用程序.....	7
5.6 访问 REST 接口.....	8
5.7 记录 REST 接口.....	9
6 有关 InterSystems IRIS 和 REST 的更多信息.....	10

技术概要: 在 InterSystems 产品中 开发 REST 接口

本文档将向您展示如何开发 REST 接口。您可以将这些 REST 接口与 UI 工具（如 Angular）一起使用，以提供对数据库和互操作性产品的访问。您也可以使用它们支持外部系统访问 InterSystems IRIS® 数据平台应用程序。

要浏览所有的技术概要（First Look），包括可以在 [InterSystems IRIS 免费的评估实例](#) 上执行的那些，请参见 [InterSystems First Looks](#)（《InterSystems 技术概要》）。

1 为什么提供 REST 接口

如果您需要从外部系统访问 InterSystems IRIS 数据库中的数据，或者想为这些数据提供用户界面，您可以通过定义 REST 接口来实现。REST——REpresentational State Transfer——是一种使用公开的 URL 从另一个系统检索、添加、更新或删除数据的方法。REST 基于 HTTP，并使用 HTTP 动词 POST、GET、PUT 和 DELETE 映射到数据库应用程序常见的创建、读取、更新和删除（CRUD）功能。您还可以在 REST 中使用其他 HTTP 动词，如 HEAD、PATCH 和 OPTIONS。

REST 是在应用程序之间共享数据的众多方式之一，因此，如果您选择直接使用其他协议，如 TCP、SOAP 或 FTP 进行通信，则您可能并不总是需要设置 REST 服务。但是使用 REST 有以下优点：

- REST 通常只有一个很小的开销（overhead）。它通常使用 JSON，这是一个轻量级的数据包装器。JSON 使用标签标识数据，但标签未在正式模式定义中指定，也没有显式的数据类型。REST 通常比 SOAP 更易于使用，SOAP 使用 XML 且开销（overhead）更大。
- 通过在 REST 中定义一个接口，可以很容易地将客户端和数据库服务器之间的依赖关系降到最低。这使您可以在不影响数据库实现的情况下更新用户界面。您还可以在不影响用户界面或访问 REST API 任何外部应用程序的情况下更新数据库实现（database implementation）。

2 对 InterSystems IRIS 的 REST 调用

在定义 REST 接口之前，您应该了解 REST 调用如何流经 InterSystems IRIS。首先，考虑 REST 调用的部分，如：

```
GET http://localhost:52773/rest/coffeemakerapp/coffeemakers
```

这由以下部分组成：

- GET——这是 http 动词。
- http: ——这指定了通信协议。
- //localhost:52773——这指定了托管 REST 接口的 InterSystems IRIS 实例的服务器和端口号。

- `/rest/coffeemakerapp/coffeemakers`——这是 URL 的主要部分，标识了 REST 调用所指向的资源。在下面的讨论中，URL 指的是 REST 调用的这一部分。

注意： 尽管这个技术概要（First Look）使用安装了 InterSystems IRIS 的 Web 服务器（在本例中是在本地系统的端口 52773 上），但对于部署的任何代码，您都应该使用一个商业 Web 服务器来替换它。安装 InterSystems IRIS 的 web 服务器仅供开发代码时临时使用，并不具备商业 web 服务器的强大功能。

当客户端应用程序进行 REST 调用时：

1. InterSystems IRIS 将它指向与 URL 对应的 web 应用程序。例如，以 `/coffeemakerapp` 开头的 URL 将被发送到处理咖啡机的应用程序，以 `/api/docdb` 开头的 URL 将被发送到处理文档数据模型（Document Data Model）的 web 应用程序。
2. Web 应用程序根据 HTTP 动词（verb）和 URL 中标识 web 应用程序的部分之后的任何部分，将调用指向一个方法。它通过将动词（verb）和 URL 与称为 URLMap 的结构进行比较来实现这一点。
3. 该方法使用 URL 来标识 REST 调用所指定的资源，并根据动词（verb）来执行操作。例如，如果动词（verb）是 GET，该方法返回关于资源的一些信息；如果动词（verb）是 POST，该方法创建一个新的资源实例；如果动词（verb）是 DELETE，该方法删除指定资源。对于 POST 和 PUT 动词（verb），通常有一个提供更多数据的数据包。
4. 该方法执行所请求的操作，并向客户端应用程序返回一个响应信息。

