Introduction to Surlex

Cody Soyland
Djangocon 2009

Purpose

- Alternative to regex matching/capturing
- Actually, a regex generator
- Concise syntax
- Designed for data extraction

Why reinvent the wheel?

Regex vs. Surlex

Basic matching:

- Surlex: `<var>`
- Regex: `(?P<var>.+)`

Makes easy-to-read URL patterns:

- Surlex:
  - `/blog/<year>/<month>/<slug>/`
- Regex:
  - `/blog/(?P<year>.+)/(?P<month>.+)/(?P<slug>.+)/`

Embedded Regex Allows Specificity

- Specific is better than general
- Target: the string “2009”
  - Bad: .+
  - Better: \d+
  - Perfect: \d{4}
- Surlex Equivalents:
  - <year>
  - <year=\d+>
  - <year=\d{4}>

Wait, I'm back to writing regular expressions?!

Macros ease common tasks

```
surlex: <slug:s>
==
surlex: <slug=[\w-]+>
==
regex: (?P<slug>[\w-]+)
```

Macros shorten things a lot

Built-in date and slug macros enable conciseness:

```
/blog/<year:Y>/<month:M>/<day:d>/<slug:s>/
==
/blog/(?P<year>\d{4})/(?P<month>(jan|feb|mar|
apr|may|jun|jul|aug|sep|oct|nov|dec))/(?
P<day>\d{1,2})/ (?P<slug>[\w-]+)/
```

Matching without capturing

Simply omit the variable name

Macro matching a slug: <:s> (regex: [\w-]+)
Regex matching a digit: <=\d> (regex: \d)

Other features

Optional strings in parentheses:

Surlex: “/blog/(<year:Y>/)”

Regex: “/blog/((?P<year>\d{4})/)?”

Wildcards:

Surlex: “/*.*”

Regex: “/.*\..*”

Start and end of string:

“^” and “$” work just like regex

Django Integration

```
from surlex.dj import surl

urlpatterns = patterns('',
    surl(r'^blog/<year:Y>/<month:M>/<day:d>/<slug:s>/$',
        blog.views.post_detail,
        name='blog_post_detail'),
)
```

Try it yourself!

pip install surlex
or
git clone git://github.com/codysoyland/surlex.git

Thanks!

Cody Soyland
codysoyland@gmail.com
http://www.codysoyland.com/
http://www.github.com/codysoyland/