Router

以 backbone.js vue.js 為藍本

```
var Workspace = Backbone.Router.extend({

  routes: {
    "help":                 "help",    // #help
    "search/:query":        "search",  // #search/kiwis
    "search/:query/p:page": "search"   // #search/kiwis/p7
  },

  help: function() {
    ...
  },

  search: function(query, page) {
    ...
  }

});
```

The routes hash maps URLs with parameters to
functions on your router
(or just direct function definitions,
if you prefer), similar to the View's events hash.
Routes can contain parameter parts,
:param, which match a single URL component between slashes;
and splat parts *splat,
which can match any number of URL components.

Part of a route can be made optional by surrounding
it in parentheses **(/:optional)**.
For example, a route of "search/:query/p:page"
will match a fragment of **#search/obama/p2**,
passing "obama" and "2" to the action as positional arguments.

A route of "**file/*path**" will match **#file/folder/file.txt**,
passing "folder/file.txt" to the action.

A route of "**docs/:section(/:subsection)**" will match #docs/faq
and **#docs/faq/installing**,
passing "faq" to the action in the first case,
and passing "faq" and "installing" to the action in the second.

A nested optional route of "**docs(/:section)(/:subsection)**"
will match **#docs**,
#docs/faq, and **#docs/faq/installing**,
passing "faq" to the action in the second case,
and passing "faq" and "installing" to the action in the third.

Trailing slashes are treated as part of the URL,
and (correctly) treated as a unique route when accessed.
docs and docs/ will fire different callbacks.
If you can't avoid generating both types of URLs,
you can define a "docs**(/)**" matcher to capture both cases.

模式　　匹配路徑　　$route.params

/user/:username => /user/evan
{ username: 'evan' }

/user/:username/post/:post_id => /user/evan/post/123
{ username: 'evan', post_id: 123 }

2 版規則

用 **/** 作為階層分割

命令符號有 **(){}*/**
命令符號可加 \ 做脫逸

(...) 視為 (?:...)?

a / b / file(.text)
可匹配
(a / b / file)
(a / b / file.text)

(...) 裡面不能有 **/** 分隔符號，會讓情況變複雜
(...) 裡面不能有 **()** 套嵌
(…) 可以有 {…} {*...} 變數符號

{…} 視為變數
{...} 裡面不能有 **() {} /**

a / b/ {a}_{b}
可匹配
(a / b / page_5)
(a / b / book_ggyy)

{}{} 不可連在一起，必須要有字元作區隔，如 {}-{}
{...} 視為 ([^/]*) 或 ([^/]*?)
{…} 後面有字元視為 ([^/]*?)
{…} 後面無字元視為 ([^/]*)

{*…}

{*...} 裡面不能有 **(){}/**

a / b / {*…}

可匹配

a / b / c / d

a / b / cgcg

{*…} 必須放在最後

注意事項

() 裡面若有 / 不好處理的地方
意義不大，會讓情況更複雜

a / b (/)

取代方案

a / b

a / b /

a / b (cd /)

取代

a / bcd /

a / b

a/ b (/ c)

取代

a / b / c

a / b

a / b (/ c / d)

取代

a / b / c / d

a / b

a / b (/ c)(/ d)

讓情況變複雜，不容易判別

取代

a / b / c / d

a / b / c

a / b / d

a / b

a / b (d / c)