REST 调用需要执行方法和访问数据所需的任何权限。用户或 web 应用程序都可以拥有这些权限。您可以把这些权限分配给角色，然后把角色分配给用户或 web 应用程序。在这个示例中，%All 角色被分配给 web 应用程序。这允许未知用户（未经身份认证）访问 `coffeemakerapp` 数据。如果您没有把这个角色分配给 web 应用程序，未知用户将收到 401 未经授权的错误。您必须在 REST 身份认证调用中指定一个有足够权限的用户。详情请参见 "[Securing REST Services](#)（《[确保 REST 服务](#)》）"。

3 如何在 InterSystems IRIS 中定义 REST 接口

在 InterSystems IRIS 中，有两种方法来定义 REST 接口：

- 定义 OpenAPI 2.0 规范，然后使用 API 管理（API Management）工具来生成 REST 接口的代码。
- 手动编码 REST 接口，然后在管理门户（Management Portal）中定义一个 web 应用程序。

本技术概要（First Look）展示了如何手动编码 REST 接口，包括开发一个调度类（dispatch class）。如果您更喜欢使用规范优先的方法，这些调度类（dispatch class）是自动生成的，不应该被编辑。使用规范优先定义的优点包括能够从规范自动生成文档和客户端代码，但您可以使用任何一种方式来定义 REST 接口。有关使用规范定义 REST 接口的更多信息，请参见 [Creating Rest Services](#)（《[创建 REST 服务](#)》）。

4 手动编码 REST 接口

手动编码 REST 接口包括以下步骤：

- 创建 %CSP.REST 的子类并定义 UrlMap。

- 对处理 REST 调用的方法进行编码。
- 定义 web 应用程序——您通常使用管理门户（Management Portal）来执行此操作。

本技术概要（First Look）使用了一个访问咖啡机数据库的示例应用程序 `coffeemakerapp` 来演示如何手动编码 REST 接口。`Coffeemakerapp` 提供 REST 接口来获取咖啡机的信息、在数据库中创建新的记录、更新现有的记录或删除记录。以下几节通过注释探讨了此示例应用程序的类定义和一些方法，以增强您对代码的理解。您将在完成本技术概要（First Look）的练习部分后，从 GitHub 下载整个应用程序的代码。

4.1 创建 %CSP.REST 的子类并定义 URLMap

这是 `demo.CoffeeMakerRESTServer` 类定义的第一部分。它扩展了 `%CSP.REST` 类。

```
Class Demo.CoffeeMakerRESTServer Extends %CSP.REST
{
Parameter HandleCorsRequest = 1;

XData UrlMap [ XMLNamespace = "http://www.intersystems.com/urlmap" ]
{
  <Routes>
    <Route Url="/test" Method="GET" Call="test"/>
    <Route Url="/coffeemakers" Method="GET" Call="GetAll" />
    <Route Url="/coffeemaker/:id" Method="GET" Call="GetCoffeemakerInfo" />
    <Route Url="/newcoffeemaker" Method="POST" Call="NewMaker" />
    <Route Url="/coffeemaker/:id" Method="PUT" Call="EditMaker" />
    <Route Url="/coffeemaker/:id" Method="DELETE" Call="RemoveCoffeemaker"/>
  </Routes>
}
```

