EPrints 入力項目変更事例

はじめに
１．Title ヨミフィールドの追加
２．既存フィールドを利用した Creator ヨミの追加
３．フリーキーワード入力欄の変更
４．Language フィールドの追加
５．Subject list のカスタマイズ・日本語化
６．Type.nii の自動設定
７．その他

はじめに

以下で紹介する事例は EPrints の既定のフィールドを改造した一例です。修正を要するファイルは最低限

- ArchiveMetadataFieldConfig.pm
- metadata-types.xml
- phrases-ja.xml：日本語による表示のためにあらかじめ作成しておく必要がある。
- ArchiveOAIConfig.pm：DC メタデータ生成サブルーチン（sub eprint_to_unqualified_dc）を流用して junii 対応のメタデータを生成するサブルーチン（以下の事例中では sub eprint_to_unqualified_junii としている）を作成しておく必要がある。

です。当該アーカイブディレクトリ下の cfg ディレクトリに存在するこれらのファイルを修正します。

場合によってはその他のモジュールの修正も必要になりますが、いずれにせよ EPrints の公式ドキュメントにはあまり詳しい手引きは載っていません。

入力項目の変更等、Mysql のテーブル構造の変化を生じさせる修正を反映させるためには、アーカイブを一旦削除して、一から生成しなおす必要があります（ドキュメントには SQL に知識があれば、アーカイブを消去せずに変更できる旨記述してありますが、一方 SQL コマンドによる具体的手順の記述は見当たりません）。

よって、運用前に入力項目等充分検討、テストしておく必要がありそうです。

アーカイブ削除後、bin/configure_archive 以下の一連のコマンドを実行、および httpd の stop/start を行う必要がありますが、bin/configure_archive においては”Create config files”の問いに対して”no”とします。さもないとせっかく修正したファイルがデフォルトの内容に置き換えられてしまいます。

以下の記述では、ファイル修正後のアーカイブ削除、再生成の手順については記述を省略しています。

また新規のフィールドを追加する場合、フィールド名は任意ですので、この手順書と同じ名称にする必要はありません。

１．Title ヨミフィールドの追加

目的：日本語の場合ヨミが必要なため

●ArchiveMetadataFieldConfig.pm の修正
　以下の記述を追加（eprint section 内、title の次に追加）

```
{ name => "titletrans", type => "longtext", multilang=>0 },
```

●metadata-types.xml の修正
　各 eprint type の中、（title の次）に

```
<field name="titletrans" />
```

を記述します。

●phrases-ja.xml に以下の例ように補記します。

```
<ep:phrase ref="eprint_fieldname_titletrans">Title ヨミ</ep:phrase>
<ep:phrase ref="eprint_fieldhelp_titletrans">Title のヨミを分かち書きにより記述します。Title が日本語を含む場合のみ記述対象となります。
<br />Example: <span class="example">ジカン ノ レキシ</span>
<br />Example: <span class="example">エンジニア ノタメノ スウガク</span>
<br />Example: <span class="example">エンジニア ノタメノ スウガク ダイ 5 ハン~</span>
<br />Example: <span class="example">セカイ ノ エコシステム ダイ 26 カン セカイ ノ イリエ</span>
</ep:phrase>
```

●ArchiveOAIConfig.pm の修正
　sub eprint_to_unqualified_junii 中の

```
push @dcdata, [ "title", $eprint->get_value( "title" ) ];
```

に続けて

```
push @dcdata, [ "title.transcription", $eprint->get_value( "titletrans" ) ];
```

を記述します。

２．既存フィールドを利用した Creators ヨミの追加

目的：日本人名ヨミが必要なため。EPrints における creators と editors は４つの要素で構成され、これらが内部的に処理されて creator（あるいは contributor）としてのメタデータが生成されます。
すなわち ArchiveMetadataFieldsConfig.pm 項目定義で"name"タイプの場合、

[　title　][given name][　family name　][　lineage　]

の４つです。
title と lineage は非表示にすることができます。姓と名の順序も変更することができます。

我が国では個人名に対し title と lineage を使用することはないと思われるので、
lineage に姓のヨミ、title(honourific)に名のヨミを当てることも可能です。
その場合、

[　姓　][　名　][姓のヨミ][名のヨミ]

とし、ヨミ部分は creator.transcription としてメタデータが生成されるようにします。
なお、editors に関しても同様の方法を適用できます。

