YOURI, une solution modulaire aux besoins d'un distributeur de paquetages

Guillaume Rousse

Journés francophones de Perl, 2006

Plan

Tâches

Nombreuses tâches

- création
  - infrastructure de construction
  - procédure de soumission
- distribution
  - indexation
  - synchronisation des mirroirs
- maintainance
  - tests QA
  - retour des utilisateurs

Outils

Nombreux outils

- vérification
  - rpmlint
  - deblint
- indexation
  - createrepo
  - rpm2html
- gestion des bugs
  - bugzilla
  - request tracker

Fort besoin d'intégration et d'automatisation

Premier essai

Deux projects distincts

- PLF
- JPackage

Ensemble de scripts ad-hoc

- soumission des paquetages
- mise à jour du dépôt
- synchronisation bugzilla

Implémentation

- différent languages
- système monolithique

Premier essai

Deux projects distincts

- PLF
- JPackage

Ensemble de scripts ad-hoc

- soumission des paquetages
- mise à jour du dépôt
- synchronisation bugzilla

Implémentation

- différent languages
- système monolithique

Premier essai

Deux projects distincts

- PLF
- JPackage

Ensemble de scripts ad-hoc

- soumission des paquetages
- mise à jour du dépôt
- synchronisation bugzilla

Implémentation

- différent languages
- système monolithique

Premières limites

Différent projets = différent besoins

- politiques
- structures du dépôt
- outils employés

Gestion de la diversité

- super-ensemble de tous les besoins
  - cauchemar en terme de maintenance
  - effets de bord continuels
  - problème de mise à l'échelle
- sous-ensemble des besoins communs
  - solution insastisfaisante
- divergences spécifiques
  - perte de réusabilité
  - duplication de code

Premières limites

Différent projets = différent besoins

- politiques
- structures du dépôt
- outils employés

Gestion de la diversité

- super-ensemble de tous les besoins
  - cauchemar en terme de maintenance
  - effets de bord continuels
  - problème de mise à l'échelle
- sous-ensemble des besoins communs
  - solution insastisfaisante
- divergences spécifiques
  - perte de réusabilité
  - duplication de code

Avancement

Présentation

Youri Offers an Upload & Repository Infrastucture

Boite à outils

- ensemble de composants logiciels réutilisables
- quelques outils génériques

Développement obligatoire

- pas destiné à l'utilisateur final
- destiné à un administrateur de projet

Présentation

Youri Offers an Upload & Repository Infrastucture

Boite à outils

- ensemble de composants logiciels réutilisables
- quelques outils génériques

Développement obligatoire

- pas destiné à l'utilisateur final
- destiné à un administrateur de projet

Présentation

Youri Offers an Upload & Repository Infrastucture

Boite à outils

- ensemble de composants logiciels réutilisables
- quelques outils génériques

Développement obligatoire

- pas destiné à l'utilisateur final
- destiné à un administrateur de projet

Objectifs

Versatilité

- pas de format imposé (.rpm, .deb, . . .)
- pas d'outil imposé (bugzilla, DBMS, LDAP, . . .)
- pas de politique imposée
- pas de structure de dépôt imposée

Extensibilité

- ajouter des fonctionalités doit être simple
- en enlever également
- en modifier également

Délégabilité

- laisser les mainteneurs décider
- leur faciliter le travail

Objectifs

Versatilité

- pas de format imposé (.rpm, .deb, . . .)
- pas d'outil imposé (bugzilla, DBMS, LDAP, . . .)
- pas de politique imposée
- pas de structure de dépôt imposée

Extensibilité

- ajouter des fonctionalités doit être simple
- en enlever également
- en modifier également

Délégabilité

- laisser les mainteneurs décider
- leur faciliter le travail

Objectifs

Versatilité

- pas de format imposé (.rpm, .deb, . . .)
- pas d'outil imposé (bugzilla, DBMS, LDAP, . . .)
- pas de politique imposée
- pas de structure de dépôt imposée

Extensibilité

- ajouter des fonctionalités doit être simple
- en enlever également
- en modifier également