查看路由（Route）要素。每个都有三个属性：

- **Url**——这标识了可由该路由（Route）处理的 REST URL。由于 IRIS 指向以 `/rest/coffeemakerapp` 开头的 URL，因此这个属性指定了紧随其后的 URL 部分。如果 `Url` 属性是 `/coffeemakers`，则此路由（Route）处理以 `/rest/coffeemakerapp/coffeemakers` 开头的 URL。
- **Method（方法）**——这标识了路由（Route）处理的动词。请注意，最后两行的 `Url` 值相同，`/coffeemaker/:id`。有 `PUT` 方法的路由（Route）将处理以 `/rest/coffeemakerapp/coffeemaker/:id` 开头的 URL 的 `PUT` 动词，有 `DELETE` 方法的路由（Route）将处理具有相同开头的 URL 的 `DELETE` 动词。
- **Call（调用）**——指定要调用的方法来处理这个 REST 调用。该方法会被传递完整的 URL 和任何数据，以便它可以根据 URL 做出响应。

`Url` 值中以 `:` 开头的部分表示通配符。也就是说，`/coffeemaker/:id` 将匹配 `/coffeemaker/5`，`/coffeemaker/200`，甚至是 `/coffeemaker/XYZ`。被调用的方法将在参数中获得 `:id` 的值。在这种情况下，它标识了要更新（使用 `PUT`）或删除的咖啡机的 ID。

`Url` 值有额外的选项，允许您将 REST URL 转发到 `%CSP.REST` 子类的另一个实例，但您不需要在这个技术概要（First Look）中处理它。`HandleCorsRequest` 参数指定浏览器是否应该允许跨源资源共享（Cross-origin Resource Sharing, CORS），即在一个域中运行的脚本试图访问在另一个域中运行的 REST 服务，但这也是一个高级主题。

4.2 编码方法

`GetAll` 方法检索有关所有咖啡机的信息。以下是其代码：

```
ClassMethod GetAll() As %Status
{
  Set tArr = []
  Set rs = ##class(%SQL.Statement).%ExecDirect(,"SELECT * FROM demo.coffeemaker")
  While rs.%Next() {
    Do tArr.%Push({
      "img": (rs.%Get("Img")),
      "coffeemakerID": (rs.%Get("CoffeemakerID")),
      "name": (rs.%Get("Name")),
      "brand": (rs.%Get("Brand")),
      "color": (rs.%Get("Color")),
      "numcups": (rs.%Get("NumCups")),
      "price": (rs.%Get("Price"))
    })
  }

  Write tArr.%ToJSON()
  Quit $$$OK
}
```

此方法的注意事项:

- 没有任何参数。每当这个方法被调用时，它就会执行一个 SQL 语句，从 demo.coffeemaker 数据库中选择所有记录。
- 对于数据库中的每条记录，它都会将这些值作为名称、值对附加到数组中。
- 它将数组转换为 JSON，并通过将 JSON 编写到 stdout 来将 JSON 返回给调用的应用程序。
- 最后，它成功退出。

NewMaker() 方法没有参数，但有一个 JSON 有效负荷，用于指定要创建的咖啡机。以下是其代码:

```
ClassMethod NewMaker() As %Status
{
  If '..GetJSONFromRequest(.obj) {
    Set %response.Status = ..#HTTP400BADREQUEST
    Set error = {"errorMessage": "JSON not found"}
    Write error.%ToJSON()
    Quit $$$OK
  }

  If '..ValidateJSON(obj,.error) {
    Set %response.Status = ..#HTTP400BADREQUEST
    Write error.%ToJSON()
    Quit $$$OK
  }

  Set cm = ##class(demo.coffeemaker).%New()
  Do ..CopyToCoffeemakerFromJSON(.cm,obj)

  Set sc = cm.%Save()

  Set result={}
  do result.%Set("Status",$s($$ISERR(sc):$system.Status.GetOneErrorText(sc),1:"OK"))
  write result.%ToJSON()
  Quit sc
}
```