●ArchiveMetadataFieldsConfig.pm の変更

{ name => "creators", type => "name", multiple => 1, input_boxes => 4, hasid => 1, input_id_cols=>20, family_first=>1, hide_honourific=>1, hide_lineage=>1 }

を

{ name => "creators", type => "name", multiple => 1, input_boxes => 4, hasid => 1, input_id_cols=>20, family_first=>1, hide_honourific=>0, hide_lineage=>0 },

にし、title と lineage を表示させるようにします。

●ArchiveConfig.pm の変更

```
$c->{field_defaults}->{input_name_cols} = {
  honourific=>8,
  given=>20,
  family=>20,
  lineage=>8
};
```

を

```
$c->{field_defaults}->{input_name_cols} = {
  given=>10,
  family=>10,
  honourific=>20,
  lineage=>20
};
```

に変更します。

「=>」に続く値はテキストボックスの長さ（バイト）です。適当なバイト長にしてください。

●/opt/eprints2/perl_lib/EPrints/MetaField?/Name.pm の変更

```
 sub get_input_bits
 {
        my( $self, $session ) = @_;

        my @namebits;
        unless( $self->get_property( "hide_honourific" ) )
        {
                push @namebits, "honourific";
        }
        if( $self->get_property( "family_first" ) )
        {
                push @namebits, "family", "given";
        }
        else
        {
                push @namebits, "given", "family";
        }
        unless( $self->get_property( "hide_lineage" ) )
        {
                push @namebits, "lineage";
        }

        return @namebits;
 }
```

を

```
sub get_input_bits
{
	my( $self, $session ) = @_;

	my @namebits;
	if( $self->get_property( "family_first" ) )
	{
		push @namebits, "family", "given";
	}
	else
	{
		push @namebits, "given", "family";
	}
	unless( $self->get_property( "hide_lineage" ) )
	{
		push @namebits, "lineage";
	}
	unless( $self->get_property( "hide_honourific" ) )
	{
		push @namebits, "honourific";
	}

	return @namebits;
}
```

に変更します。

次に creator のヨミを creator.transcription として出力するための修正を行います。

●/opt/eprints2/perl_lib/EPrints/Utils.pm の修正

このファイル中の sub make_name_string において [family][given][lineage][honourific]を組み合わせて creator エレメントを生成しています。

よって、sub make_name_string では{family}{given}のみを組み合わせて creator を生成させ、また sub make_name_string を流用して sub make_name_string_trans を作り、これにより{lineage}{honourific}のみを組み合わせ、creator.transcription を生成させることにします。
lineage に姓のヨミ、title(honourific)に名のヨミを当てると、流用して sub make_name_string_trans を記述する際、多少書き換えが少なくて済むでしょう。

変更例）

```
sub make_name_string
{
	my( $name, $familylast ) = @_;

	my $firstbit = "";
	if( defined $name->{given} )
	{
		$firstbit= $name->{given};
	}



	my $secondbit = "";
	if( defined $name->{family} )
	{
		$secondbit = $name->{family};
	}

	if( defined $familylast && $familylast )
	{
		return $firstbit.", ".$secondbit;
	}

	return $secondbit.", ".$firstbit;
}

sub make_name_string_trans
{
	my( $name, $familylast ) = @_;

	my $firstbit = "";
	if( defined $name->{honourific} && $name->{honourific} ne "" )
	{
		$firstbit = $name->{honourific}." ";
	}



	my $secondbit = "";
	if( defined $name->{lineage} && $name->{lineage} ne "" )
	{
		$secondbit .= ", ".$name->{lineage};
	}
```

```
        if( defined $familylast && $familylast )
        {
                return $firstbit." ".$secondbit;
        }

        return $secondbit.", ".$firstbit;
}
```

●ArchiveOAIConfig.pm の修正
sub eprint_to_unqualified_junii 中の

```
    my $creators = $eprint->get_value( "creators", 1 );
    if( defined $creators )
    {
        foreach my $creator ( @{$creators} )
        {
            push @dcdata, [ "creator", EPrints::Utils::make_name_string( $creator ) ];
        }
    }
```

を

```
    my $creators = $eprint->get_value( "creators", 1 );
    if( defined $creators )
    {
        foreach my $creator ( @{$creators} )
        {
            push @dcdata, [ "creator", EPrints::Utils::make_name_string( $creator ) ];
            push  @dcdata,["creator.transcription",  EPrints::Utils::make_name_string_trans
           (  $creator  ) ];
        }
    }
```

