

文章

[Stefan Wittmann](#) · 一月 4, 2021 阅读大约需 4 分钟

从 SQL 生成 JSON

最近我一直在发布[对 JSON 功能的一些更新](#)，我很高兴许多人提供了反馈。今天，我想将重点放在另一个方面：使用 SQL 查询生成 JSON。

显然，SQL 是从关系模型中检索记录的重要概念。假设你要查询数据并将其以简单 JSON 结构的形式公开给 REST 服务，通常，你必须查询数据，然后对结果集进行遍历，最后为每条记录构造一个 JSON 结构。你需要编写自定义代码。

我们添加了标准 SQL 函数 `JSONOBJECT` 和 `JSONARRAY`，这样可以更容易地直接从 SQL 查询生成 JSON，而无需编写代码。这两个函数是 Caché 2016.2 开始新增的。

以下表 `Baking.Pastry` 为例：

行	类型	说明
1	Choux Pastry	A light pastry that is often filled with cream
2	Puff Pastry	Many layers that cause the pastry to expand or puff when baked
3	Filo Pastry	Multiple layers of a paper-thin dough wrapped around a filling

JSONOBJECT

`JSONOBJECT` 是一个接受多个键值对并生成 JSON 对象的函数。

```
SELECT JSON_OBJECT('pastry_type' : Type, 'pastry_description' : Description) AS pastryJSON FROM Baking.Pastry
```

```
Row    pastryJSON
```

```
----
```

```
1      {"pastry_type" : "Choux Pastry", "pastry_description" : "A light pastry that is often filled with cream"}
2      {"pastry_type" : "Puff Pastry", "pastry_description" : "Many layers that cause the pastry to expand or puff when baked"}
3      {"pastry_type" : "Filo Pastry", "pastry_description" : "Multiple layers of a paper-thin dough wrapped around a filling"}
```

在此例中，`JSONOBJECT` 函数为每条记录生成一个 JSON 对象。每个对象都包含两个属性 `pastrytype` 和 `pastrydescription`，因为我们为函数提供了两个参数。每个参数由两部分组成，用冒号分隔：

1. 应注入到 JSON 对象中的键的名称
2. 与该键关联的字段值

此示例设置了静态键，因为我只提供了字符串文字，例如 `'pastrytype'`。对于字段值（我指的是列），例如

Type, 无论该列的内容是什么, 都将设置为关联键的值。这是构造 JSON 对象的常见用例, 但是通过为键传递列, 还可以动态创建键。

JSONARRAY

JSONARRAY 的工作方式类似。它构造一个 JSON 数组, 每传递一个参数都会将相应的值推送到数组中。

```
SELECT JSON_ARRAY(Type, Description) AS pastryJSON FROM Baking.Pastry
```

```
Row  pastryJSON
```

```
-----
```

```
1  ["Choux Pastry" , "A light pastry that is often filled with cream"]
2  ["Puff Pastry" , "Many layers that cause the pastry to expand or puff when baked"]
3  ["Filo Pastry" , "Multiple layers of a paper-thin dough wrapped around a filling"]
```

JSONARRAY 是一个非常简单的函数。此示例创建一个数组, 其中每行包含两个元素。第一项由列 Type 的值填充, 而第二项由列 Description 的值填充。

高级方案

也许你需要创建一个更复杂的 JSON 结构。值参数可以是嵌套的 JSONARRAY 或 JSONOBJECT 函数调用, 从而可以构造嵌套的 JSON 结构。我们回到第一个示例, 将 JSON 对象包装在标头结构中:

```
SELECT JSON_OBJECT('food_type' : 'pastry', 'details' : JSON_OBJECT('type' : Type, 'description' : Description)) AS pastryJSON FROM Baking.Pastry
```

```
Row  pastryJSON
```

```
-----
```

```
1  {"food_type" : "pastry", "details" : {"type" : "Choux Pastry", "description" : "A light pastry that is often filled with cream"}}
2  {"food_type" : "pastry", "details" : {"type" : "Puff Pastry", "description" : "Many layers that cause the pastry to expand or puff when baked"}}
3  {"food_type" : "pastry", "details" : {"type" : "Filo Pastry", "description" : "Multiple layers of a paper-thin dough wrapped around a filling"}}
```

我们计划在将来的版本中实现更多的 JSON SQL 函数, 但这两个函数是坚实的开始。主要用例是从关系型数据构造简单的 JSON 元素, 而无需编写代码。这样即使无法更改后端代码, 也可以从系统直接发布 JSON。

要创建更复杂的结构, 使用新的复合接口可以更高效地构建, 该接口允许将 Persistent/Registered 对象转换为动态对象/数组。然后可以根据需要轻松修改结构, 最后通过 \$toJSON() 调用将其导出为 JSON 字符串。

从 SQL 生成 JSON

Published on InterSystems Developer Community (<https://community.intersystems.com>)

我会在将来的帖子中更详细地介绍这个主题。

[#JSON](#) [#SQL](#)

源 URL:<https://cn.community.intersystems.com/post/%E4%BB%8E-sql-%E7%94%9F%E6%88%90-json>