
Subject: Re: why no virtual destructor for RegExp?

Posted by [navi](#) on Fri, 25 Jan 2013 13:58:33 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hi All,

Just an update what I am up to so far. I manage to code a replica of preg_replace of php. due to the difference in nature of variable type, I had to almost 4 overloaded function to match all the things "preg_replace" can do in a single function definition.

```
//=====
=====
//      My Extension to RegExp
//=====
=====
```

protected:

```
bool ReplacePos(String& t, int p, int q, const String r);
int  Replace0(String& t, Vector<String>& rv, const int& rv_count, int& offset);
Vector<String> ResolveBackRef(Vector<String>& rv);
Vector<String> Make_rv(const String& r);
```

public:

```
int  ReplaceGlobal(String& t, Vector<String>& rv, const String pattern=NULL, bool backref=false);
//DONE
```

```
int  ReplaceGlobal(String& t, Vector<String>& rv, int offset, const String pattern=NULL, bool
backref=false); //DONE
```

```
int  ReplaceGlobal(String& t, const String r, const String pattern=NULL, bool backref=false);
//DONE
```

```
int  ReplaceGlobal(String& t, const String r, int offset, const String pattern=NULL, bool
backref=false); //DONE
```

```
int  ReplaceGlobal(String& t, Callback cbr, const String pattern=NULL, bool backref=false); // NOT
YET
```

```
int  ReplaceGlobal(String& t, Callback cbr, int offset, const String pattern=NULL, bool
backref=false); //NOT YET
```

My plan is to implement 6 flavor of ReplaceGlobal() and 6 flavor of Replace(). all based on how replace text or callback function are given.

replace text can either be given as single String in Parentheses format just like a regular expression. ie (replace_a)(replace_b)(replace_c). Or can be given a vector containing three Strings. Also callback function can be passed in place of replace texts.

regex pattern can also be given as argument. instead of setting it using `SetPattern()`

backref = is a on/off switch for resolving/expanding back reference given in replace text as `/1`, `/2`, `/3` etc. where the number is the number of the match string caught by the regex pattern.

Thanks & Regards

Navi