此方法的注意事项:

- 首先，它通过调用类中稍后定义的 GetJSONFromRequest() 和 ValidateJSON() 方法，测试有效负荷是否包含有效的 JSON 对象。
- 然后它使用 JSON 对象创建一个新的 demo.coffeemaker，将记录保存在数据库中。
- 它通过将状态写入 stdout 来返回状态。

最后，RemoveCoffeemaker() 方法展示了 Url 的 :id 部分是如何作为参数传递给该方法的:


```

ClassMethod RemoveCoffeemaker(id As %String) As %Status
{
  Set result={}
  Set sc=0

  if id'="",##class(demo.coffeemaker).%ExistsId(id)
  { Set sc=##class(demo.coffeemaker).%DeleteId(id)
    do result.%Set("Status",$s($$$ISERR(sc):$system.Status.GetOneErrorText(sc),1:"OK"))
  }
  else {
    do result.%Set("Status","")
  }

  write result.%ToJSON()

  quit sc
}

```

总之，由路由调用（RouteCall）属性指定的方法通过以下方式处理REST调用：

- 获得任何参数作为调用参数。
- 通过 obj 值访问有效负荷。
- 通过将响应写入 stdout，将其返回给客户端应用程序。

5 为自己定义 REST 接口

本节逐步向您展示如何使用咖啡机应用程序来处理REST调用。与其让您编写REST接口的代码，不如从GitHub下载完成的应用程序。构建应用程序后，您将定义Web应用程序，然后通过REST调用测试应用程序。

5.1 用前须知

要使用这个程序，您需要在系统上工作，安装一个REST API 应用程序（如Postman、Chrome 高级 REST 客户端（Chrome Advanced REST Client）或cURL），并连接一个正在运行的InterSystems IRIS实例。您对InterSystems IRIS的选择包括多种类型的已授权的和免费的评估实例；该实例不需要由您正在工作的系统托管（尽管它们必须相互具有网络访问权限）。关于如何部署每种类型的实例的信息（如果您还没有可使用的实例），请参见*InterSystems IRIS Basics: Connecting an IDE*（《InterSystems IRIS 基础：连接一个IDE》）中的[Deploying InterSystems IRIS](#)（部署InterSystems IRIS）。有关将REST API 应用程序连接到您的InterSystems IRIS实例所需的信息，请参见同一文档中的[InterSystems IRIS Connection Information](#)（InterSystems IRIS 连接信息）。

5.2 下载示例应用程序

通过克隆或下载已完成的coffeemakerapp应用程序来开始这个练习，该应用程序包括测试对InterSystems IRIS的REST调用所需的所有REST接口。这个技术概要-REST（FirstLook-REST）示例代码可以在以下网站获得：<https://github.com/intersystems/FirstLook-REST>。从GitHub下载的内容必须能从您的InterSystems IRIS实例访问。

下载文件的程序取决于您所使用的实例类型，如下所示：

- 如果您使用的是ICM部署的实例：
 1. 使用带有 **-machine** 和 **-interactive** 选项的 **icm ssh** 命令，在托管实例的节点上打开默认 shell，例如：

```
icm ssh -machine MYIRIS-AM-TEST-0004 -interactive
```

- 在 Linux 命令行上，使用以下命令之一将 repo 克隆到实例的**数据存储卷 (data storage volume)**。例如，对于部署在 Azure 上的配置，数据卷的**默认挂载点 (default mount point)** 是 /dev/sdd，因此您可以使用如下命令：

```
$ git clone https://github.com/intersystems/FirstLook-REST /dev/sdd/FirstLook-REST
OR
$ wget -qO- https://github.com/intersystems/FirstLook-REST/archive/master.tar.gz | tar xvz -C /dev/sdd
```

