



AUTO-FORMATION BUILDSYSPRO

2. DYMOLA ET MODELICA

Support de formation de BuildSysPro
Dernière révision : janvier 2016

EDF R&D
Département Enerbat (Energie dans les Bâtiments et les Territoires)
Groupe « Simulation énergétique et bâti »





2. ENVIRONNEMENT DYMOLA / LANGAGE MODELICA

A. Introduction

- Différence Dymola / Modelica
- Des équations aux résultats numériques
- Dymola et ses applications ...

B. Onglet 'Modeling'

- Contenu de l'onglet 'Modeling'
- Gestion des modèles *avec démarrage du tutoriel*

C. Onglet 'Simulation'

- Contenu de l'onglet 'Simulation'
- Lancement d'une simulation
- Visualisation & export des résultats

2. ENVIRONNEMENT DYMOLA / LANGAGE MODELICA

D. Éléments du langage

- Les différentes classes dans Modelica
- Composition d'un modèle
- Tests et boucles
- Définition des types
- Définition d'un connecteur
- Héritage
- Gestion d'évènements
- Manipulation de vecteurs et matrices

E. Exercices d'application

- Création d'une paroi R2C1 et comportement à une excitation sinusoïdale

Dymola Multi-Engineering Modeling and Simulation



Extrait de <https://modelica.org>

model Capacitor "Ideal linear electrical capacitor"
parameter SI.Capacitance C "Capacitance";
Interfaces.PositivePin p;
Interfaces.NegativePin n;
SI.Voltage v "Voltage drop between pins";
equation
0 = p.i + n.i;
v = p.v - n.v;
C*der(v) = p.i;
end Capacitor;

Modelica
User's Guide
Blocks
Mechanics
Fluid
Electrical
Analog
Examples
Basic
Ground
Resistor
Conductor
Capacitor
Inductor
SaturatingInductor
Transformer
M_Transformer
Gyrator

plug pin[1] plug

time [s]

MODELICA

Introduction ENVIRONNEMENT DYMOLA / LANGAGE MODELICA

DIFFÉRENCE DYMOLA / MODELICA

■ Modelica = Langage de

- Langage normalisé non propriétaire
- Langage orienté objet ⇒ notion d'héritage
- Développé pour répondre à des besoins de modélisation multi physiques
- Écriture formelle des équations de la physique
- Modélisation acausale
- Bibliothèques multi physiques standards
- Soutenu par une association : www.modelica.org



■ Dymola = Environnement de simulation

- Environnement propriétaire qui utilise le langage Modelica (distribué par Dassault Systèmes)
- Interface graphique unique pour la modélisation et la simulation
- Générateur de code : plusieurs solveurs & méthodes d'intégration à disposition
- Peut s'interfacer avec d'autres outils comme Matlab/Simulink



DES ÉQUATIONS AUX RÉSULTATS NUMÉRIQUES

Modélisation

Écriture du code Modelica grâce à l'éditeur graphique et textuel de Dymola

Vérification de la syntaxe grâce à la commande *Check* de Dymola

Compilation

Génération du système matriciel d'équations (DAE)

Optimisation du code avec réduction de la taille du système

Génération en C (modèles et solveur attaché)

Génération d'un exécutable (indépendant de Dymola)

