Field.pm

Table of Contents
Field.pm

NAME	1
SYNOPSIS	1
DESCRIPTION	1
EXPORT	1
METHODS	1
new()	1
tag()	1
indicator(indno)	1
subfield(code)	1
subfields()	1
data()	2
add_subfields(code,text[,code,text ...])	2
update()	2
replace_with()	2
as_string([$subfields])	2
as_formatted()	2
as_usmarc()	2
clone()	3
warnings()	3
SEE ALSO	3
TODO	3
LICENSE	3
AUTHOR	3

NAME

MARC::Field - Perl extension for handling MARC fields

SYNOPSIS

```
  use MARC::Field;

  my $field = MARC::Field->new( 245, '1', '0',
       'a' => 'Raccoons and ripe corn / ',
       'c' => 'Jim Arnosky.'
  );
  $field->add_subfields( "a", "1st ed." );
```

DESCRIPTION

Defines MARC fields for use in the MARC::Record module. I suppose you could use them on their own, but that wouldn‘t be very interesting.

EXPORT

None by default. Any errors are stored in `$MARC::Field::ERROR`, which `$MARC::Record` usually bubbles up to `$MARC::Record::ERROR`.

METHODS

new()

The constructor, which will return a MARC::Field object. Typically you will pass in the tag number, indicator 1, indicator 2, and then a list of any subfield/data pairs. For example:

```
  my $field = MARC::Field->new(
       245, '1', '0',
       'a' => 'Raccoons and ripe corn / ',
       'c' => 'Jim Arnosky.'
  );
```

Or if you want to add a field < 010 that does not have indicators.

```
  my $field = MARC::Field->new( '001', ' 14919759' );
```

tag()

Returns the three digit tag for the field.

indicator(indno)

Returns the specified indicator. Returns `undef` and sets `$MARC::Field::ERROR` if the *indno* is not 1 or 2, or if the tag doesn‘t have indicators.

subfield(code)

When called in a scalar context returns the text from the first subfield matching the subfield code.

```
    my $subfield = $field->subfield( 'a' );
```

Or if you think there might be more than one you can get all of them by calling in a list context:

```
    my @subfields = $field->subfield( 'a' );
```

If no matching subfields are found, `undef` is returned in a scalar context and an empty list in a list context.
If the tag is less than an 010, `undef` is returned and `$MARC::Field::ERROR` is set.

subfields()

Returns all the subfields in the field. What‘s returned is a list of lists, where the inner list is a subfield code and the subfield data.
For example, this might be the subfields from a 245 field:

```
          [
            [ 'a', 'Perl in a nutshell :' ],
            [ 'b', 'A desktop quick reference.' ],
          ]
```

data()

Returns the data part of the field, if the tag number is less than 10.

add_subfields(code,text[,code,text ...])

Adds subfields to the end of the subfield list.

```
    $field->add_subfields( 'c' => '1985' );
```

Returns the number of subfields added, or `undef` if there was an error.

update()

Allows you to change the values of the field. You can update indicators and subfields like this:

```
  $field->update( ind2 => '4', a => 'The ballad of Abe Lincoln');
```

If you attempt to update a subfield which does not currently exist in the field, then a new subfield will be appended to the field. If you don‘t like this auto-vivification you must check for the existence of the subfield prior to update.

```
  if ( $field->subfield( 'a' ) ) {
    $field->update( 'a' => 'Cryptonomicon' );
  }
```

If you want to update a field that has no indicators or subfields (000-009) just call update() with one argument, the string that you would like to set the field to.

```
  $field = $record->field( '003' );
  $field->update('IMchF');
```

Note: when doing subfield updates be aware that `update()` will only update the first occurrence. If you need to do anything more complicated you will probably need to create a new field and use `replace_with()`.
Returns the number of items modified.

replace_with()

Allows you to replace an existing field with a new one. You need to pass `replace()` a MARC::Field object to replace the existing field with. For example:

```
  $field = $record->field('245');
  my $new_field = new MARC::Field('245','0','4','The ballad of Abe Lincoln.');
  $field->replace_with($new_field);
```

Doesn‘t return a meaningful or reliable value.

as_string([$subfields])

Returns a string of all subfields run together. A space is added to the result between each subfield. The tag number and subfield character are not included.
Subfields appear in the output string in the order in which they occur in the field.
If `$subfields` is specified, then only those subfields will be included.

```
  my $field = MARC::Field->new(
                245, '1', '0',
                        'a' => 'Abraham Lincoln',
                        'h' => '[videorecording] :',
                        'b' => 'preserving the union /',
                        'c' => 'A&E Home Video.'
                );
  print $field->as_string( 'abh' ); # Only those three subfields
  # prints 'Abraham Lincoln [videorecording] : preserving the union /'.
```

Note that subfield h comes before subfield b in the output.

as_formatted()

Returns a pretty string for printing in a MARC dump.

as_usmarc()

Returns a string for putting into a USMARC file. It‘s really only useful by
`MARC::Record::as_usmarc()`.

clone()

Makes a copy of the field. Note that this is not just the same as saying

```
    my $newfield = $field;
```

since that just makes a copy of the reference. To get a new object, you must

```
    my $newfield = $field->clone;
```

Returns a MARC::Field record.

warnings()

Returns the warnings that were created when the record was read. These are things like "Invalid indicators converted to blanks".

The warnings are items that you might be interested in, or might not. It depends on how stringently you‘re checking data. If you‘re doing some grunt data analysis, you probably don‘t care.

SEE ALSO

See the "SEE ALSO" section for *MARC::Record*.

TODO

See the "TODO" section for *MARC::Record*.

LICENSE

AUTHOR

Andy Lester, `< <andy@petdance.com>`