Regular Expressions

April 29th, 2008

MITRE

POSIX: The Current Standard

- 1003.2 Chosen because:
  - Small set of features
  - POSIX is more standardized than if we created our own subset
  - At the time it was felt that there should be full POSIX support available on many regular expression engines.

POSIX 1003.2: Concerns

- Lack of Unicode functionality expressed within the standard.
- Not many regular expression engines fully support POSIX 1003.2
  - Many do not support character collating
- Many are unfamiliar with POSIX 1003.2 character class syntax.
  - Example: Submissions have included “\d” instead of “[:digit:]” in the past.

Other Flavors

- There are many other flavors of regular expression syntax out there:
  - Perl
  - Java
  - .NET
  - Ruby
  - XML Schema
  - XQuery 1.0 / XPath 2.0
  - Python

Problem: Too Many Flavors

- Many programming languages utilize their own flavor of regular expressions with their own character classes or functionalities.
  - "^foobar$" does not mean the same thing to XML Schema and .NET regular expressions
- We need to find a common subset of features, or utilize an existing standard that bridges the gap between programming environments

What do we want?

- Small set of features
- Unicode support
- Programming language independence
- Established standard
- Widely used

Previous Suggestions

- Unicode (UTS 18)
- PCRE
- XPath 2.0 / XQuery 1.0

MITRE

Unicode Regular Expression Standard

- What it is not
  - It is not a complete syntactical form of regular expressions such as the Perl or POSIX flavors.
- What it is
  - The Unicode Technical Standard, No. 18 describes how regular expression engines can be adapted to handle Unicode within regular expressions.

Perl Compatible Regular Expressions

- Libpcre is a C/C++ library which allows the use of regular expressions that follow Perl's syntax.

- Advantages
  - It is widely used
  - It can be used by C/C++ and the syntax is obviously compatible with Perl by default
  - Most modern regular expression engines handle Perl-like regular expressions

Perl Compatible Regular Expressions (cont)

- Disadvantages
  - PCRE (libpcre) is not fully Perl Compatible (but mostly)
    - Some Unicode classes found in Perl such as \p{Letter} is not found within PCRE. PCRE does use an equivalent of \p{L}
  - libpcre is a C/C++ library only
  - The Perl regular expression dictionary is large
    - Many character classes
  - Many features

MITRE

XQuery 1.0 / XPath 2.0

- XML Schema documents support a small set of regular expression capabilities for validation.
- XPath 2.0 regular expressions were built on top of XML Schema regular expression rules to allow more features
  - Lazy quantifiers
  - Grouping and back-referencing
  - ^ and $ anchors

XQuery 1.0 / XPath 2.0 (cont)

- Advantages
  - Developers are already processing XML Documents so as long as they can execute XPath 2.0 queries, they can do matching
    - Example query: fn:matches(“foobar”, “fo\w+bar”)
  - Supports commonly used character classes
  - Standard is small
  - Supported by Saxon for .NET and Java
  - Supported by XQilla (Xerces-C) for C++
  - Similar to Perl syntax
  - UTS #18 Level 1 support

XQuery 1.0 / XPath 2.0 (cont)

- Disadvantages
  - XSLT 2.0 / XQuery 1.0 / XPath 2.0 are not widely used at this time and are still being adopted
  - .NET Framework 3.5 does not support these technologies right now out of the box
    - This is supposedly in development for the next release of the .NET Framework
  - Two character classes (\i, and \c) are specific to XPath 2.0 and not supported by any other regular expression engines to my knowledge.
    - These character classes are specific to XML documents and should not be used anyways.
  - Does not support {n} (matched exactly n times)or {n,m} (matched at least n but no more than m) syntax

Discussion

MITRE