Simulation

Création possible d'un script pour lancer des simulations de façon itérative

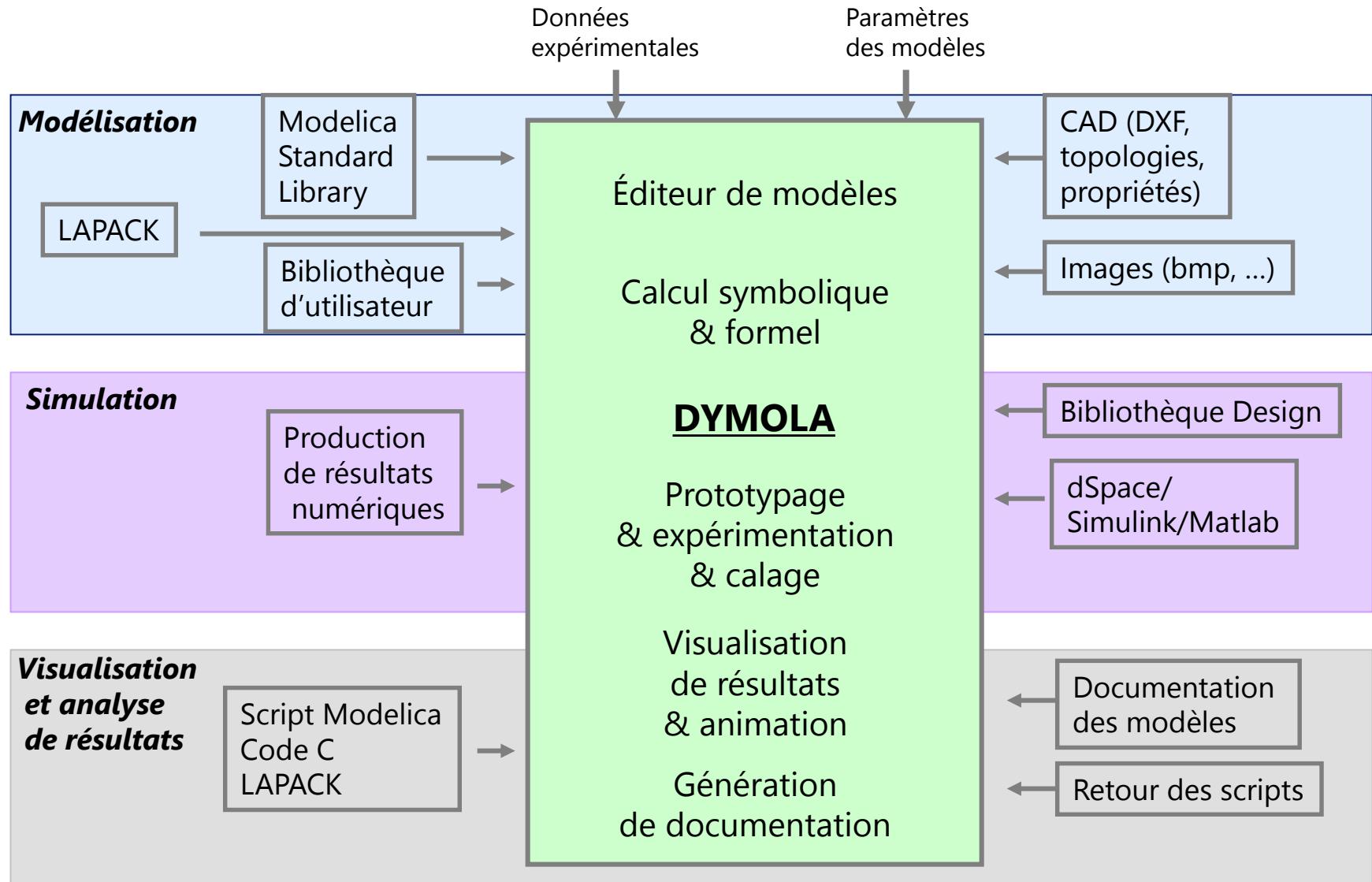
Lancement de l'exécutable ou du script

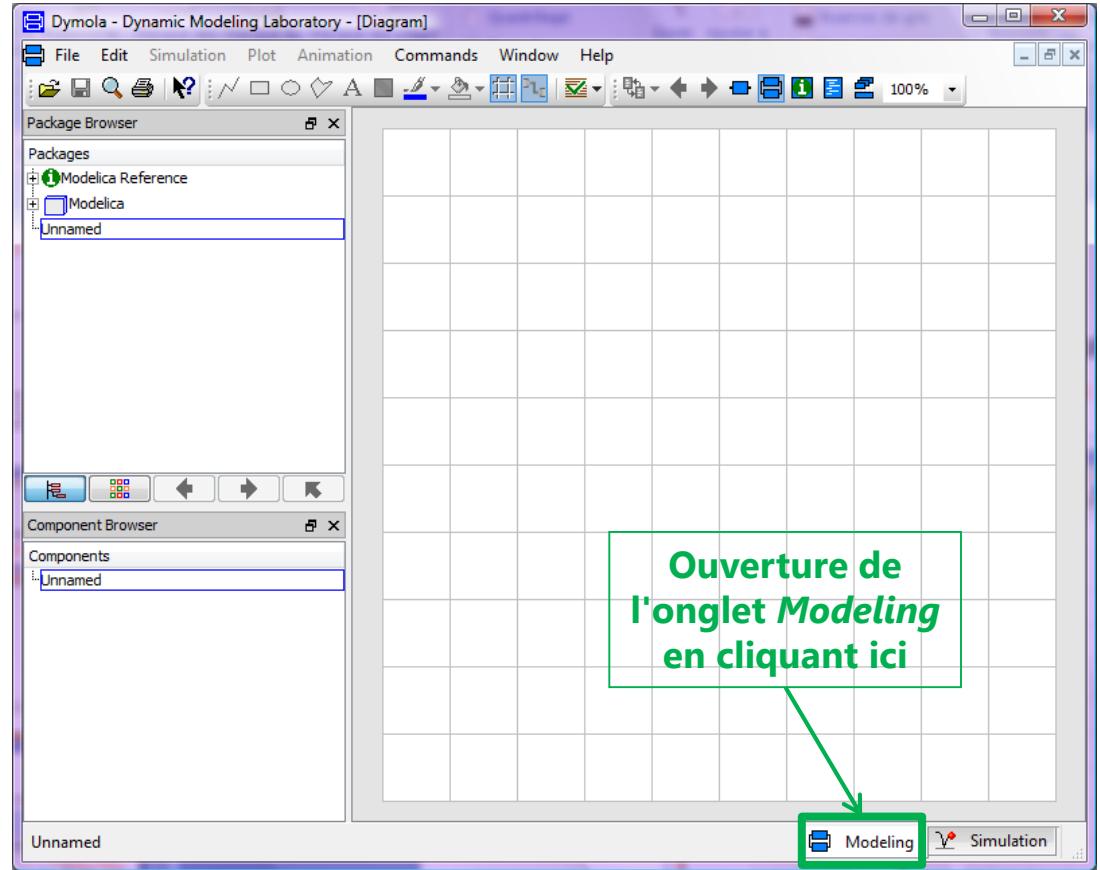
Post-traitement pour analyse

- Visualisation des résultats sous Dymola directement
- Et/ou export des résultats vers d'autres environnements (Matlab, Excel) (automatisable via le code)

Étapes automatiques lors de la commande *Translate Model*

DYMOLA ET SES APPLICATIONS ...