沒太大作用

取代

a / b

a / bd / c

{}{} 會讓狀況無法判定

a / b / **{x}{y}**

轉換為正則為 /^a\/b\/**([^/]*?)([^/]*)**$/

a/b/aasssss => {x} 將無法捕獲 {y} = aasssss

a / b/ {x}({y}age) 也不行

{} {} 之間必須至少要有一個判定字元

a / b / **{x}_{y}**

轉換為正則為 /^a\/b\/**([^/]*?)_([^/]*)**$/

a/b/aas__ssss => {x} = aas， {y} = _ssss

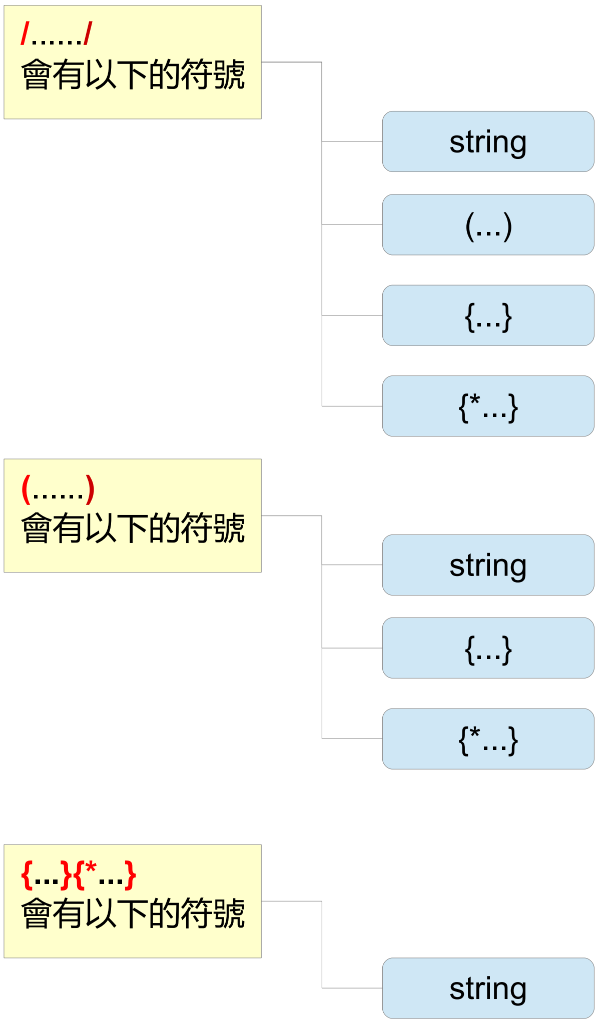
~~{*...} 必須擺在最後~~

a / b / **{*...}** / **{x}**

轉換為正則為 /^a\/b\/**(.*)**\/**([^/]*)**$/

a / b / c / d / e

規則演化



```
// 取得  keyList  的方法
// (  ()) 這情況不要發生，會橫生更多難度
// 也沒必要

let reg = /^([^/]*?)\/(?:_([^/]*?))?\/([^/]*?)$/;
let strList = ["a/_b/c", "a//c"];



strList.forEach(function(str){
    let res = reg.exec(str);

    console.dir(res);
    console.log("--------------");
});
```

router 規則

流程歸納

BlockNode

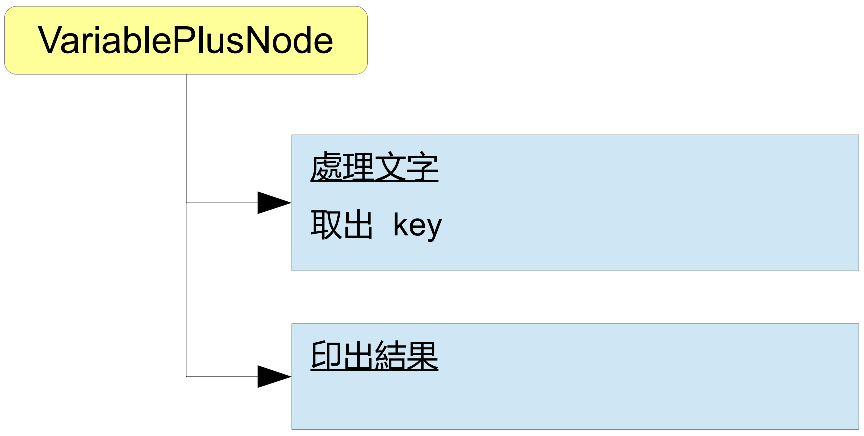
- 處理文字
  將文字轉變成 nodelList
- 攤平 nodeList(攤平 ScratchNode)
  檢規則查
  取得使用者設定的 key
- 印出結果

