Note: This tutorial assumes that you have completed the previous tutorials: Naviguer dans le système de fichier ROS (/fr/ROS/Tutorials/NavigatingTheFilesystem).

Please ask about problems and questions regarding this tutorial on answers.ros.org (http://answers.ros.org). Don't forget to include in your question the link to this page, the versions of your OS & ROS, and also add appropriate tags.

Créer un package ROS

Description: Ce tutoriel présente l'utilisation de roscreate-pkg (/roscreate) ou catkin (/catkin) pour créer un nouveau package et rospack (/rospack) pour lister les dépendances des packages.

Tutorial Level: BEGINNER

Next Tutorial: Construire un package ROS (/fr/ROS/Tutorials/BuildingPackages)

catkin	rosbuild

Qu'est ce qu'un package catkin?

Pour qu'un package puisse être considéré comme un package catkin il doit répondre à quelques exigences:

- Le package doit contenir un fichier package.xml conforme à catkin.
  - Le fichier package.xml fournit les meta-informations sur le package.
- Le package doit contenir aussi un fichier CMakeLists.txt quiappelle catkin (/catkin/CMakeLists.txt).
  - Pour les metapackage catkin (/catkin/package.xml#Metapackages) il et nécessaire d'avoir tous les fichiers CMakeLists.txt nécessaires.
- Chaque package doit avoir son propre répertoire (Il ne peut y avoir plus d'un package dans chaque dossier)
  - Cela signifie qu'il n'y a pas de packages imbriqués ni plusieurs packages partagent le même répertoire.

Le package le plus simple possible pourrait ressembler à ceci:

```
my_package/
CMakeLists.txt
package.xml
```

Les packages dans un workspace catkin

Pour travailler avec des packages dans catkin, la méthode recommandée consiste à travailler dans un workspace (espace de travail) catkin, mais il est aussi possible de construire des packages catkin à part (en standalone).

Un workspace typique pourrait ressembler à ceci:

```
workspace_folder/        -- WORKSPACE
src/                    -- SOURCE SPACE
CMakeLists.txt       -- 'Toplevel' CMake file, provided by catkin
package_1/
CMakeLists.txt     -- CMakeLists.txt file for package_1
package.xml        -- Package manifest for package_1
...
package_n/
CMakeLists.txt     -- CMakeLists.txt file for package_n
package.xml        -- Package manifest for package_n
```

Avant de continuer avec ce tutoriel vous devez créer un workspace catkin vide en suivant ce tutoriel Creating a workspace for catkin (http://wiki.ros.org/catkin/Tutorials/create_a_workspace)

Création d'un package catkin

Ce tutorial montre comment utiliser le script catkin_create_pkg (http://wiki.ros.org/catkin/commands/catkin_create_pkg) pour créer un nouveau package catkin, et ce que vous pouvez faire avec, une fois créé.

D'abord on va se rendre dans le répertoire des fichiers sources du workspace catkin que l'on a créé lors du tutorial Creating a workspace for catkin (http://wiki.ros.org/catkin/Tutorials/create_a_workspace)

```
# You should have created this in the Creating a Workspace Tutorial
cd ~/catkin_ws/src
```

Maintenant, utilisons le script catkin_create_pkg pour créer un nouveau package appelé "beginner_tutorials" qui dépend de std_msgs, roscpp et Rospy:

```
catkin_create_pkg beginner_tutorials std_msgs rospy roscpp
```

Cela va créer un dossier beginner_tutorials qui contient un package.xml et un CMakeLists.txt , qui ont été partiellement remplis avec les informations que vous avez données à catkin_create_pkg. catkin_create_pkg demande que vous lui donniez un nom de package et éventuellement une liste

des packages dont ce package dépend:

```
# This is an example, do not try to run this
# catkin_create_pkg <package_name> [depend1] [depend2] [depend3]
```

catkin_create_pkg possède également des fonctionnalités plus avancées qui sont décrites dans le catkin/commands/catkin_create_pkg (http://wiki.ros.org/catkin/commands/catkin_create_pkg) .