这些文件现在对容器文件系统上 /irissys/data/FirstLook-REST 中的 InterSystems IRIS 可用。

- 如果您正在使用通过其他方式部署的容器化实例（授权版或社区版（Community Edition））：
 - 在主机上打开 Linux 命令行。（如果您在云节点上使用社区版（Community Edition），请使用 [SSH 连接该节点](#)，如在 *Deploy and Explore InterSystems IRIS*（《*部署和探索 InterSystems IRIS*》）中所述。）
 - 在 Linux 命令行上，使用 **git clone** 或 **wget** 命令，如上所述，将 repo 克隆到容器中挂载为卷的存储位置。
 - 对于社区版（Community Edition）实例，您可以克隆到实例的**持久化 %SYS 目录**（存储指定于实例的配置数据的目录）。在 Linux 文件系统中，这个目录是 /opt/ISC/dur。这使得文件对容器文件系统上 /ISC/dur/FirstLook-REST 中的 InterSystems IRIS 可用。
 - 对于已授权的容器化实例（containerized instance），选择容器中作为卷挂载的任何存储位置（如果使用它，包括持久化 %SYS 目录）。例如，如果您的 **docker run** 命令包含选项 **-v /home/user1:external**，并且您将 repo 克隆到 /home/user1，则文件对容器文件系统上 /external/FirstLook-REST 中的 InterSystems IRIS 可用。
- 如果您使用的是 InterSystems 学习实验室（Learning Labs）实例：
 - 在集成 IDE 中打开命令行终端。
 - 将目录更改为 /home/project/shared 并使用 **git clone** 命令克隆 repo：

```
$ git clone https://github.com/intersystems/FirstLook-REST
```

该文件夹被添加到左边资源管理器（Explorer）面板的 **Shared（共享）** 下，并且该目录对 /home/project/shared 中的 InterSystems IRIS 可用。

- 如果您使用的是已安装的实例：
 - 如果实例的主机是安装了 GitHub 桌面（GitHub Desktop）和 GitHub 大文件存储（GitHub Large File Storage）的 Windows 系统：
 - 在主机的 web 浏览器中进入 <https://github.com/intersystems/FirstLook-REST>。
 - 选择 **Clone or download（克隆或下载）** 然后选择 **Open in Desktop（在桌面上打开）**。

这些文件对您的 GitHub 目录中的 InterSystems IRIS 可用，例如在 C:\Users\User1\Documents\GitHub\FirstLook-REST 中。

- 如果主机是 Linux 系统，只需在 Linux 命令行上使用 **git clone** 命令或 **wget** 命令，就可以将 repo 克隆到您所选择的位置。

5.3 创建一个支持互操作性的命名空间

如果您已经有了一个互操作性命名空间，就可以使用它。否则，创建一个命名空间：

- 使用 *InterSystems IRIS Basics: Connecting an IDE*（《*InterSystems IRIS 基础：连接一个 IDE*》）中**为您的实例描述的 URL**，在浏览器中打开您的实例的管理门户（Management Portal）。

- 通过导航到命名空间 (Namespaces) 页面(**System Administration (系统管理)** > **Configuration (配置)** > **System Configuration (系统配置)** > **Namespaces (命名空间)**)，并按照 *System Administration Guide* (《系统管理指南》) 的“Configuring InterSystems IRIS (《配置 InterSystems IRIS》)”一章中[创建/修改命名空间](#)中的使用新命名空间 (New Namespace) 页面的说明，点击 **Create New Namespace (创建新命名空间)** 按钮，创建一个支持互操作性的命名空间。

5.4 构建示例代码

要构建示例代码，请按照以下步骤操作

- 使用 *InterSystems IRIS Basics: Connecting an IDE* (《*InterSystems IRIS 基础: 连接一个IDE*》) 中[为您的实例描述的程序](#)，打开 InterSystems 终端 (Terminal)。
- 将命名空间设置为您将使用的互操作性命名空间。输入以下命令，替换 <Namespace-name> 与您所使用的命名空间。

```
set $namespace = "<namespace-name>"
```