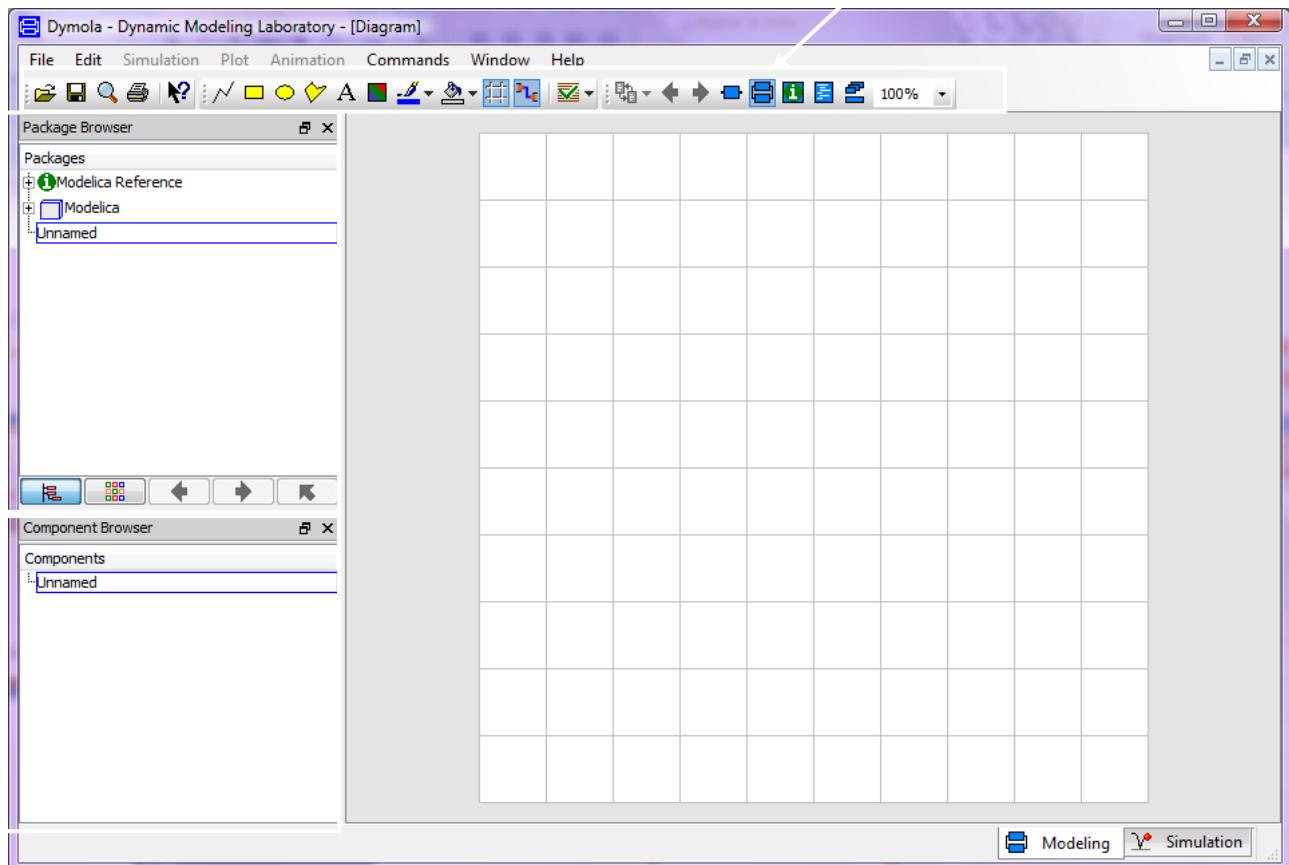




Onglet 'Modeling' ENVIRONNEMENT DYMOLA / LANGAGE MODELICA

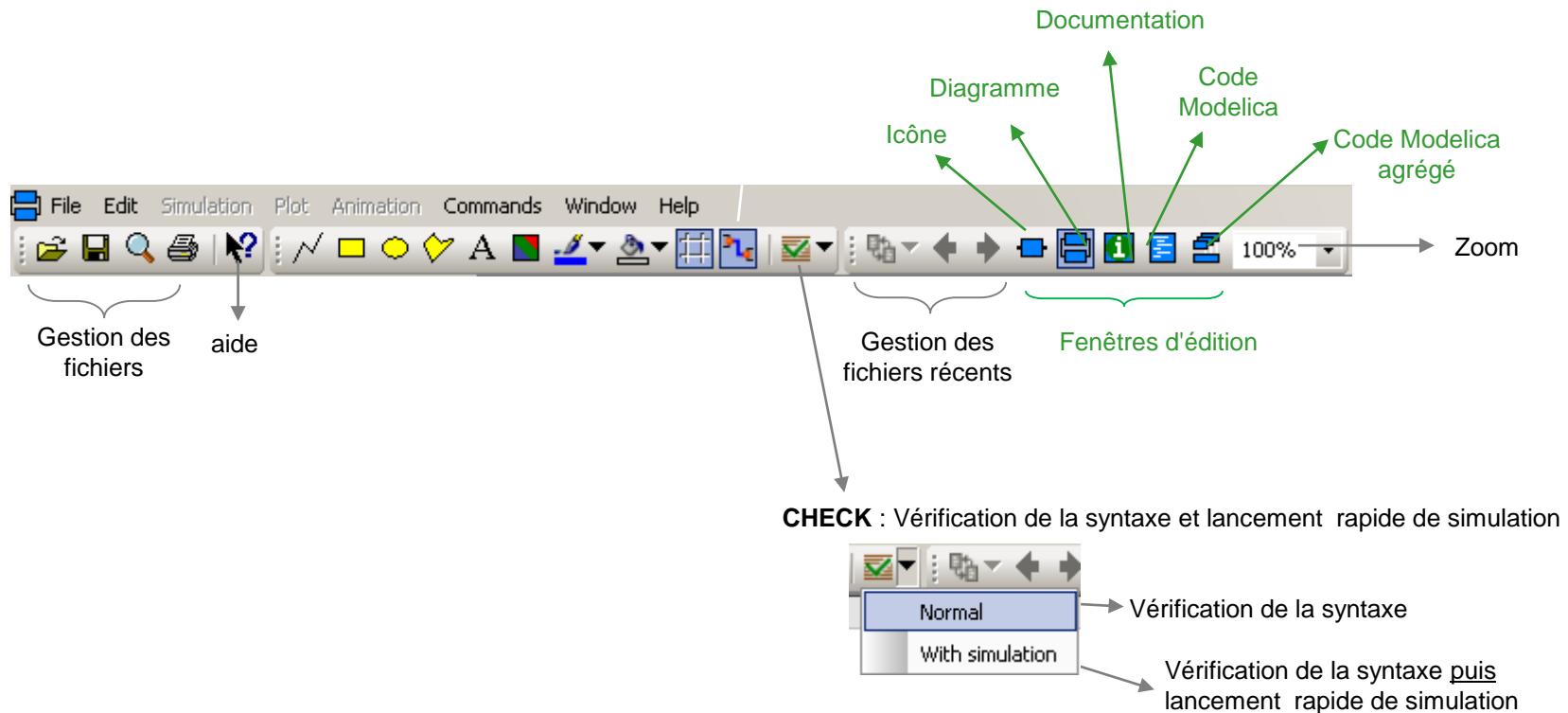
CONTENU DE L'ONGLET 'MODELING'

ASPECT GÉNÉRAL



CONTENU DE L'ONGLET 'MODELING'