Dépendances du package

1. Dépendances de premier ordre

Lors de l'utilisation de catkin_create_pkg plus tôt, quelques dépendances des packages ont été fournies. Ces dépendances de premier ordre peuvent maintenant être examinées avec l'outil rospack.

```
```

(9 janvier 2013) Il y a http://answers.ros.org/question/51555/beginner-tutorials-segmentation-fault-with-rospack-depends1/?comment=51762#comment-51762 (http://answers.ros.org/question/51555/beginner-tutorials-segmentation-fault-with-rospack-depends1/?comment=51762#comment-51762) avec la commande suivante, vous pouvez passer à la prochaine commande.

```
rospack depends1 beginner_tutorials
```

```
std_msgs
rospy
roscpp
```

Comme vous pouvez le voir, rospack énumère les mêmes dépendances qui ont été utilisées comme arguments lors de l'exécution catkin_create_pkg. Les dépendances d'un package sont stockées dans le fichier **package.xml** :

```
roscd beginner_tutorials
cat package.xml
```

Afficher/masquer les numéros de lignes

```
1 <package>
2 ...
3 <buildtool_depend>catkin</buildtool_depend>
4 <build_depend>roscpp</build_depend>
5 <build_depend>rospy</build_depend>
6 <build_depend>std_msgs</build_depend>
7 ...
8 </package>
```

Dépendances indirectes

Dans de nombreux cas, une dépendance aura aussi ses propres dépendances. Par exemple, Rospy a d'autres dépendances.

```
(9 janvier 2013) Il y a  [un bug](https://github.com/ros/rospack/issue
s/4) signalé et déjà fixé dans  [rospack](http://wiki.ros.org/rospack)
en groovy, ce qui prend un certain temps jusqu'à ce que le changement
 se reflète sur votre ordinateur. Si vous voyez  [un problème semblabl
e à celui-ci](http://answers.ros.org/question/51555/beginner-tutorials
-segmentation-fault-with-rospack-depends1/?comment=51762#comment-5176
2) avec la commande suivante, vous pouvez passer à la prochaine comman
de.
```

```
rospack depends1 rospy
```

```
genpy
rosgraph
rosgraph_msgs
roslib
std_msgs
```

Un package peut avoir pas mal de dépendances indirectes. Heureusement rospack peut déterminer de façon récursive toutes les dépendances imbriquées.

```
rospack depends beginner_tutorials
```

```
cpp_common
rostime
roscpp_traits
roscpp_serialization
genmsg
genpy
message_runtime
rosconsole
std_msgs
rosgraph_msgs
xmlrpcpp
roscpp
rosgraph
catkin
rospack
roslib
rospy
```

Personnalisation de votre Package

Dans cette partie du tutoriel on va regarder dans chaque fichier généré par catkin_create_pkg et décrire, ligne par ligne, chaque composante de ces fichiers et comment vous pouvez les personnaliser pour votre package.

1. Personnalisation du package.xml

Le fichier généré package.xml devrait déjà être dans votre nouveau package. Maintenant allons dans le fichier package.xml et regardons chaque élément.

1.1 Balise de description

Première ligne, la balise description :

Afficher/masquer les numéros de lignes

```
1 <description>The beginner_tutorials package</description>
```

Modifier la description comme vous voulez, mais, par convention, la première phrase doit être courte tout en couvrant le champ d'application du package. S'il est difficile de décrire le package en une seule phrase, alors il faudra peut-être la couper.

1.2 La balises du responsable

Vient ensuite le maintainer tag :

Afficher/masquer les numéros de lignes

```
   1 <!-- One maintainer tag required, multiple allowed, one person p
er tag -->
   2 <!-- Example:  -->
   3 <!-- <maintainer email="jane.doe@example.com">Jane Doe</maintain
er> -->
   4 <maintainer email="user@todo.todo">user</maintainer>
```

Il s'agit d'une balise requise et importante pour le fichier package.xml , car il permet aux autres de savoir qui contacter au sujet du package. Au moins un nom de mainteneur est requis, mais vous pouvez en avoir plus si vous voulez. Le nom du responsable va dans le corps de la balise, mais il y a aussi l'attribut email qui doit être rempli:

Afficher/masquer les numéros de lignes

```
   1 <maintainer email="you@yourdomain.tld">Your Name</maintainer>
```

1.3 La balise licence

La balise licence est également requise :

Afficher/masquer les numéros de lignes

```
   1 <!-- One license tag required, multiple allowed, one license per
tag -->
   2 <!-- Commonly used license strings: -->
   3 <!--   BSD, MIT, Boost Software License, GPLv2, GPLv3, LGPLv2.1,
LGPLv3 -->
   4 <license>TODO</license>
```

Vous devez choisir une licence et l'indiquer ici. Les licences open source les plus communes sont : BSD, MIT, Boost Software License, GPLv2, GPLv3, LGPLv2.1 et LGPLv3. Vous pouvez en apprendre plus sur la plupart d'entre elles à l' Open Source Initiative (http://opensource.org/licenses/alphabetical) . Pour ce tutoriel, nous allons utiliser la licence BSD parce que la majorité des autres composants ROS l'utilisent déjà:

