Making Agile a Reality ®

Just Enough Regular Expressions for Cucumber

July 20, 2010 | by: **Richard Lawrence**

Jon Archer wrote last week about how **Cucumber makes knowledge of regular expressions important**. He's right: Regular expressions are the key to Cucumber's flexibility. Well-crafted regular expressions developers find them mysterious and overwhelming.

Fortunately, you don't need regular expressions like this one to wield the power of Cucumber:

```
(?:[a-z0-9!#$%&'*+/=?^_`{|}~-]+(?:.[a-z0-
9!#$%&'*+/=?^_`{|}~-]+)*|"(?:[x01-x08x0bx0cx0e-
x1fx21x23-x5bx5d-x7f]|[x01-x09x0bx0cx0e-
x7f])*")@(?:(?:[a-z0-9](?:[a-z0-9-]*[a-z0-
9])?.)+[a-z0-9](?:[a-z0-9-]*[a-z0-9])?|[(?:
(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?).){3}
```

Richard Lawrence

He trains and coaches teams and organizations to become happier and more productive. He draws on a diverse background in software development, engineering, anthropology, and political science. Richard is a Scrum Alliance Certified Enterprise Coach and Certified Scrum Trainer, as well as a certified trainer of the accelerated learning method, Training from the Back of the Room.

```
x5ax53-x7f]|[x01-x09x0bx0cx0e-x7f])+)])
```

In fact, if you use regular expressions like this in your step definitions, you've gone too far. (This regular expression, in case you're wondering, matches the official spec for valid email addresses.)

As with most things, the 80/20 rule applies. There are a handful of useful patterns that are sufficient to make you a Cucumber power user.

Anchors

The regular expression `I'm logged in` matches `I'm logged in` and `I'm logged in as an admin`. To avoid ambiguous matches, use `^I'm logged in$`.

The caret at the beginning anchors to the beginning of the string. The dollar at the end does the same with the end of the string. Use these with all your step definitions and you won't have surprise matches.

Wildcards and quantifiers

Matching specific words is fine. But you often want flexibility to match a variety of strings. Here are some common patterns for non-exact matches.

`.*`	matches anything (or nothing), literally "any character (except a newline) 0 or more times"
`.+`	matches at least one of anything
`[0-9]*` or `d*`	matches a series of digits (or nothing)

richard.lawrence@agileforall.c

Twitter

LinkedIn

Subscribe to RSS Feed
Blog posts by this author

search articles

All

Uncategorized

`"[^"]*"`	matches something (or nothing) in double quotes
`an?`	matches *a* or *an* (the question mark makes the *n* optional)

Capturing and not capturing

When you put part of a regular expression in parentheses, whatever it matches gets captured for use later. This is known as a “capture group.” In Cucumber, captured strings become step definition parameters. Typically, if you’re using a wildcard, you probably want to capture the matching value for use in your step definition. Here’s a Cuke4Nuke example,

```
[Given(@"^I'm logged in as an? (.*)$")]
public void ImLoggedInAsA(string role)
{
// log in as the given role
}
```

If your step is `Given I'm logged in as an admin`, then the step definition gets passed `"admin"` for `role`.

Cuke4Nuke converts captured strings to the step definition parameter type, which is handy for step definitions like this:

```
[Given(@"^I have (d+) cukes$")]
public void IHaveNCukes(int cukeCount)
{
```

The step `Given I have 42 cukes` means the step definition gets called with `42` (as an integer) for `cukeCount`.

Sometimes, you have to use parentheses to get a regular expression to work, but you don't want to capture the match.

For example, suppose I want to be able to match both `When I log in as an admin` and `Given I'm logged in as an admin`. After all, both step definitions do the same thing. There's no reason to have duplicated automation code in my step definitions simply because one is a Given and one is a When.

I might write something like this:

```
[When(@"^(I'm logged|I log) in as an? (.*)$")]
public void LogInAs(string role)
{
// log in as the given role
}
```

The parentheses and pipe indicate a logical OR, just what I need to match two different strings.

This will fail to run, though. My regular expression is capturing two strings, but my step definition method only takes one. I need to designate the first group as non-capturing like this:

```
(.*)$")]
public void LogInAs(string role)
{
// log in as the given role
}
```

Now, with the addition of `?:` at the beginning of the group, it will perform as I expect.

By the way: You may be wondering how the attribute can be When and still match `Given I'm logged in as an admin`. It turns out that in Cuke4Nuke, just like in Cucumber for Ruby, it doesn't matter whether you use Given, When, or Then to define a step definition. They're all step definitions and are interchangeable. It's fairly common for today's When to be tomorrow's Given, so this is a nice feature.

Just enough

This is only the tip of the regular expression iceberg. Here's a **book** and **website** if you're interested in going deeper. But for day-to-day work with Cucumber, anchors, simple wildcards and quantifiers, and capturing and non-capturing groups are all you need.

PREVIOUS ARTICLE NEXT ARTICLE

SHARE THIS

Your email address will not be published. Required fields are marked *

Name *

Email *

Website

Notify me of follow-up comments by email.

POST COMMENT

Notify me of new posts by email.

Jon Archer says:
July 20, 2010 at 6:27 pm

This is just the kind of info I needed: a simple cheat sheet of techniques for the commonest things I think I'm going to need to do w/Cuke4Nuke. I'm very glad you wrote it up. Thanks also for the link to my post :-)

One thing I did notice here that I want to ask about. After explaining the anchors and their use, I note that you didn't actually have them in place on your examples in the captures section. Now is that just accidental, or is there a cunning rationale to that which I'm missing?

Richard Lawrence says:
July 20, 2010 at 6:40 pm
Good catch. I'll update the post.

Reply

Abder-Rahman says:
July 21, 2010 at 9:34 am
Thanks a lot for this excellent article. What was missing for me especially when coming to regular expressions.

Thanks a lot.

Reply

Jon Archer says:
July 21, 2010 at 10:27 am
So this may be due to the fact that I'm recovering from a 3 year bout of management and that C# is new to me, but I had a few moments of trouble getting the pattern for capturing quoted text to work, i.e. this one: "[^"]*"

It seems it was all down to the fact that when using the @"..." form in C# to declare the string pattern one escapes inline quotes by double quoting not using the "

```
[When(@"^.*username ""
([^""]*)"".*password ""([^""]*)"".*$")]
public void Login(string username, string
password)
{
user =
securityFactility.Authenticate(username,
password);
}
```

Although I think I prefer

```
[When(@"^.*username ""(.*)"".*password ""
(.*)"".*$")]
```

Reply

Richard Lawrence says:
July 21, 2010 at 11:31 am

You're right, Jon. I think I've only used that pattern with Ruby and Java, so I didn't think about the .NET escape sequence. It definitely reads better with `(.*)`, but it's less precise. Since the * is greedy, `"(.*)"` will jump over double quotes to match a later double quote, which can lead to some surprising matches.

Reply

Colin Jackson says:
October 29, 2017 at 5:38 pm

Reply