BARRE D'OUTILS



GESTION DES MODÈLES

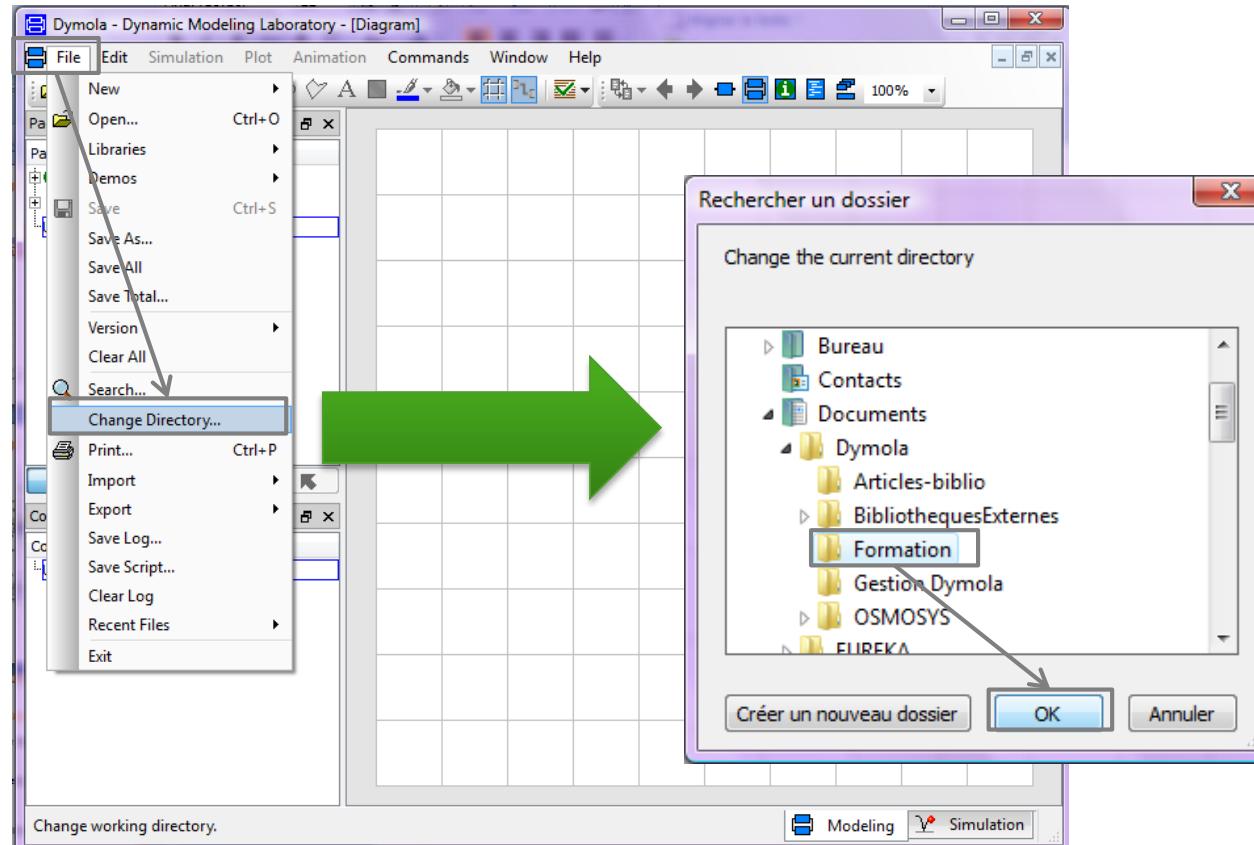
RÉPERTOIRE DE TRAVAIL

- Par défaut après l'installation le répertoire de travail est :
C:\Users\utilisateur\Documents\DM
- Les modèles créés lors de ce tutoriel seront stockés dans un dossier 'Formation'
 - **Créer sous Windows ce nouveau dossier 'Formation'**

GESTION DES MODÈLES

CHANGEMENT DU RÉPERTOIRE DE TRAVAIL

- Pour changer le répertoire de travail :

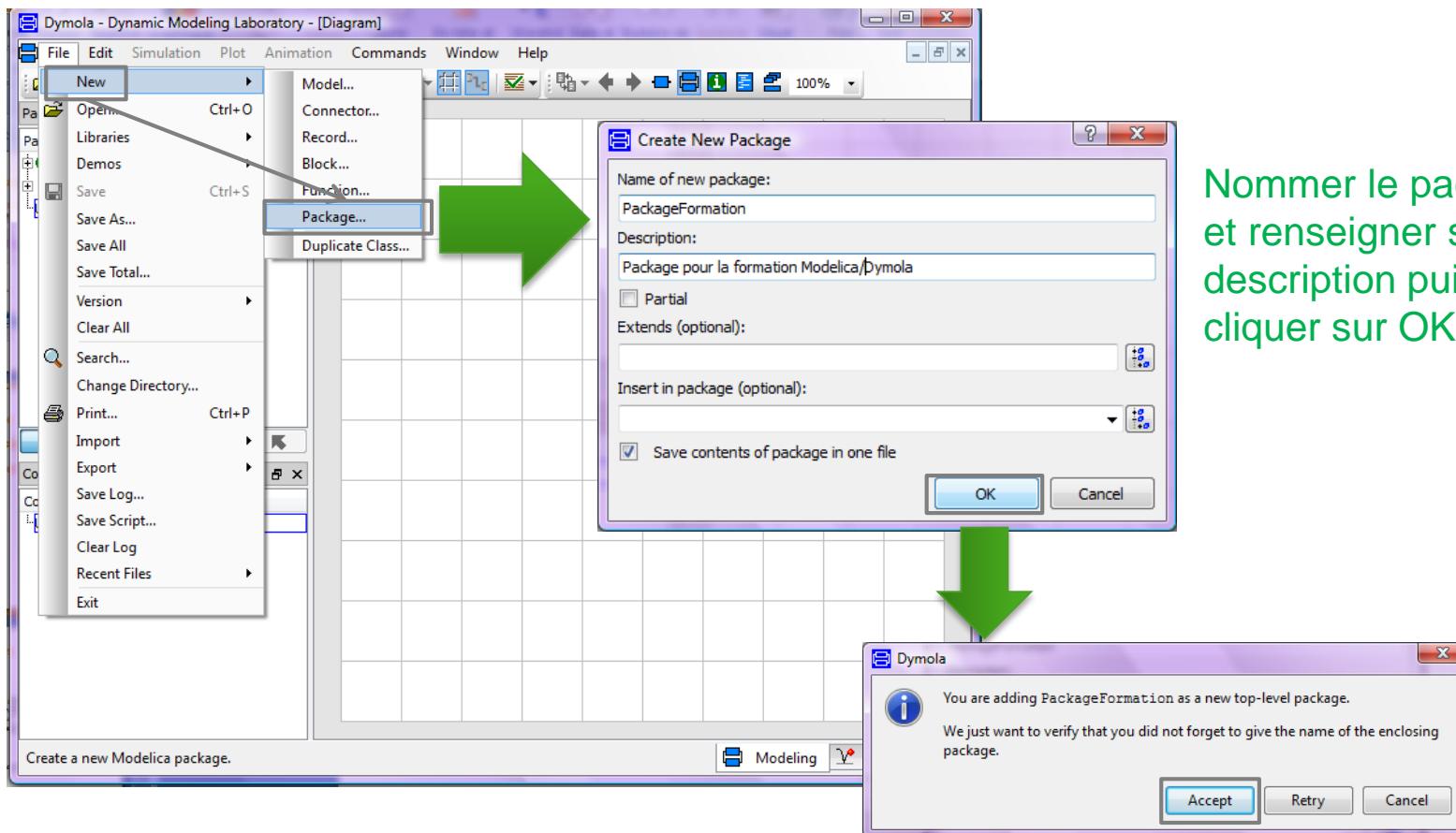


Sélectionner le nouveau dossier 'Formation' et cliquer sur OK

GESTION DES MODÈLES

CRÉATION D'UN NOUVEAU PACKAGE

- Voir '[Qu'est-ce qu'un package dans le langage Modelica ?](#)'
- Crédit d'un package :