Afficher/masquer les numéros de lignes

```
   1 <license>BSD</license>
```

1.4 Les balises de dépendances

La prochaine série de balises décrit les dépendances de votre package. Les dépendances sont divisées en 4 catégories :

- build_depend, (dépendances pour la construction)
- buildtool_depend, (dépendances pour les outils de construction)
- run_depend, (dépendances pour l'exécution)
- test_depend. (dépendances pour le test)

Pour une explication plus détaillée de ces balises consultez la documentation sur les dépendances catkin Catkin Dependencies (http://wiki.ros.org/catkin/package.xml#Build.2C_Run.2C_and_Test_Dependencies)

Comme nous avons passé std_msgs, roscpp, et rospy comme arguments à catkin_create_pkg, la déclaration des dépendances ressemblera à ceci:

Afficher/masquer les numéros de lignes

```
 1 <!-- The *_depend tags are used to specify dependencies -->
 2 <!-- Dependencies can be catkin packages or system dependencies
-->
 3 <!-- Examples: -->
 4 <!-- Use build_depend for packages you need at compile time: -->
 5 <!--   <build_depend>genmsg</build_depend> -->
 6 <!-- Use buildtool_depend for build tool packages: -->
 7 <!--   <buildtool_depend>catkin</buildtool_depend> -->
 8 <!-- Use run_depend for packages you need at runtime: -->
 9 <!--   <run_depend>python-yaml</run_depend> -->
10 <!-- Use test_depend for packages you need only for testing: -->
11 <!--   <test_depend>gtest</test_depend> -->
12 <buildtool_depend>catkin</buildtool_depend>
13 <build_depend>roscpp</build_depend>
14 <build_depend>rospy</build_depend>
15 <build_depend>std_msgs</build_depend>
```

Toutes les dépendances que l'on avait déclarées à la création de notre package ont été ajoutées pour nous en tant que build_depend, en plus de la déclaration par défaut de buildtool_depend pour catkin (c'est le compilateur on est donc bien obligé de l'avoir. Pour les versions antérieures à la groovy le compilateur se nomme rosmake). Dans ce cas, nous voulons que toutes les dépendances spécifiées soient accessibles au moment de la construction et au moment de l'exécution, nous allons ajouter un tag run_depend pour chacun d'eux ainsi (ndt : il s'avère que pour la groovy les dépendances d'exécution sont déjà présentes) :

Afficher/masquer les numéros de lignes

```
1 <buildtool_depend>catkin</buildtool_depend>
2 <build_depend>roscpp</build_depend>
3 <build_depend>rospy</build_depend>
4 <build_depend>std_msgs</build_depend>
5 <run_depend>roscpp</run_depend>
6 <run_depend>rospy</run_depend>
7 <run_depend>std_msgs</run_depend>
```

1.5 Le package.xml final

Comme vous pouvez le voir le package.xml final, sans commentaires et sans les tags inutilisés, est beaucoup plus concis:

Afficher/masquer les numéros de lignes

```
1 <?xml version="1.0"?>
2 <package>
3 <name>beginner_tutorials</name>
4 <version>0.1.0</version>
5 <description>The beginner_tutorials package</description>
6 <maintainer email="you@yourdomain.tld">Your Name</maintainer>
7 <license>BSD</license>
8 <url type="website">http://wiki.ros.org/beginner_tutorials</url>
9 <author email="you@yourdomain.tld">Jane Doe</author>
10 <buildtool_depend>catkin</buildtool_depend>
11 <build_depend>roscpp</build_depend>
12 <build_depend>rospy</build_depend>
13 <build_depend>std_msgs</build_depend>
14 <run_depend>roscpp</run_depend>
15 <run_depend>rospy</run_depend>
16 <run_depend>std_msgs</run_depend>
17 </package>
```

2. Personnalisation du CMakeLists.txt

Maintenant que le package.xml , qui contient des informations meta, a été adapté à votre package, vous êtes prêt à passer à la suite des tutoriels. Le fichier CMakeLists.txt créé par catkin_create_pkg sera abordé plus tard dans les didacticiels où vous écrirez du code ROS.

Maintenant que vous avez fait un nouveau package ROS, allons construire notre package ROS (/fr/ROS/Tutorials/BuildingPackages).

 Wiki: fr/ROS/Tutorials/CreatingPackage (dernière édition le 2019-11-03 21:55:33 par fangorn_fr (/fangorn_fr))

Brought to you by:  Open Robotics

(https://www.openrobotics.org/)