に書き換えます。

３．フリーキーワード入力欄の変更

目的：EPrints のフリーキーワード入力欄は、1 個のテキストボックスに複数のキーワードを入力するようになっています。
英語等の入力ではあまり問題ないのでしょうが、「一つのテキストボックスに 1 個のキーワード」を入力するよう修正してみます。またそれらを一つずつ、junii に subject として引き渡せるようにします。

●ArchiveMetadataFieldsConfig.pm の修正

{ name => "keywords", type => "longtext", input_rows => 2 },
を

{ name => "keywords", type => "text", multiple => 1, input_boxes => 4, input_add_boxes=>1, input_cols=>10},

に変更します。

●phrases-ja.xml において、1 入力 BOX に１個ずつキーワードを入力する旨記述しておきます。

●ArchiveOAIConfig.pm の修正

sub eprint_to_unqualified_junii 中に

```
 my $kw = $eprint->get_value( "keywords" );
 if( defined $kw )
 {
        foreach my $kwords ( @{$kw} )
        {
        push @dcdata, [ "subject", $kwords ];
        }
 }
```

を記述します。

４．Language フィールドの追加

目的：明示的に言語を指定する局面がないため。junii では必須なため。

●ArchiveMetadataFieldConfig.pm の eprint section 内に以下を追加します。

このセクション内なら場所はどこでも良いと思われます。

{name=>"lang",type=>"text", multiple => 1, input_boxes => 1, hasid=>1, input_add_boxes=>1, input_cols=>3}

※"language"という名称が EPrint ですでに規定されているようで（その役割は未確認)、
かつこの language の働きがよくわからないので、バッティングしないよう名称を"lang"としてみました。

●metadata-types.xml の各 type の中に

<field name="lang" required="yes"/>

を追加します（required="yes"により必須項目扱い)。

このとき、各 eprint type の<page name="hogehoge" />で、どの画面においてどの項目を表示するか決めているらしいので、ここでの記述位置は画面を意識しておきましょう。
※東大では<page name="pubinfo" /> 内の最後に記述した（つまり Publication Information 画面の最後の項目として表示される)。

●phrases-ja.xml に以下のように記述します（あくまで例)。

```
    <ep:phrase ref="eprint_fieldname_lang">言語</ep:phrase>
    <ep:phrase ref="eprint_fieldhelp_lang">ドキュメント中で用いられている言語をコードで記述します。~
 日英併記など複数言語を使用している場合は、[More Spaces]で記入欄を追加して記述して下さい。
 <br />Example: <span class="example">日本語= jpn</span>
 <br />Example: <span class="example">英語= eng</span>
     </ep:phrase>
```

※ファイル中の記述位置は多分上述の metadata-types.xml に依存すると思われるのでどこでも良いと思いますが、東大の場合は ISSN の次に記述してみました。

●ArchiveOAIConfig.pm の修正

sub eprint_to_unqualified_junii 中の最後のあたりに

```
 my $lang = $eprint->get_value( "lang", 1 );
 if( defined $lang )
 {
        foreach my $language ( @{$lang} )
```

```
        {
        push @dcdata, [ "language..ISO639-2", $language ];
        }
}
```

を記述します。

５．Subject list のカスタマイズ・日本語化

目的:デフォルトの Subject list を変更するため。ここでは例として科研費の分野体系としてみる。
Subject list で日本語を使用するには XML ファイルとして、list ファイルを作成しなければならない。
EXCEL で Subjects リストを作成し、簡単な Perl スクリプトにかけて XML 形式のファイルを生成させる例を述べる。

●EXCEL 表
subjectid、name、parents、depositable の４項目（４列）の表。記述については eprints-docs.txt を読んでください。
これを TAB 区切り text ファイルとして保存します。