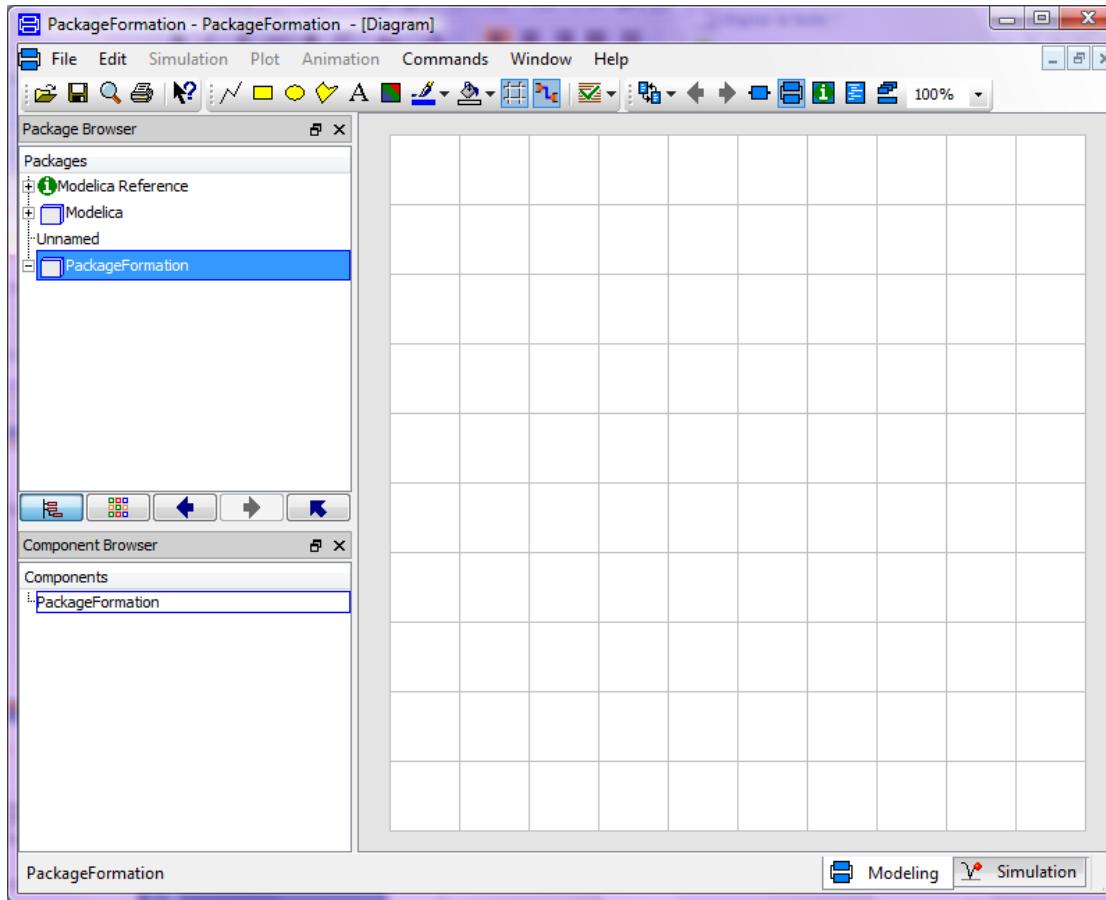


Nommer le package et renseigner sa description puis cliquer sur OK

GESTION DES MODÈLES

ENREGISTREMENT D'UN FICHIER

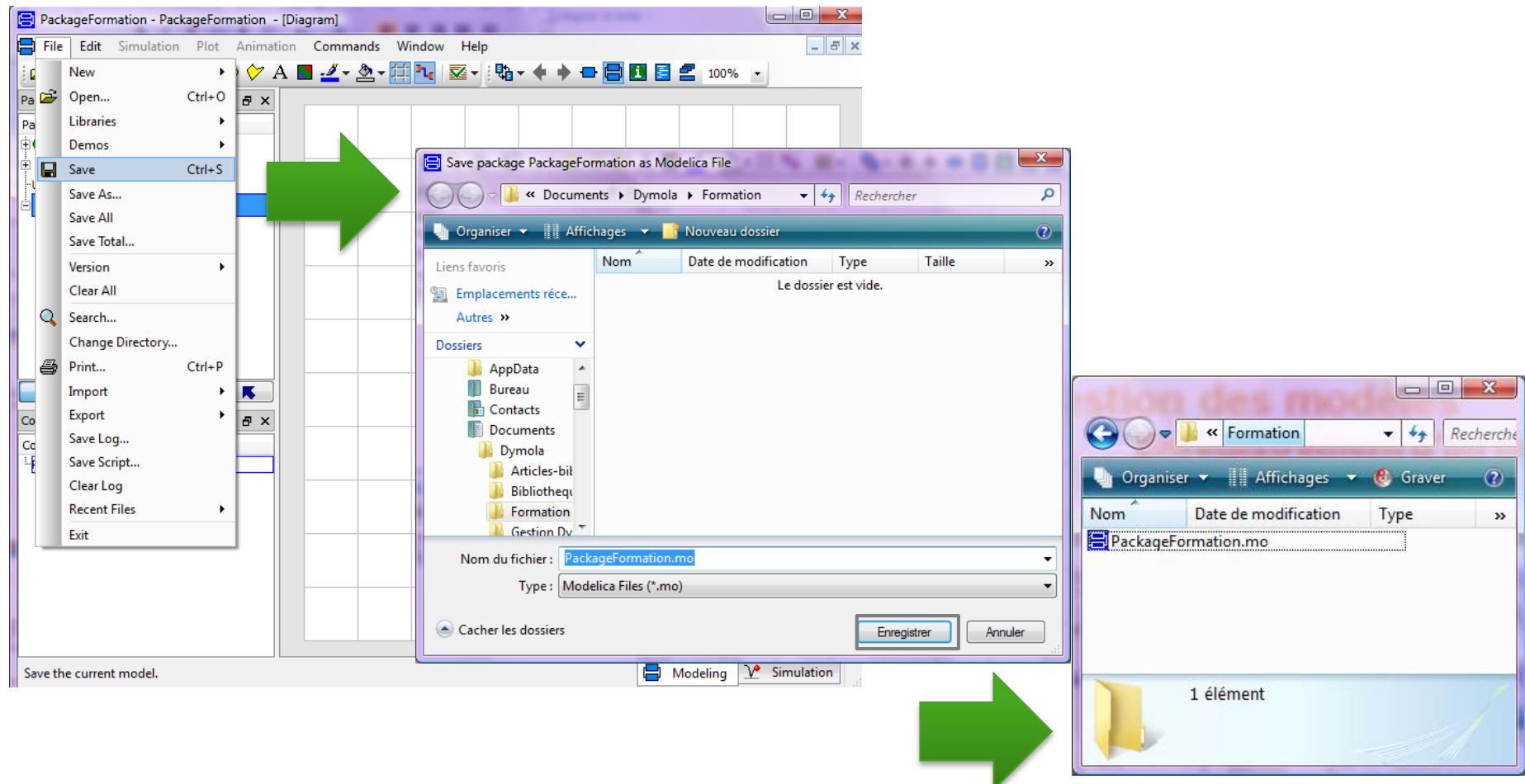
- Sélectionner le package par un double-clic sur celui-ci dans le Package Browser