Délégabilité

- laisser les mainteneurs décider
- leur faciliter le travail

Conception

Architecture modulaire

- une interface
- plusieurs implémentations

Couche d'abstraction

- dépôt
- media
- paquetage
- gestionnaire de bug
- identification du mainteneur

Greffons

- format de sortie
- source de mise à jour

Conception

Architecture modulaire

- une interface
- plusieurs implémentations

Couche d'abstraction

- dépôt
- media
- paquetage
- gestionnaire de bug
- identification du mainteneur

Greffons

- format de sortie
- source de mise à jour

Conception

Architecture modulaire

- une interface
- plusieurs implémentations

Couche d'abstraction

- dépôt
- media
- paquetage
- gestionnaire de bug
- identification du mainteneur

Greffons

- format de sortie
- source de mise à jour

Avancement

Présentation

`youri-submit`

Synopsis

- entrée : ensemble de paquetages + cible
- exécution d'un ensemble de tests arbitraires
- exécution d'un ensemble d'actions arbitraires

Utilisation collective

- accès direct pour l'admin
- accès indirect pour les utilisateurs

Présentation

`youri-submit`

Synopsis

- entrée : ensemble de paquetages + cible
- exécution d'un ensemble de tests arbitraires
- exécution d'un ensemble d'actions arbitraires

Utilisation collective

- accès direct pour l'admin
- accès indirect pour les utilisateurs

Présentation

`youri-submit`

Synopsis

- entrée : ensemble de paquetages + cible
- exécution d'un ensemble de tests arbitraires
- exécution d'un ensemble d'actions arbitraires

Utilisation collective

- accès direct pour l'admin
- accès indirect pour les utilisateurs

Tests

Exemples

- y a-t-il une révision plus récente déjà présente pour cette cible ?
- y a-t-il une révision plus ancienne déjà présente pour une autre cible ?
- y a-t-il une rupture de l'historique ?

Actions

Exemples

- signer un paquetage
- archiver les anciennes révisions
- versionner les fichiers sensibles
- ...

Avancement

Présentation

`youri-check`

Synopsis

- entrée : ensemble de medias
- exécution d'un ensemble de tests arbitraires
- production d'un ensemble de rapports arbitraires

Utilisation collective

- identification des mainteneurs
- surcharge de configuration

Présentation

`youri-check`

Synopsis

- entrée : ensemble de medias
- exécution d'un ensemble de tests arbitraires
- production d'un ensemble de rapports arbitraires

Utilisation collective

- identification des mainteneurs
- surcharge de configuration

Présentation

`youri-check`

Synopsis

- entrée : ensemble de medias
- exécution d'un ensemble de tests arbitraires
- production d'un ensemble de rapports arbitraires

Utilisation collective

- identification des mainteneurs
- surcharge de configuration

Tests

Exemples

- à l'échelle du paquetage
  - y a-t-il des mises à jour disponibles ?
  - y a-t-il des rapports d'échec de construction ?
- à l'échelle du dépôt
  - y a-t-il des dépendances manquantes ?
  - y a-t-il des conflits avec d'autre paquetages ?
  - y a-t-il des paquetages binaires sans leur paquetages source ?

Rapports

Exemples

- page HTML
- flux RSS
- mails
- ouverture automatique de tickets sur bugzilla
- ...

Avancement

Ajouter une fonctionalité

```
package Whatever::WorkInspection;

=head1 NAME

Whatever::WorkInspection - Enforce employement regulations

=cut

use warnings;
use strict;
use Time::Period;
use base 'Youri::Submit::Check';

sub run {
    my ($self, $package, $repository, $target, $define) = @_;

    my $time = localtime();
    if (!inPeriod($time, 'hr 9am-18pm')) {
        return 'Thou shall not work outside of work hours';
    }

    return;
}

1;
```

Ajouter une fonctionalité

```
package Whatever::WorkInspection;

=head1 NAME

Whatever::WorkInspection - Enforce employement regulations

=cut

use warnings;
use strict;
use Time::Period;
use base 'Youri::Submit::Check';

sub run {
    my ($self, $package, $repository, $target, $define) = @_;

    my $time = localtime();
    if (!inPeriod($time, 'hr 9am-18pm')) {
        return 'Thou shall not work outside of work hours';
    }

    return;
}

1;
```