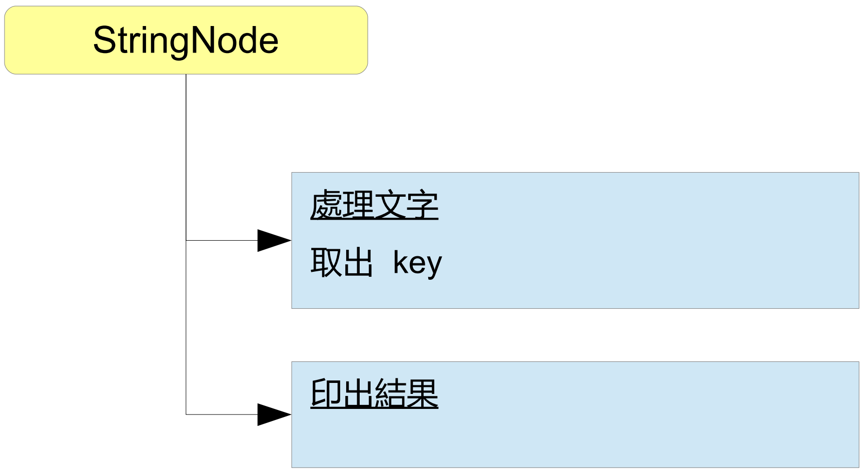
ScratchNode

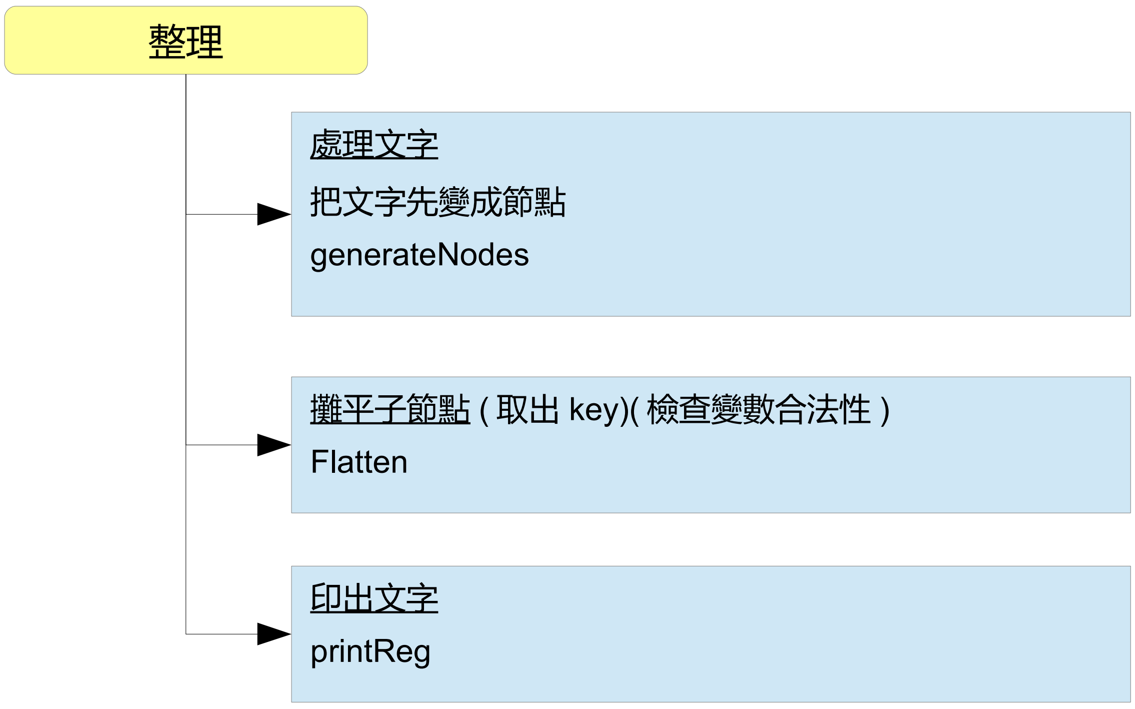
- 處理文字
  將文字轉變成 nodelList
- 印出結果

VariableNode

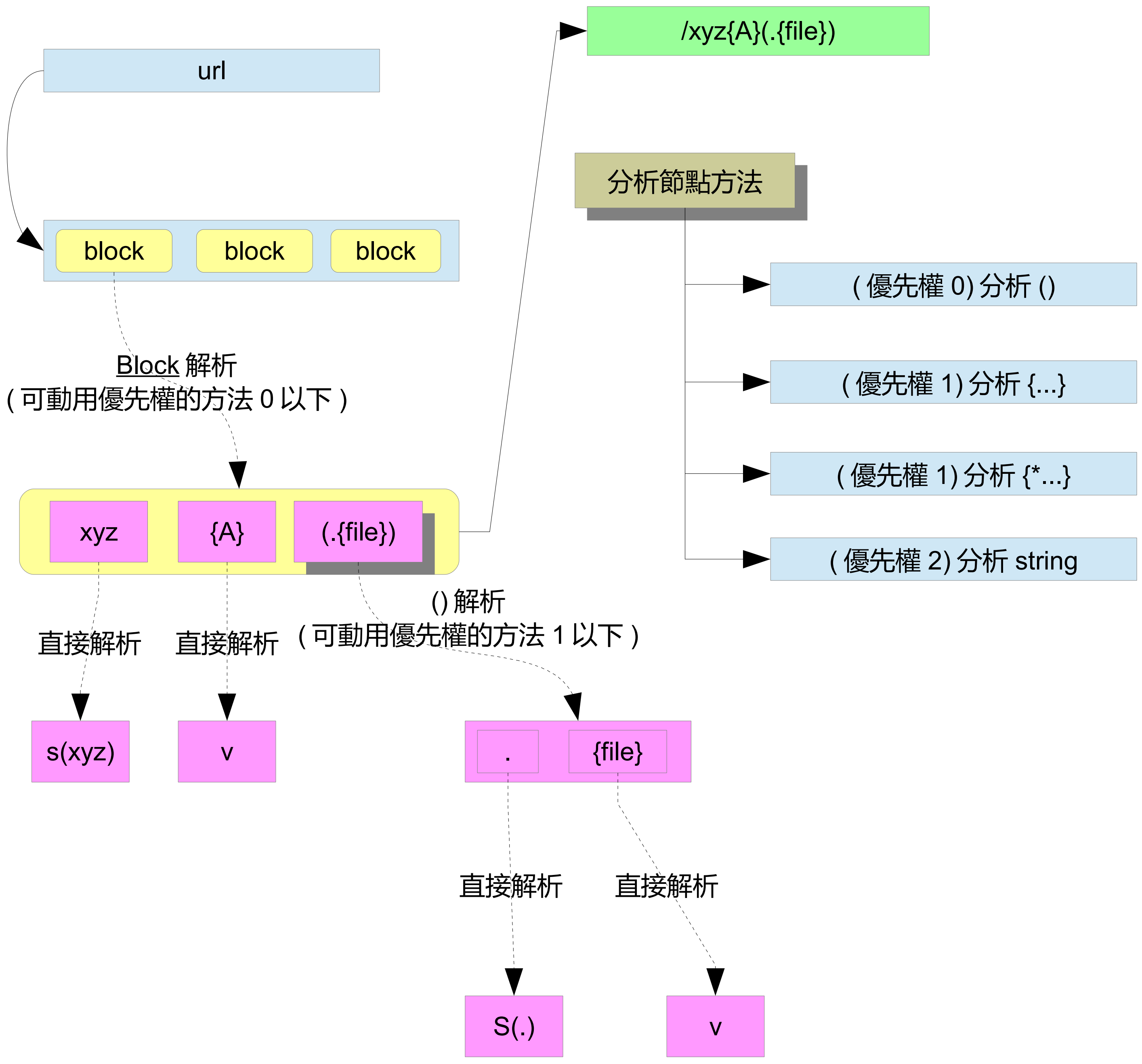
- 處理文字
  取出 key
- 印出結果


VariablePlusNode
處理文字