GESTION DES MODÈLES

ENREGISTREMENT D'UN FICHIER

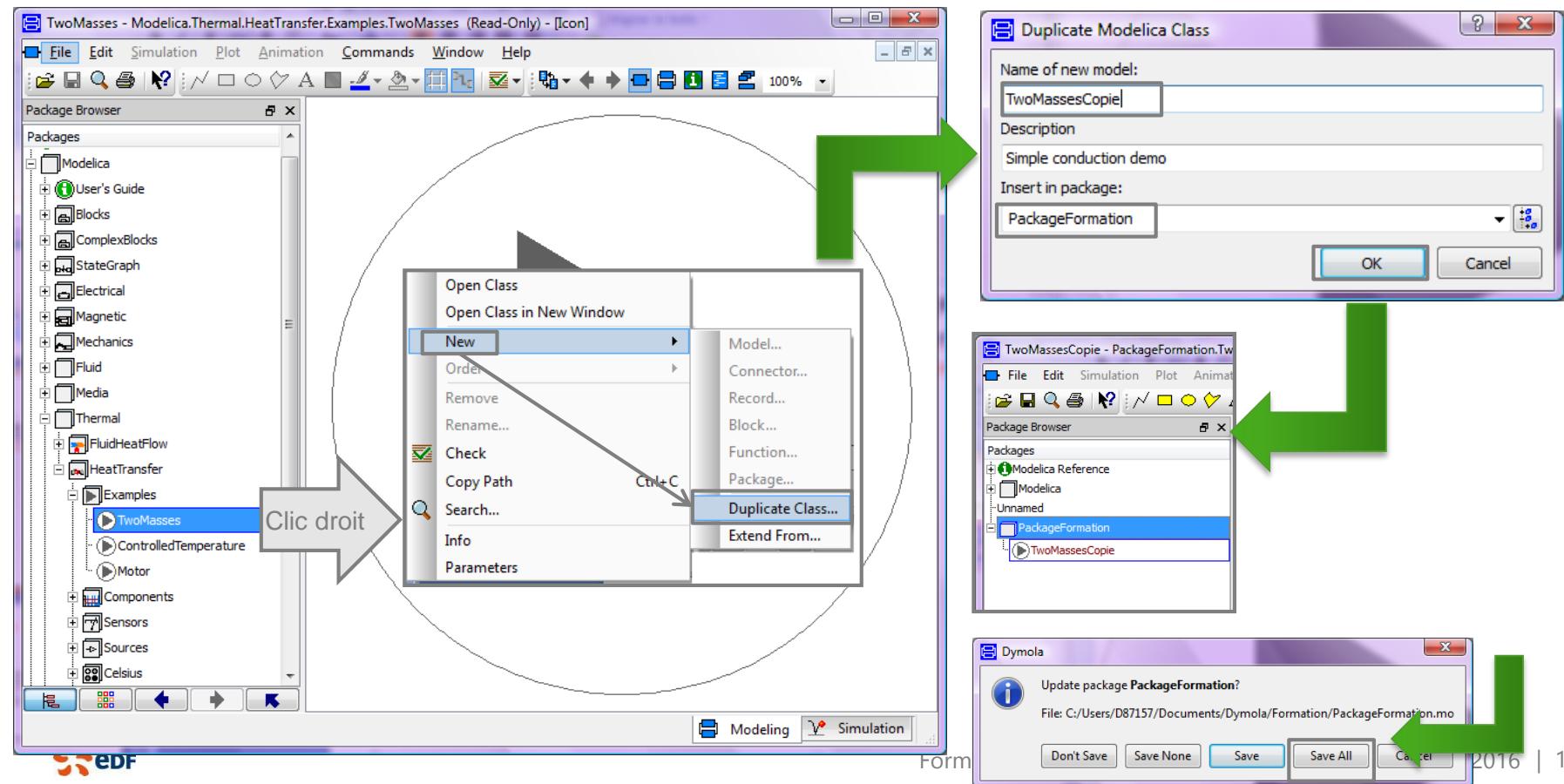
- Sauvegarder le package sélectionné : **Ctrl+s** ou **File\Save**