Ajouter une fonctionalité

```
package Whatever::WorkInspection;

=head1 NAME

Whatever::WorkInspection - Enforce employement regulations

=cut

use warnings;
use strict;
use Time::Period;
use base 'Youri::Submit::Check';

sub run {
    my ($self, $package, $repository, $target, $define) = @_;

    my $time = localtime();
    if (!inPeriod($time, 'hr 9am-18pm')) {
        return 'Thou shall not work outside of work hours';
    }

    return;
}

1;
```

Ajouter une fonctionalité

```
package Whatever::WorkInspection;

=head1 NAME

Whatever::WorkInspection - Enforce employement regulations

=cut

use warnings;
use strict;
use Time::Period;
use base 'Youri::Submit::Check';

sub run {
    my ($self, $package, $repository, $target, $define) = @_;

    my $time = localtime();
    if (!inPeriod($time, 'hr 9am-18pm')) {
        return 'Thou shall not work outside of work hours';
    }

    return;
}

1;
```

Ajouter une fonctionalité

```
package Whatever::WorkInspection;

=head1 NAME

Whatever::WorkInspection - Enforce employement regulations

=cut

use warnings;
use strict;
use Time::Period;
use base 'Youri::Submit::Check';

sub run {
    my ($self, $package, $repository, $target, $define) = @_;

    my $time = localtime();
    if (!inPeriod($time, 'hr 9am-18pm')) {
        return 'Thou shall not work outside of work hours';
    }

    return;
}

1;
```

Remplacer une fonctionnalité

```
package Whatever::LDAP;

=head1 NAME

Whatever::LDAP - Use LDAP to resolve maintainers

=cut

use warnings;
use strict;
use Net::LDAP;
use Youri::Package;
use base 'Youri::Check::Maintainer::Resolver';

sub _init {
    my ($self, %options) = @_;
    # create an LDAP connection with given parameters
    $self->{_ldap} = ...
}

sub get_maintainer {
    my ($self, $package) = @_;
    # retrieve maintainer from LDAP
    my $maintainer = ...
    return $maintainer;
}

1;
```

Remplacer une fonctionnalité

```
package Whatever::LDAP;

=head1 NAME

Whatever::LDAP - Use LDAP to resolve maintainers

=cut

use warnings;
use strict;
use Net::LDAP;
use Youri::Package;
use base 'Youri::Check::Maintainer::Resolver';

sub _init {
    my ($self, %options) = @_;
    # create an LDAP connection with given parameters
    $self->{_ldap} = ...
}

sub get_maintainer {
    my ($self, $package) = @_;
    # retrieve maintainer from LDAP
    my $maintainer = ...
    return $maintainer;
}

1;
```

Remplacer une fonctionnalité

```
package Whatever::LDAP;

=head1 NAME

Whatever::LDAP - Use LDAP to resolve maintainers

=cut

use warnings;
use strict;
use Net::LDAP;
use Youri::Package;
use base 'Youri::Check::Maintainer::Resolver';

sub _init {
    my ($self, %options) = @_;
    # create an LDAP connection with given parameters
    $self->{_ldap} = ...
}

sub get_maintainer {
    my ($self, $package) = @_;
    # retrieve maintainer from LDAP
    my $maintainer = ...
    return $maintainer;
}

1;
```

Remplacer une fonctionnalité

```
package Whatever::LDAP;

=head1 NAME

Whatever::LDAP - Use LDAP to resolve maintainers

=cut

use warnings;
use strict;
use Net::LDAP;
use Youri::Package;
use base 'Youri::Check::Maintainer::Resolver';

sub _init {
    my ($self, %options) = @_;
    # create an LDAP connection with given parameters
    $self->{_ldap} = ...
}

sub get_maintainer {
    my ($self, $package) = @_;
    # retrieve maintainer from LDAP
    my $maintainer = ...
    return $maintainer;
}

1;
```