取出 key
印出結果


StringNode
處理文字
取出 key
印出結果


整理
處理文字
把文字先變成節點
generateNodes
攤平子節點 (取出 key)(檢查變數合法性)
Flatten
印出文字
printReg

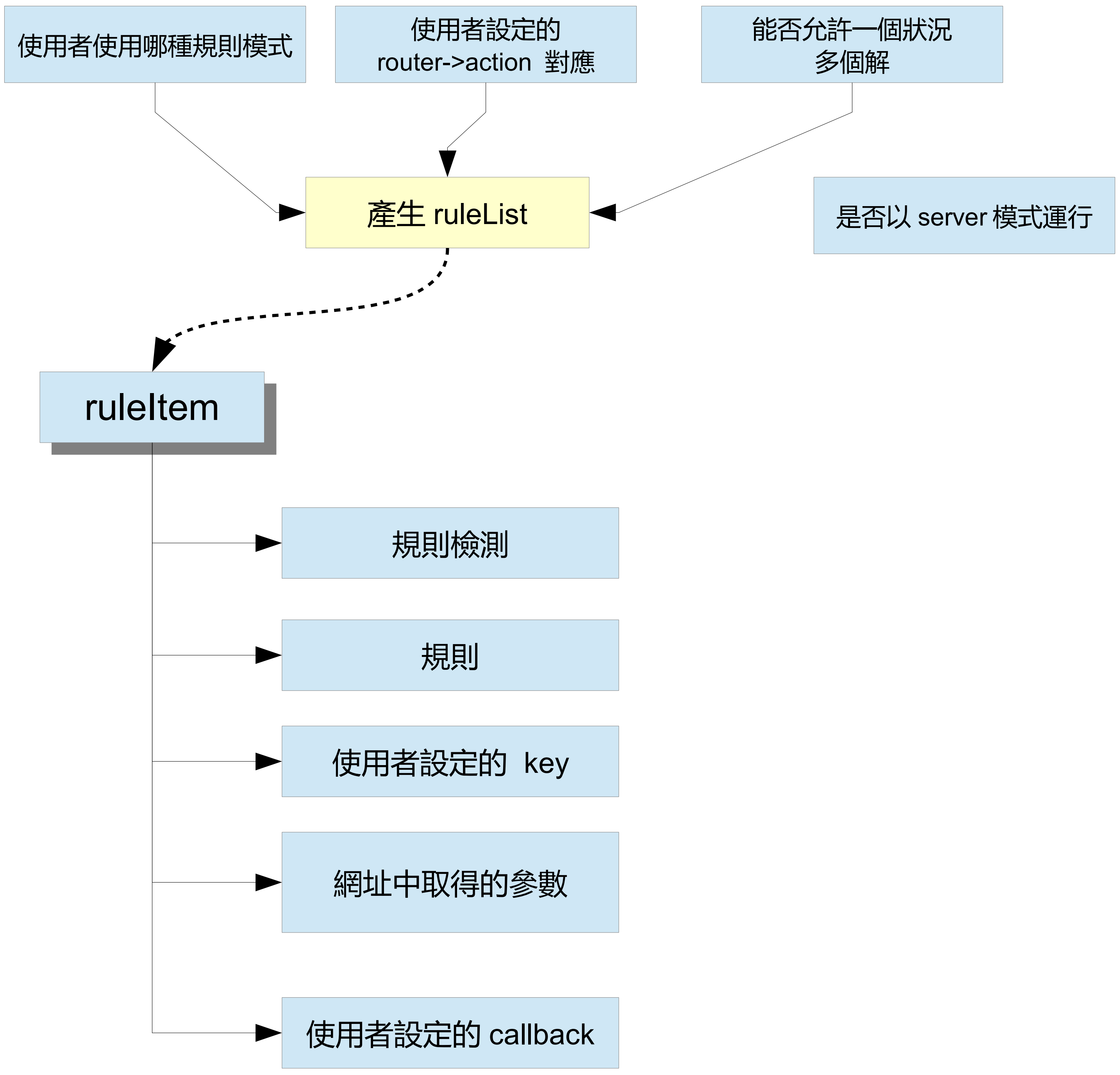
流程



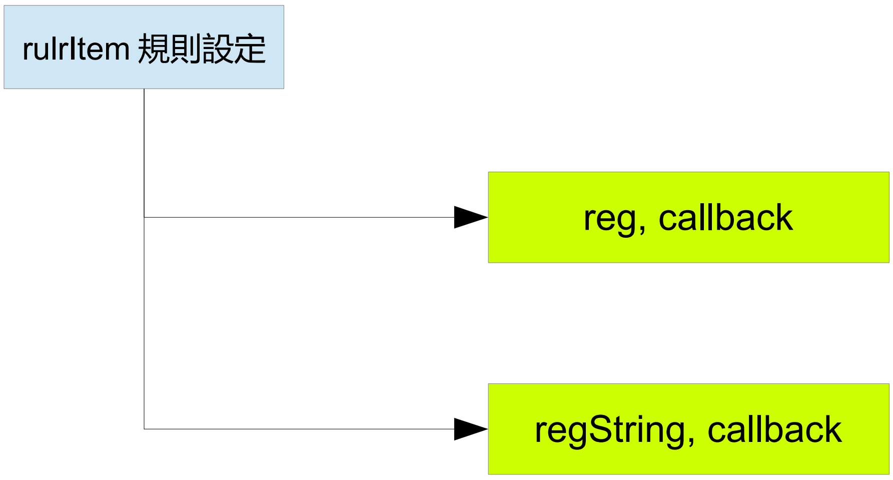
/xyz{A}(.{file})
url