- 输入以下命令，替换 <path> 与您下载示例的路径。该目录包含许可证 (License) 和 Readme 文件以及 buildsample 目录：

```
do $system.OBJ.Load("<path>/buildsample/Build.RESTSample.cls", "ck")
do ##class (Build.RESTSample) .Build()
```

- 当出现提示时，输入下载此示例的目录的完整路径。然后，该方法加载和编译代码，并执行其他需要的设置步骤。

注意： FirstLook-REST/README.md 文件包含这些说明的一个版本。

5.5 定义一个 web 应用程序

现在您已经使用 REST 接口构建了示例应用程序，您需要定义一个 web 应用程序：

- 使用 *InterSystems IRIS Basics: Connecting an IDE* (《*InterSystems IRIS 基础: 连接一个IDE*》) 中[为您的实例描述的 URL](#)，在浏览器中打开您的实例的管理门户 (Management Portal)。
- 选择 **System Administration (系统管理)** > **Security (安全)** > **Applications (应用程序)** > **Web Applications (web 应用程序)**。
- 选择 **Create New Web Application (创建新的 web 应用程序)** 并输入以下设置
 - Name (名称)** : /Rest/coffeemakerapp——它指定了将由这个 web 应用程序处理的 URL。InterSystems IRIS 会将所有以 /rest/coffeemakerapp 开头的 URL 定向到这个 web 应用程序。
 - Namespace (命名空间)** : 您创建的支持互操作性的名称空间的名称。
 - Enable (支持)** : 选择 REST。
 - Dispatch Class (调度类)** : Demo.CoffeeMakerRESTServer——这是定义了 URLMap 的类。
 - Security Settings/Allowed Authentication Methods (安全设置/允许的身份认证方法)** : 同时选择 **Unauthenticated (未经身份认证)** 和 **Password (密码)** 复选框。
- 选择 **Save (保存)**。
- 要允许此示例的未经身份认证的访问，必须赋予 web 应用程序 %All 角色。要执行此操作：
 - 选择 **Save (保存)** 后，选择 **Application Roles (应用程序角色)** 标签。
 - 从 **Available (可用的)** 角色中选择 %All 角色。
 - 点击右箭头 (选择) 按钮，将 %All 角色移动到 **Selected (选定的)** 角色。

- d. 点击 **Assign (分配)** 按钮。
- e. %All 角色现在被列为 **Application Role (应用程序角色)**。这确保了来自未经身份认证用户的 REST 调用将拥有访问 `coffeemakerapp` 数据所需的权限。如果没有这个角色，REST 调用将必须为拥有足够权限的用户指定身份认证。

5.6 访问 REST 接口

CoffeeMaker REST 应用程序现在可以工作了。您将输入 REST 命令来访问咖啡机数据库。在您的 REST API 工具（如 Postman）中，按照以下步骤操作：

1. 指定 REST POST 请求来添加新的咖啡机，必要时使用您的 [InterSystems IRIS 实例的指定信息](#)

- HTTP 操作：POST
- URL: `http://server:port/rest/coffeemakerapp/newcoffeemaker`，其中 *服务器* 和 *端口* 是您的实例的主机标识符和 web 服务器端口。
- 您的实例的登录凭证。
- 输入数据：

```
{"img":"img/coffee3.png","coffeemakerID":"99","name":"Double Dip","brand":"Coffee+","color":"Blue","numcups":2,"price":71.73}
```

尽管数据包含一个 `coffeemakerID` 的值，但这是一个计算字段，并且在添加记录时会分配一个新值。该调用返回成功状态：

```
{"Status":"OK"}
```

2. 重复上一步骤两次，添加以下两个咖啡机：

```
{"img":"img/coffee4.png","coffeemakerID":"99","name":"French Press","brand":"Coffee For You","color":"Blue","numcups":4,"price":50.00}
```

```
{"img":"img/coffee9.png","coffeemakerID":"99","name":"XPress","brand":"Shiny Appliances","color":"Green","numcups":1,"price":95.00}
```