GESTION DES MODÈLES

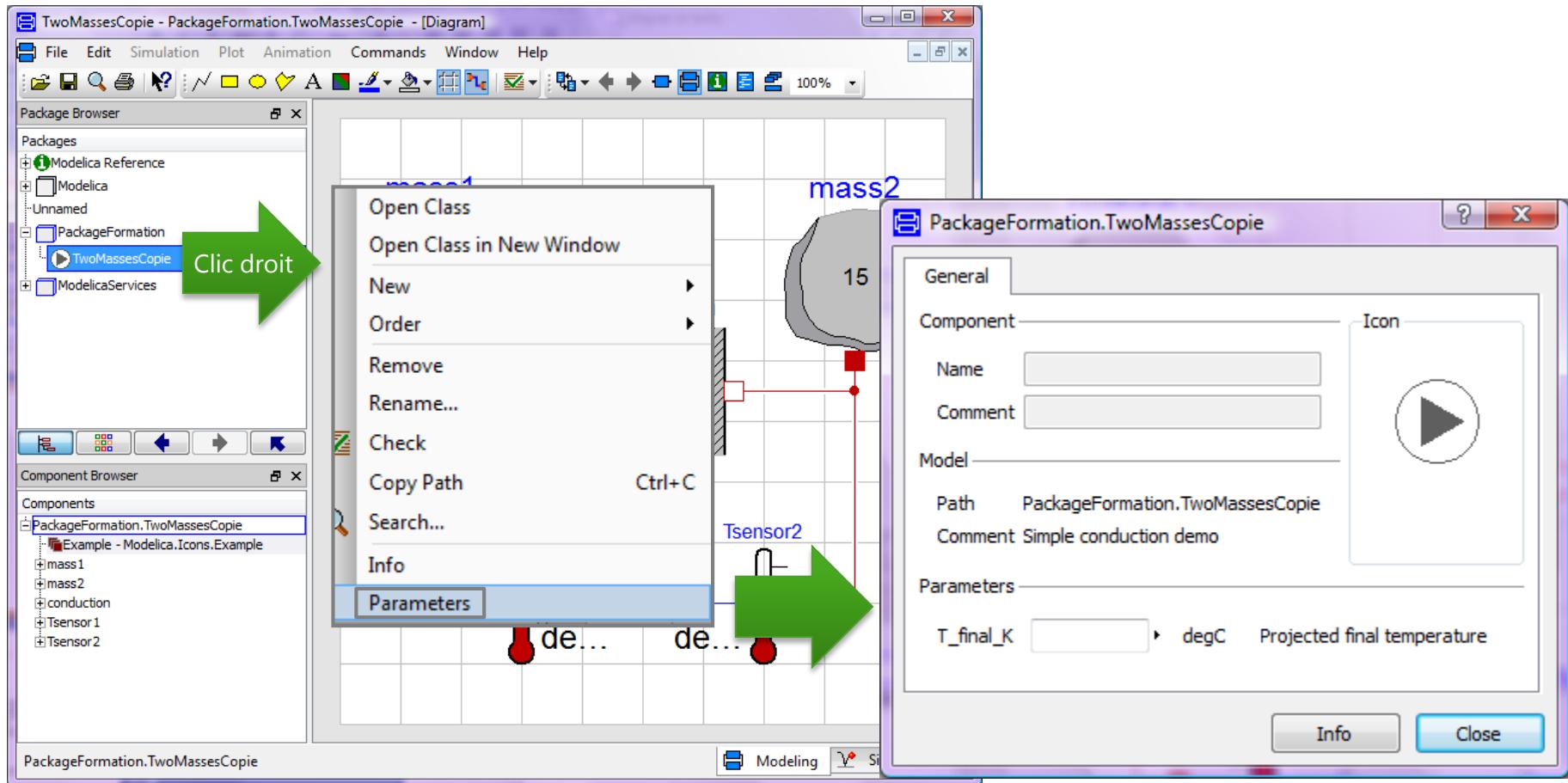
COPIE D'UN MODÈLE EXISTANT : 'DUPLICATE CLASS'

- Copie du modèle 'TwoMasses' du package 'Modelica'
 - Chemin du modèle : Modelica.Thermal.HeatTransfer.Examples.TwoMasses
 - Nouveau chemin souhaité : PackageFormation.TwoMassesCopie



GESTION DES MODÈLES

VISUALISATION DES PARAMÈTRES D'UN MODÈLE



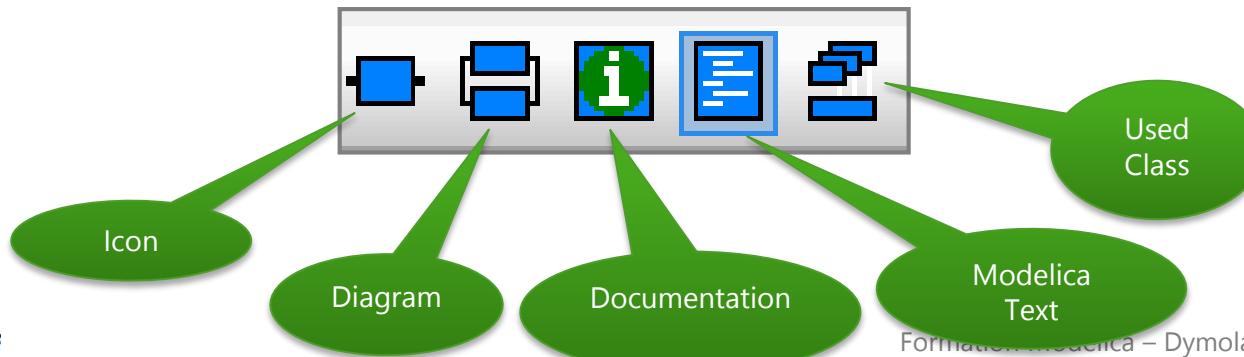
FENÊTRES D'ÉDITION DISPONIBLES

- 5 fenêtres d'édition disponibles

- **Icon** = représentation externe graphique du modèle
- **Diagram** = sa constitution s'il est issu de connexions graphiques entre plusieurs modèles
- **Documentation** = informations aidant à la compréhension du modèle
- **Modelica Text** = code du modèle en langage Modelica
- **Used Class** = code du modèle, utile lors d'une impression car le code des autres modèles utilisées y est indiqué explicitement

