> 你不愿意种花，你说不愿意看它慢慢凋零的样子，没错你避免了结束，同时也避免了所有的开始

FlinkSql之入门概述

> Flink本身是批流统一的处理框架，所以Table API和SQL，就是批流统一的上层处理API。
>
> **目前功能尚未完善，处于活跃的开发阶段。**Table API是一套内嵌在Java和Scala语言中的查询API，它允许我们以非常直观的方式，组合来自一些关系运算符的查询（比如select、filter和join）。而对于Flink SQL，就是直接可以在代码中写SQL，来实现一些查询（Query）操作。Flink的SQL支持，基于实现了SQL标准的Apache Calcite（Apache开源SQL解析工具）。

无论输入是批输入还是流式输入，在这两套API中，指定的查询都具有相同的语义，得到相同的结果。

依赖引入

```
<dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-table-planner_2.11</artifactId>
    <version>1.10.0</version>
</dependency>
<dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-table-api-scala-bridge_2.11</artifactId>
    <version>1.10.0</version>
</dependency>
```

- flink-table-planner：planner计划器

  > 是table API最主要的部分，提供了运行时环境和生成程序执行计划的planner；

- flink-table-api-scala-bridge：bridge桥接器

  > 主要负责table API和 DataStream/DataSet API的连接支持，按照语言分java和scala。

这里的两个依赖，是IDE环境下运行需要添加的；**如果是生产环境，lib目录下默认已经有了planner，就只需要有bridge就可以了。**当然，如果想使用用户自定义函数，或是跟kafka做连接，需要有一个SQ+client，这个包含在flink-table-common里。

两种planner（old & blink）的区别

- 批流统一：**Blink将批处理作业，视为流式处理的特殊情况。**所以，**blink不支持表和DataSet之间的转换**，批处理作业将不转换为DataSet应用程序，而是跟流处理一样，转换为DataStream程序来处理。
- 因为批流统一，Blink planner也不支持BatchTableSource，而使用有界的StreamTableSource代替。
- Blink planner只支持全新的目录，不支持已弃用的ExternalCatalog。
- 旧planner和Blink planner的FilterableTableSource实现不兼容。旧的planner会把PlannerExpressions下推到filterableTableSource中，而blink planner则会把Expressions下推。
- 基于字符串的键值配置选项仅适用于Blink planner。
- PlannerConfig在两个planner中的实现不同。
- Blink planner会将多个sink优化在一个DAG中（仅在TableEnvironment上受支持，而在StreamTableEnvironment上不受支持）。而旧planner的优化总是将每一个sink放在一个新的DAG中，其中所有DAG彼此独立。

- **旧的planner不支持目录统计，而Blink planner支持。**

FlinkSql表环境

基本程序结构

> Table API 和 SQ+的程序结构，与流式处理的程序结构类似；也可以近似地认为有这么几步：首先创建执行环境，然后定义source、transform和sink。

具体操作流程如下：

```
va+tableEnv = ...      // 创建表的执行环境

// 创建一张表，用于读取数据
tableEnv.connect(...).createTemporaryTable("inputTable")
// 注册一张表，用于把计算结果输出
tableEnv.connect(...).createTemporaryTable("outputTable")

// 通过 Table API 查询算子，得到一张结果表
// 注册的表转换为Table类型
va+result = tableEnv.from("inputTable").select(...)
// 通过 SQL查询语句，得到一张结果表
va+sqlResult  = tableEnv.sqlQuery("SELECT ... FROM inputTable ...")

// 将结果表写入输出表中
result.insertInto("outputTable")
```

创建表环境

> 创建表环境最简单的方式，就是基于流处理执行环境，调create方法直接创建【有问题都可以私聊我WX：focusbigdata，或者关注我的公众号：FocusBigData，注意大小写】

```
va+tableEnv = StreamTableEnvironment.create(env)
```

表环境（TableEnvironment）是flink中集成Table API & SQL的核心概念。它负责:

- 注册catalog
- 在内部 catalog 中注册表
- 执行 SQ+查询
- 注册用户自定义函数
- 将 DataStream 或 DataSet 转换为表
- 保存对 ExecutionEnvironment 或 StreamExecutionEnvironment 的引用

> 在创建TableEnv的时候，**可以多传入一个EnvironmentSettings或者TableConfig参数**，可以用来配置 TableEnvironment的一些特性。

基于老版本的流处理环境

```
val settings = EnvironmentSettings.newInstance()
  .useOldPlanner()      // 使用老版本planner
  .inStreamingMode()    // 流处理模式
  .build()
val tableEnv = StreamTableEnvironment.create(env, settings)
```

基于老版本的批处理环境

```
val batchEnv = ExecutionEnvironment.getExecutionEnvironment
val batchTableEnv = BatchTableEnvironment.create(batchEnv)
```

基于blink版本的流处理环境

```
val bsSettings = EnvironmentSettings.newInstance()
.useBlinkPlanner()
.inStreamingMode().build()
val bsTableEnv = StreamTableEnvironment.create(env, bsSettings)
```

基于blink版本的批处理环境

```
val bbSettings = EnvironmentSettings.newInstance()
.useBlinkPlanner()
.inBatchMode().build()
val bbTableEnv = TableEnvironment.create(bbSettings)
```