Remplacer une fonctionnalité

```
package Whatever::LDAP;

=head1 NAME

Whatever::LDAP - Use LDAP to resolve maintainers

=cut

use warnings;
use strict;
use Net::LDAP;
use Youri::Package;
use base 'Youri::Check::Maintainer::Resolver';

sub _init {
    my ($self, %options) = @_;
    # create an LDAP connection with given parameters
    $self->{_ldap} = ...
}

sub get_maintainer {
    my ($self, $package) = @_;
    # retrieve maintainer from LDAP
    my $maintainer = ...
    return $maintainer;
}

1;
```

Remplacer une fonctionnalité

```
package Whatever::LDAP;

=head1 NAME

Whatever::LDAP - Use LDAP to resolve maintainers

=cut

use warnings;
use strict;
use Net::LDAP;
use Youri::Package;
use base 'Youri::Check::Maintainer::Resolver';

sub _init {
    my ($self, %options) = @_;
    # create an LDAP connection with given parameters
    $self->{_ldap} = ...
}

sub get_maintainer {
    my ($self, $package) = @_;
    # retrieve maintainer from LDAP
    my $maintainer = ...
    return $maintainer;
}

1;
```

Modifier une fonctionnalité existante

```
package Whatever::Conflicts;

=head1 NAME

Whatever::Conflicts - Check file conflicts with exception

=cut

use warnings;
use strict;
use base 'Youri::Check::Input::Conflicts';

sub _directory_duplicate_exception {
    my ($self, $package1, $package2, $file) = @_;

    # allow shared modules directories between perl packages
    my $name = $file->get_name();
    return 1 if
        name =~ /^usrlibperl5vendor_perl/o
        && $name !~ /^(auto|[^]+-linux)$/o;

    return 0;
}

1;
```

Modifier une fonctionnalité existante

```
package Whatever::Conflicts;

=head1 NAME

Whatever::Conflicts - Check file conflicts with exception

=cut

use warnings;
use strict;
use base 'Youri::Check::Input::Conflicts';

sub _directory_duplicate_exception {
    my ($self, $package1, $package2, $file) = @_;

    # allow shared modules directories between perl packages
    my $name = $file->get_name();
    return 1 if
        name =~ /^usrlibperl5vendor_perl/o
        && $name !~ /^(auto|[^]+-linux)$/o;

    return 0;
}

1;
```

Modifier une fonctionnalité existante

```
package Whatever::Conflicts;

=head1 NAME

Whatever::Conflicts - Check file conflicts with exception

=cut

use warnings;
use strict;
use base 'Youri::Check::Input::Conflicts';

sub _directory_duplicate_exception {
    my ($self, $package1, $package2, $file) = @_;

    # allow shared modules directories between perl packages
    my $name = $file->get_name();
    return 1 if
        name =~ /^usrlibperl5vendor_perl/o
        && $name !~ /^(auto|[^]+-linux)$/o;

    return 0;
}

1;
```

Modifier une fonctionnalité existante

```
package Whatever::Conflicts;

=head1 NAME

Whatever::Conflicts - Check file conflicts with exception

=cut

use warnings;
use strict;
use base 'Youri::Check::Input::Conflicts';

sub _directory_duplicate_exception {
    my ($self, $package1, $package2, $file) = @_;

    # allow shared modules directories between perl packages
    my $name = $file->get_name();
    return 1 if
        name =~ /^usrlibperl5vendor_perl/o
        && $name !~ /^(auto|[^]+-linux)$/o;

    return 0;
}

1;
```

Modifier une fonctionnalité existante

```
package Whatever::Conflicts;

=head1 NAME

Whatever::Conflicts - Check file conflicts with exception

=cut

use warnings;
use strict;
use base 'Youri::Check::Input::Conflicts';

sub _directory_duplicate_exception {
    my ($self, $package1, $package2, $file) = @_;

    # allow shared modules directories between perl packages
    my $name = $file->get_name();
    return 1 if
        name =~ /^usrlibperl5vendor_perl/o
        && $name !~ /^(auto|[^]+-linux)$/o;

    return 0;
}

1;
```

Avancement

En cours...