●perl スクリプト基本的には以下（ActivePerl による、パソコンで処理した）。

```
#************************************************************
# EXCEL 表＝＞TAB 区切り text データから
# Eprints の subjects の XML ファイルを生成する(言語は ja 固定)
# 生成されたファイルを UTF-8N、LF に変換してサーバに。
#************************************************************

 if(@ARGV<2)
{
        die "入力ファイル名と出力ファイル名を指定してください";
}

#【入力ﾌｧｲﾙ、出力ファイルの OPEN】****************
open(IN,$ARGV[0])||die "$ARGV[0] :$!";
open(OUT,">$ARGV[1]")||die "$ARGV[1] :$!";
#************************************************

print OUT "<eprintsdata>¥n";

while(<IN>)
```

```
 {
   chomp($_);
   print OUT "¥t<record>¥n";

   @buf=split(/¥t/,$_);
   print OUT "¥t¥t<field name=¥"subjectid¥">$buf[0]</field>¥n";
   print OUT "¥t¥t<field name=¥"name¥"><lang id=¥"ja¥">$buf[1]</lang></field>¥n";
   print OUT "¥t¥t<field name=¥"parents¥">$buf[2]</field>¥n";
   print OUT "¥t¥t<field name=¥"depositable¥">$buf[3]</field>¥n";

   print OUT "¥t</record>¥n";
 }

 print OUT "</eprintsdata>¥n";
  #【ファイルクローズ】****************
 close(IN);
 close(OUT);
 #***********************************
```

●出力ファイルは以下のようになる（その他の方法でも、以下の様な形式でファイルを作成すれば良い。なお、本来階層構造のインデントによる表現は必須ではない）。

```
<eprintsdata>
        <record>
                <field name="subjectid">subjects</field>
                <field name="name"><lang id="ja">testja</lang></field>
                <field name="parents">ROOT</field>
                <field name="depositable">FALSE</field>
        </record>
        <record>
                <field name="subjectid">02</field>
                <field name="name"><lang id="ja">文学</lang></field>
                <field name="parents">subjects</field>
                <field name="depositable">FALSE</field>
        </record>
  ~~~~~~~~~~~~~~~~~~~~~~~~省略~~~~~~~~~~~~~~~~~
        <record>
                <field name="subjectid">0851</field>
                <field name="name"><lang id="ja">医用生体工学・生体材料学</lang></field>
                <field name="parents">07</field>
                <field name="depositable">TRUE</field>
        </record>
</eprintsdata>
```

●出力結果を UTF8 にエディタを使って変換し、subjects.xml としてサーバの当該アーカイブの cfg 内に置く（ファイル名は任意である）。

●import_subjects の実行

bin/import_subjects test01 --xml /opt/eprints2/archives/ｱｰｶｲﾌﾞ名/cfg/subjects.xml

xml ファイルの場所をフルパスで指定しないと別の場所を探しに行ってしまうようだ（･･･/perl_lib/･･･）？

６．Type.nii の自動設定

目的：EPrints のデータ登録の流れからして、どの局面で junii の type.nii に該当するのデータを登録させるか、しっくりくる画面がなかった。
データ登録時冒頭の eprint type の選択を type.nii の選択に全面変更することも考えられるが、
・Eprints のデータ登録画面は、どの eprint type を選択したかによってその後の画面が異なること
・eprint type のタイプ設定自体も生かしたかったこと
から、選択した eprint type によって、自動的に type.nii が設定されるようにしてみた。
以下では一例として

論文、図書の一部、モノグラフ、学位論文＝＞研究成果-論文
図書、特許、その他＝＞研究成果-論文以外
会議やワークショップのアイテム＝＞研究成果リスト-講演会等

になるようにしてみる。

●ArchiveOAIConfig.pm の修正

sub eprint_to_unqualified_junii 中のこれまでの type 関係記述の次に以下を記述。

```
my $type_nii ='研究成果-論文以外';
my $type_ep="";

$type_ep=$ds->get_type_name( $session, $eprint->get_value("type"));

if((~$type_ep=~/論文/)or($type_ep=~/一部/)or($type_ep=~/モノ/)){$type_nii='研究成果-論文';}
if($type_ep=~/アイテム/){$type_nii='研究成果リスト-講演会等';}

push @dcdata, [ "type.nii", $type_nii ];
```

７．その他

７－１．Publisher のヨミフィールド追加

Title ヨミの設定にならえば簡単です（type は text）。

７－２．Creators、Editors の団体名用フィールドの追加

EPrints 所定の creators と editors（双方とも個人名用）とは別に、団体名記述用のフィールド（およびそのヨミ用フィールド）の追加も Title ヨミの設定と同様の手順で可能ですが、複数の団体名が記述できるようフィールドタイプを multiple にしておきます。

※ creators と editors について「個人名用」と「団体名用」を本来分ける必要はなく、個人名と団体名いずれであっても記述できるフィールドがあれば良い。しかし現時点で EPrints デフォルトの creators、editors を無効にしようとするとエラーが発生してしまうため暫定的にこのような対応をとっている。この点が解決されれば本ドキュメントの記述内容を変更する予定。