

Omcricp : Train DCC

Ce guide montre les étapes de construction de l'extension omcri pour les trains DCC de la plateforme portuaire.

Extension sous eclipse

Les étapes suivantes permettent de créer l'extension pour les trains DCC.

1. Créer une nouvelle extension sous omcri studio

Ouvrir OMCRI Studio et dans *File->New->OCCI Studio* sélectionner **OCCI Extension Project**

Renseigner les informations suivantes

- Project Name : *omcri4cp.train*
- OCCI Extension name : *Train*
- OCCI Extension scheme : *http://cristal.org/omcri4cp/train#*

Dans la sections *Referenced extensions*, cocher la case *http://occiware.org/cloudrobotics/core#*

Create an OCCI Extension project

Project name must be specified



Project name:

☒ Use default location

Location:

OCCI Extension name:

OCCI Extension scheme:

Referenced extensions:

- ☐ <http://occiware.org/cloudrobotics/all#>
- ☐ <http://occiware.org/cloudrobotics/turtlebot#>
- ☐ <http://occiware.org/cloudrobotics/core#>
- ☐ <http://occiware.org/cloudrobotics/parrotminidrone#>
- ☐ <http://schemas.opengis.net/occi/infrastructure/compute/templ...>
- ☐ <http://occiware.org/cloudrobotics/mindstormev3#>
- ☐ <http://occiware.org/cloudrobotics/parrotardrone#>
- ☐ <http://occiware.org/cloudrobotics/mindstorm#>
- ☐ <http://schemas.opengis.net/occi/infrastructure#>
- ☐ <http://schemas.opengis.net/occi/monitoring#>

2. Ajouter un kind qui hérite de robot en spécifiant les attributs et les actions.

- Ajouter un kind au diagramme en faisant un glisser déposer depuis category.
- Dans les propriétés du Kind, changer le *name* en *Train*.
- Ajouter une relation parent entre **Train** et **Robot**.

- Les attributs à ajouter sont :

Les actions en implémenter sont :

On obtient le diagramme ci-dessous :



3. Générer les différents classes

Clique droit sur train.occie (*omcri4cp.train->model->train.occie*); dans OCCI-Studio, sélectionner **Generate EMF Tooling**.

Les classes générées sont dans le dossier source **src-gen**.

4. Générer la documentation si nécessaire

Clique droit sur train.occie (*omcri4cp.train->model->train.occie*); dans OCCI-Studio, sélectionner Generate **Generate Latex document** ou **Generate Textile documentation**.

5. Créer un connecteur java

Dans *File->New->OCCI Studio*, sélectionner **Java Connector Project**

Renseigner le nom du projet ; Project name : *omcri4cp.train.connector*. Ensuite sélectionner l'OCCI Extension correspondante : <http://cristal.org/omcri4cp/train#>.

Create a Java Connector project

Enter a project name



Project name:

☒ Use default location

Location:

Select an OCCI extension:

- ☐ <http://occiware.org/cloudrobotics/all#>
- ☐ <http://occiware.org/cloudrobotics/turtlebot3#>
- ☐ <http://occiware.org/cloudrobotics/core#>
- ☐ <http://occiware.org/cloudrobotics/parrot#>
- ☐ <http://schemas.opengis.net/occi/infrastructure#>
- ☐ <http://occiware.org/cloudrobotics/mindstorms#>
- ☐ <http://occiware.org/cloudrobotics/parrot#>
- ☐ <http://occiware.org/cloudrobotics/mindstorms#>
- ☐ <http://schemas.opengis.net/occi/infrastructure#>
- ☐ <http://schemas.opengis.net/occi/monitoring#>

6. Importer le .jar dans le connecteur

- Créer un dossier lib à la racine du connecteur java
- Copier coller le .jar dans ce dossier

7. Déclarer le .jar dans le MANIFEST

Dans le dossier **META-INF** du connecteur, ouvrir le fichier MANIFEST.MF. Aller dans **Runtime**, dans la section **classpath**, cliquer sur **Add**. Dans la fenêtre qui s'affiche, sélectionner le .jar dans le dossier lib créé précédemment puis appuyer sur ok.

8. Implémenter les actions dans la classe train

Dans le dossier source **src-gen**, ouvrir le fichier ***TrainConnector.java*** et le compléter.

Client TCP

Le client est écrit en java et sera utiliser par le connecteur de l'extension sous omcri.

Code source

Serveur TCP

Le serveur est écrit en python et doit être lancé sur la raspberry pi connectée au contrôleur des trains DCC.

Code source

Installation des bibliothèques

Les biblithèques **speedlib** et **dccpi** sont nécessaires pour faire fonctionner correctement le serveur.

Installation de speedlib :

```
pip3 install speedlib
```

Installation de dccpi :

```
pip3 install dccpi
```

Test sous eclipse

Générer un design

Lancer le .design

Test avec MartServer

Déclarer les .jar comme dépendences pour MartServer.

Lancer MartServer avec OCCInterface