Plein d'idées

- support format .deb
- interfaces web
  - reports dynamiques
  - formulaire soumission
- plus de tests
  - faux packages
  - vrai packages générés

Avancement

Perl

Bon choix pour une boite à outils

- large couverture du CPAN
- facilités d'analyse de texte
- liaisons natives avec les systeme de gestion de paquetages

Néanmoins il y a également des limitations...

Perl

Bon choix pour une boite à outils

- large couverture du CPAN
- facilités d'analyse de texte
- liaisons natives avec les systeme de gestion de paquetages

Néanmoins il y a également des limitations...

Lisibilité

Les languages impératifs se prêtent mal à l'algorithmie :

```
for each package:
    for each file:
        index package, file type and file md5 by file name

for each package:
    for each file:
        for each other package sharing file:
            check packages differ
            check packages compatibility
            if file is a directory:
                produce warning
            else
                if file md5 differs:
                    produce error
                else
                    produce warning
```

Performances

Stocker beaucoup de chaines en mémoire est couteux.

Indexation de l'ensemble des fichiers

- nombre de paquetages
- $\times$ nombre d'architectures
- $\times$ nombre moyen de fichier par paquetage
- $\times$ nombre d'information par fichier

Solutions

Utilisation d'outils dédiés

- rpmlint
- rpmcheck

Utilisation d'autres languages

- SQL
- C

Solutions

Utilisation d'outils dédiés

- rpmlint
- rpmcheck

Utilisation d'autres languages

- SQL
- C

Avancement

Paquetage

Plusieurs interprétations

- granularité amont
  - `libxml2`
- granularité aval
  - `libxml2`
  - `libxml2-devel`
- fichiers
  - `libxml2-2.6.26-1mdk.i586.rpm`
  - `libxml2-2.6.26-1mdk.x86_64.rpm`
  - `libxml2-devel-2.6.26-1mdk.i586.rpm`
  - `lib64xml2-devel-2.6.26-1mdk.x86_64.rpm`
  - `libxml2-2.6.26-1mdk.src.rpm`

Media

Plusieurs synonymes

- media pour urpmi
- channel pour smart
- source pour apt

Plusieurs interprétations

- unité d'organisation :
  - `main`
- unité de distribution :
  - `main-source`
  - `main-binary`

Media

Plusieurs synonymes

- media pour urpmi
- channel pour smart
- source pour apt

Plusieurs interprétations

- unité d'organisation :
  - `main`
- unité de distribution :
  - `main-source`
  - `main-binary`

Version et release

Relation d'ordre pour RPM

1. epoch
2. version
3. release

Sens des mots 'version' et 'release'

- valeur globale d'ordre
- valeur atomique du même nom

Version et release

Relation d'ordre pour RPM

1. epoch
2. version
3. release

Sens des mots 'version' et 'release'

- valeur globale d'ordre
- valeur atomique du même nom

Avancement

Trouver les problèmes

Différentes échelles de test

- fichier : rpmlint
- dépôt : youri-check
- paquetage ?

Différentes modalité de test

- obligatoire : au moment de la soumission
- optionelle : après la soumission

Trouver les problèmes

Différentes échelles de test

- fichier : rpmlint
- dépôt : youri-check
- paquetage ?

Différentes modalité de test

- obligatoire : au moment de la soumission
- optionelle : après la soumission

Évaluer les problèmes

Évaluation de la gravité

- avertissement
- erreur
- ...

Évaluation de la certitude

- faux positifs
- faux négatifs

Évaluer les problèmes

Évaluation de la gravité

- avertissement
- erreur
- ...

Évaluation de la certitude

- faux positifs
- faux négatifs

Faire corriger les problèmes

Motivation des mainteneurs

- éviter les fausses alertes
- approche volontariste (opt-in)
- maximiser les rapports
  - en clareté
  - en configurabilité
- gratifier ceux qui jouent le jeu

Plus de détails

- Site web du projet : http ://youri.zarb.org
- Démonstration en ligne : http ://youri.zarb.org/demo
- Utilisation réelle : http ://plf.zarb.org/qa