3. 使用相同的实例指定信息，指定一个 REST GET 请求，以获得数据库中的咖啡机列表：

- HTTP 操作：GET
- URL: `http://server:port/rest/coffeemakerapp/coffeemakers`
- 您的实例的登录凭证。

该调用返回一个咖啡机列表，如：

```
[{"img":"img/coffee3.png","coffeemakerID":"1","name":"Double Dip","brand":"Coffee+","color":"Blue","numcups":2,"price":71.73}, {"img":"img/coffee4.png","coffeemakerID":"2","name":"French Press","brand":"Coffee For You","color":"Blue","numcups":4,"price":50}, {"img":"img/coffee9.png","coffeemakerID":"3","name":"XPress","brand":"shiny Appliances","color":"Green","numcups":1,"price":95}]
```

4. 指定以下 REST 调用，删除 ID=2 的咖啡机：

- HTTP 操作：DELETE
- URL: `http://server:port/rest/coffeemakerapp/coffeemaker/2`
- 您的实例的登录凭证。

该调用返回成功状态：

```
{"Status": "OK"}
```

5. 重复 REST GET 请求。该调用返回一个咖啡机列表，如：

```
[{"img": "img/coffee3.png", "coffeemakerID": "1", "name": "Double Dip", "brand": "Coffee+",  
"color": "Blue", "numcups": 2, "price": 71.73},  
{"img": "img/coffee9.png", "coffeemakerID": "3", "name": "XPress", "brand": "Shiny Appliances",  
"color": "Green", "numcups": 1, "price": 95}]
```

