

Freewind @ Thoughtworks [scala](#) [java](#) [javascript](#) [dart](#) [工具](#) [编程实践](#) [月结](#) [math](#) [python](#) [english](#) [\[comments\]](#) [\[admin\]](#) [\[feed\]](#)

(2014-03-18) 易用强大的解析器PetitParser – (1) 安装及Hello World

广告: [云梯: 翻墙vpn \(省10元\)](#) [土行孙: 科研用户翻墙http proxy \(有优惠\)](#)

[PetitParser](#)是一个强大、易用、灵活的解析工具，使用Dart写成。

它文档丰富，代码易懂，作者也非常热情。在学习的过程中，我曾经有过很多疑问，都得到了作者及时详尽的解答，在这里要特别表示感谢。如果大家对解析器的实现感兴趣，或者想自己写一些复杂的解析规则，建议把它的代码看懂再动手。

我花了一些学习了它的使用，并且研读了一下它的源代码，感觉收获很大。最大的收获就是知道了：解析器这么高大上的东西，竟然是用一些非常简单和基本的功能实现的，这真是太神奇了。

在这里先讲一下如何使用它写出最简单的Hello world.

安装

我们需要先到Dart的官网安装SDK: <http://dartlang.org>

Dart支持windows/linux/mac，并且默认包里内置了基本eclipse的DartEditor，如果不想用的话，可以到[Tools 页面](#)下载独立版本。

将Dart解压到本地后，将dart-sdk下的bin目录加入path，然后运行dart命令：

```
→ bin dart --version
Dart VM version: 1.2.0 (Tue Feb 25 07:31:27 2014) on "macos_x64"
```

应该能看到类似上面的输出。

在Dart里还内置了一个`pub`命令，用来管理和下载依赖包，我们一会儿要用它。

下载依赖

创建一个test目录，在里面创建一个dartpub.yaml文件，内容如下：

```
name: SharkDart
version: 0.1.0
dependencies:
  petitparser:
    git: https://github.com/renggli/PetitParserDart.git
```

如果执行：

```
pub install
```

如果一切成功，你将会看到项目目录下多了一个`packages`目录，里面有下载好的各依赖库，如petitparser。

(注意：该命令会访问国外的一些地址，由于某些众所周知的原因，有时可能无法成功运行，自备梯子)

```
## hello world
```

我们将使用petitparser实现一个很简单的功能：找到一个字符串里所有的数字，并把它们相加。

比如对于字符串：`aa12bb23cc5`，它应该能从里面取到数字`12`和`23`和`5`，可计算出结果`40`

创建文件test.dart，代码如下：

```
import "package:petitparser/petitparser.dart";

main() {
  var str = 'aa12bb23cc5';
  var parser = (digit().plus().flatten() | any().map((_) => null)).star().map((each) {
    print(each);
    var total = 0;
    for (String numStr in each) {
      if (numStr != null) {
        total += int.parse(numStr);
      }
    }
    return total;
  });
  var total = parser.parse(str);
  print(total);
}
```

运行它：

```
$ dart test.dart
```

得到结果：

```
[null, null, 12, null, null, 23, null, null, 5]
```

```
Success[1:12]: 40
```

可以看到它的确成功解析出了所有数字，并得到了正确的和。

匹配语法

下面这一行代码是关键：

```
(digit().plus().flatten() | any().map((_) => null)).star()
```

里面各函数都是在PetitParser里定义好的

- `digit()` 解析一个数字，相当于正则中的 `\d`
- `any()` 解析任意一个字符，相当于正则中的 `.`（包括换行符）
- `plus()` 重复一次或多次，相当于正则中的 `+`
- `star()` 重复零次或多次，相当于正则中的 `*`
- `flatten()` 将匹配到的结果变成一个字符串，否则是一个字符数组
- `map()` 用来对匹配到的内容进行变换

这行代码的意思就是：先匹配多个连在一起的数字，如果匹配不到，则匹配一个任意字符并返回`null`代替。然后不断重复这个过程，直到解析完字符串的全部内容。

计算

后面那个大的`map`块，是用来拿出匹配的数字并计算和：

```
.map((each) {
```

```
    var total = 0;
    for (String numStr in each) {
        if (numStr != null) {
            total += int.parse(numStr);
        }
    }
    return total;
});
```

其中`each`是前面全部匹配到的内容，由于前面是一个`star()`，所以它是一个数组，并且里面的内容将会由 `null` 和数字组成。然后取出里面的数字并计算结果返回。

parse

下面这句代码执行解析过程:

```
parser.parse(str);
```

它的返回结果可为**success**和**failure**。如果匹配成功,则会显示最后的结果,失败的话,会提示在哪一步没有匹配成功。

0条评论

Freewind's blog

1 登录 ▾

♥ 推荐

🐦 推文

f 分享

最早发布 ▾



开始讨论...

通过以下方式登录

或注册一个 DISQUS 帐号 (?)

姓名

来做第一个留言的人吧!

在 FREEWIND'S BLOG 上还有

(2014-09-12) sbt预定义的keys

1条评论 • 5年前



花米 — good documetation

(2015-05-04) 怎么显示一个maven项目中的第三方依赖

1条评论 • 4年前



uniquejava — nearest definition是指离树根最近的位置吧。
jcl-over-slf4j:jar:1.7.7依赖于slf4j-api:jar:1.7.7, 而maven

(2011-09-13) Ecere 首页设计及思考

2条评论 • 5年前



Freewind — It's really a pleasantly surprised to see your comment here, especially this blog is written in Chinese :)Time flies, I didn't realised it was already 3 years since

(2014-03-23) 正向代理与反向代理

1条评论 • 5年前



jiukun zhang — 举例通俗易懂, Techie之典范!

📧 订阅 🛡️ 在您的网站上使用 Disqus添加 Disqus添加 🔒 Disqus 隐私政策隐私政策隐私