分析節點方法
block
block
block
(優先權 0) 分析 ()
(優先權 1) 分析 {...}
(優先權 1) 分析 {*...}
(優先權 2) 分析 string
Block 解析
(可動用優先權的方法 0 以下)
xyz
{A}
(.{file})
() 解析
(可動用優先權的方法 1 以下)
直接解析
直接解析
s(xyz)
v
.
{file}
直接解析
直接解析
S(.)
v

router 運作流程

建構程序

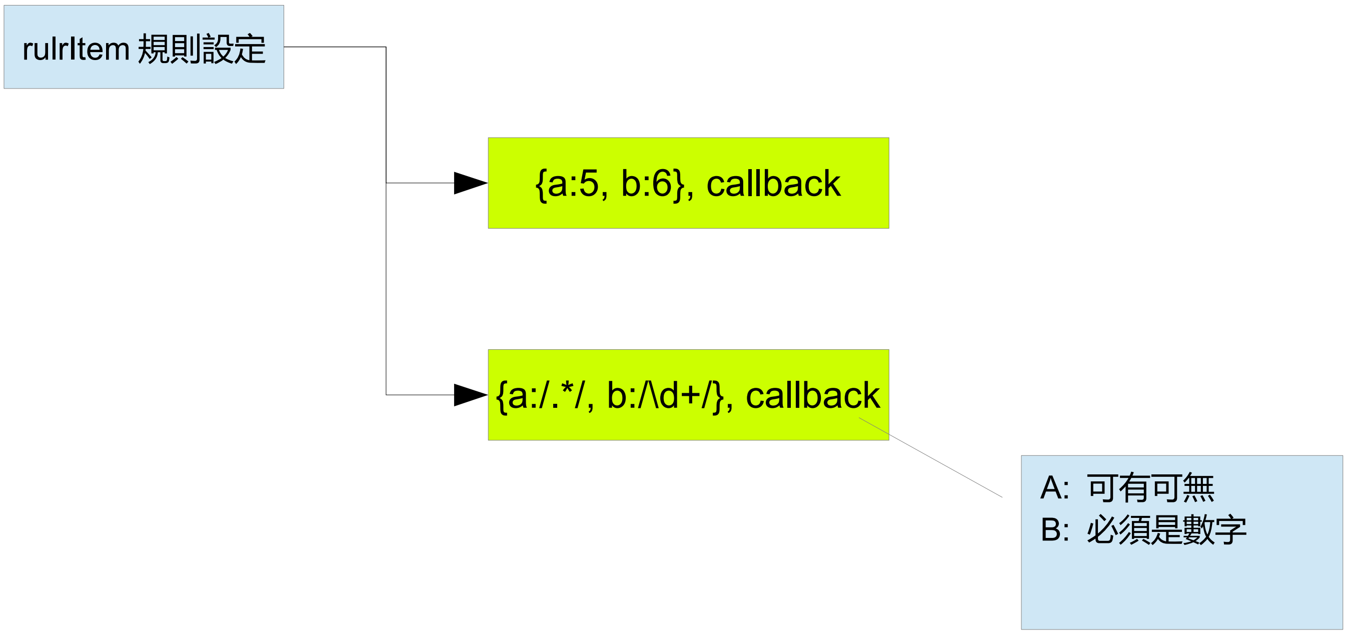

使用者使用哪種規則模式
使用者設定的 router->action 對應
能否允許一個狀況多個解
產生 ruleList
是否以 server 模式運行
ruleItem
規則檢測
規則
使用者設定的 key
網址中取得的參數
使用者設定的 callback

以 reg 格式為規則



以 query 格式為規則

格式 : 網址 #a=...&b=...&c=...



偵測網址變動