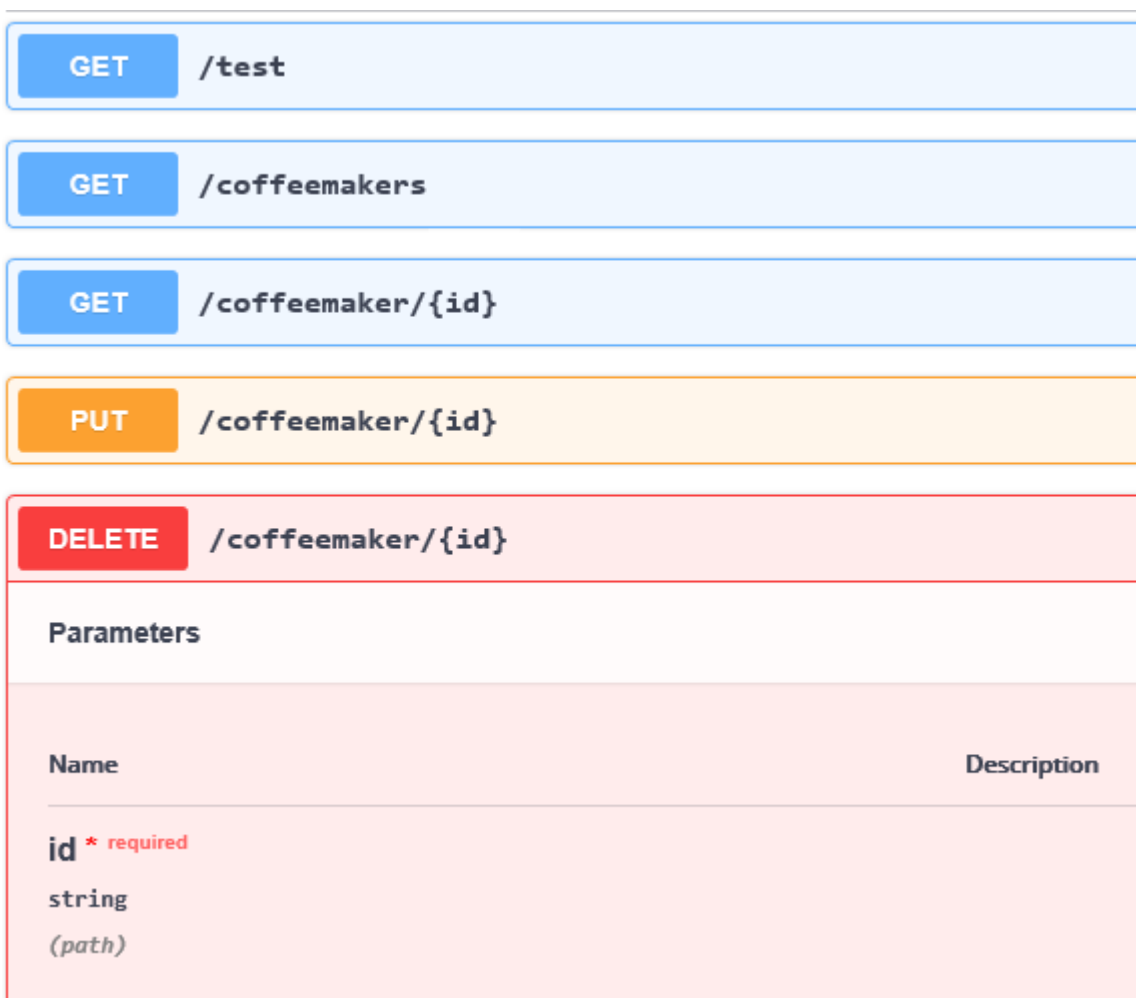
5.7 记录 REST 接口

当您向开发者提供 REST 接口时，您应该提供文档，以便他们知道如何调用接口。您可以使用 [OpenAPI Spec](#) 来记录 REST 接口，并使用工具，如 [Swagger](#) 来编辑和格式化文档。[InterSystems](#) 正在开发一项功能来支持这个文档。这个版本包含了 API 管理（API Management）中的一个功能，它可以为您的 REST API 生成文档框架。您仍然需要编辑生成的文档，以添加注释和额外的信息，例如参数和 HTTP 返回值的内容。

要为 CoffeeMakerApp REST 示例生成文档，请输入以下 REST 调用，使用您的 [InterSystems IRIS 实例的指定信息](#) 和您创建的命名空间的名称：

- HTTP 操作：GET
- URL: `http://server:port/api/mgmt/v1/namespace/spec/rest/coffeemakerapp/`
- 您的 InterSystems IRIS 实例的登录凭证。

您可以把这个调用的输出粘贴到 `swagger` 编辑器中。它将 JSON 转换为 YAML（另一种标记语言（Yet Another Markup Language））并显示文档。您可以使用 `swagger` 编辑器向文档中添加更多信息。`Swagger` 编辑器显示的文档如下所示：



The image displays a list of REST API endpoints for a coffee maker application. Each endpoint is shown in a colored box with its HTTP method and path. The endpoints are: GET /test, GET /coffeemakers, GET /coffeemaker/{id}, PUT /coffeemaker/{id}, and DELETE /coffeemaker/{id}. Below the DELETE endpoint, there is a 'Parameters' section with a table listing the 'id' parameter as a required string path parameter.

Name	Description
id * required	
string	
(path)	

6 有关 InterSystems IRIS 和 REST 的更多信息

有关在 InterSystems IRIS 中创建 REST 服务的更多信息，请参见以下内容：

- [Setting Up RESTful Services](#)（设置 RESTful 服务）是 InterSystems 的在线课程，它使用与本文档相同的咖啡机应用程序，但更详细。您需要登录 learning.intersystems.com 才能参加这个课程。如果您没有账户，可以创建一个。
- [Creating REST Services](#)（《创建 REST 服务》）
- [Using REST Services and Operations in Productions](#)（《在产品中使用 REST 服务和操作》）