- Fenêtres

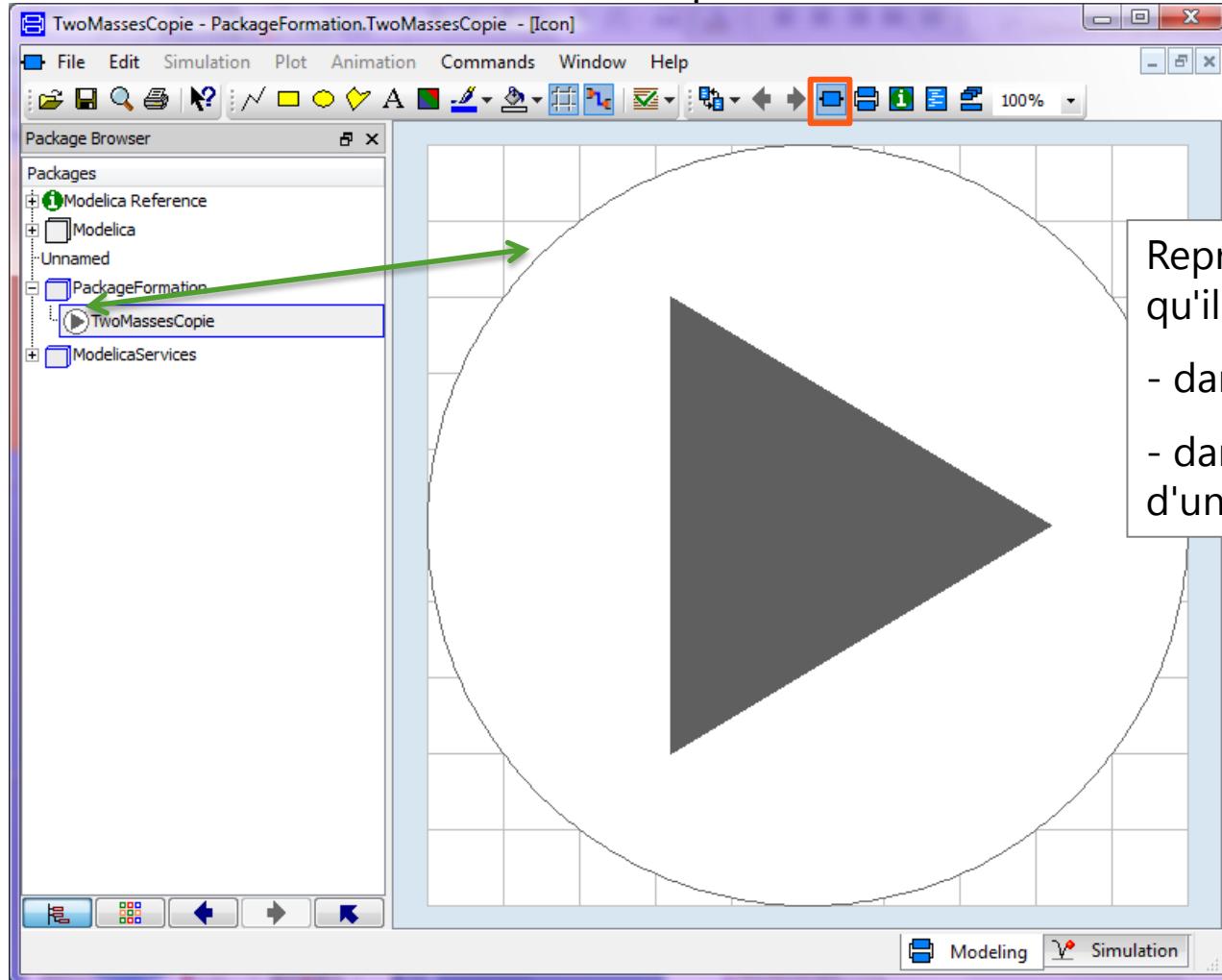
- Toute l'information est contenue dans le code Modelica et tout changement dans l'une des fenêtres se répercute dans les autres fenêtres



FENÊTRES D'ÉDITION DISPONIBLES

'ICON' : ICÔNE DU MODÈLE

Exemple du modèle TwoMasses



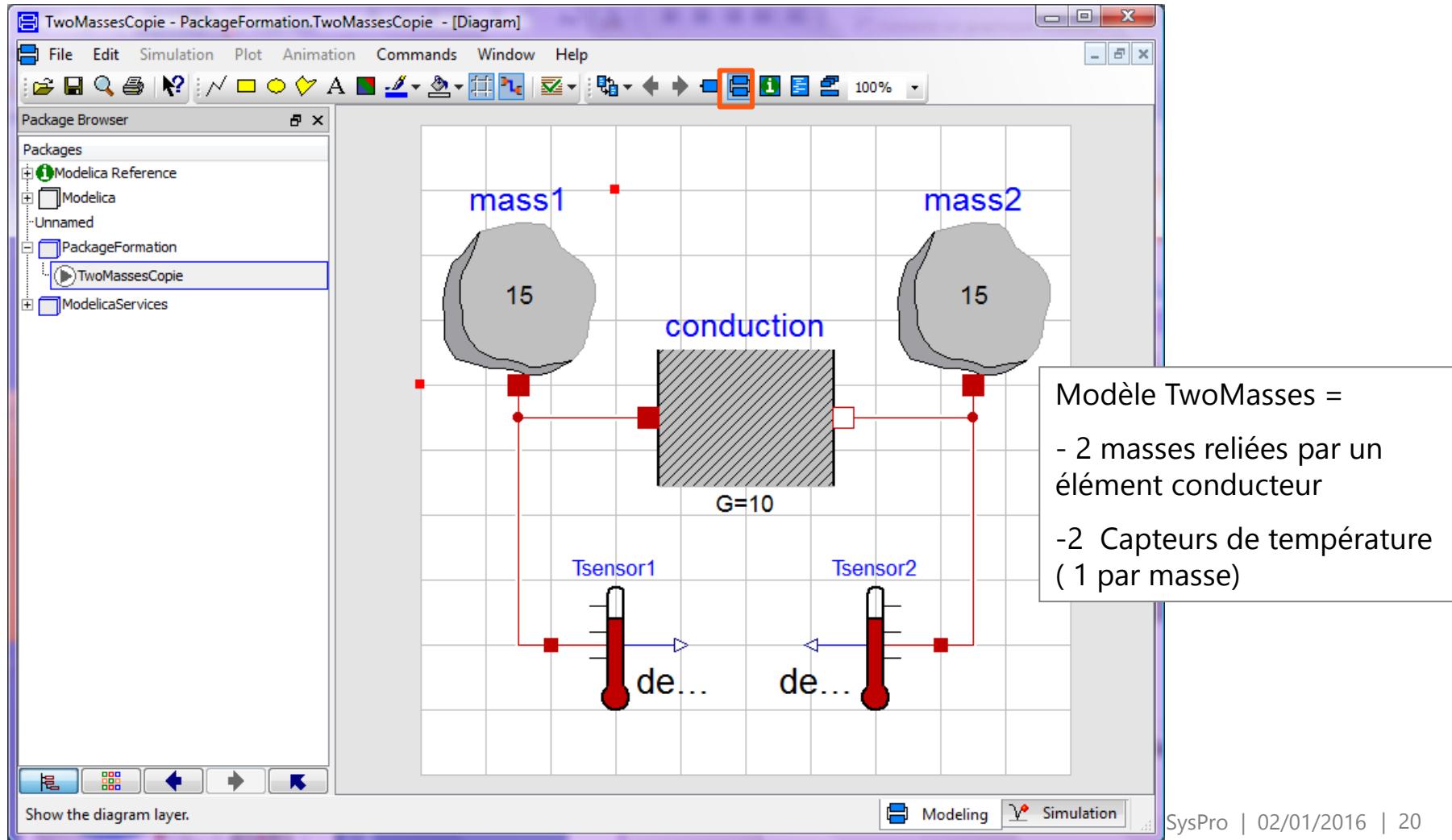
Représentation du modèle tel qu'il apparaît

- dans le *package browser*
- dans l'*assemblage (diagram)* d'un autre modèle

FENÊTRES D'ÉDITION DISPONIBLES

'DIAGRAM' : ASSEMBLAGE DU MODÈLE

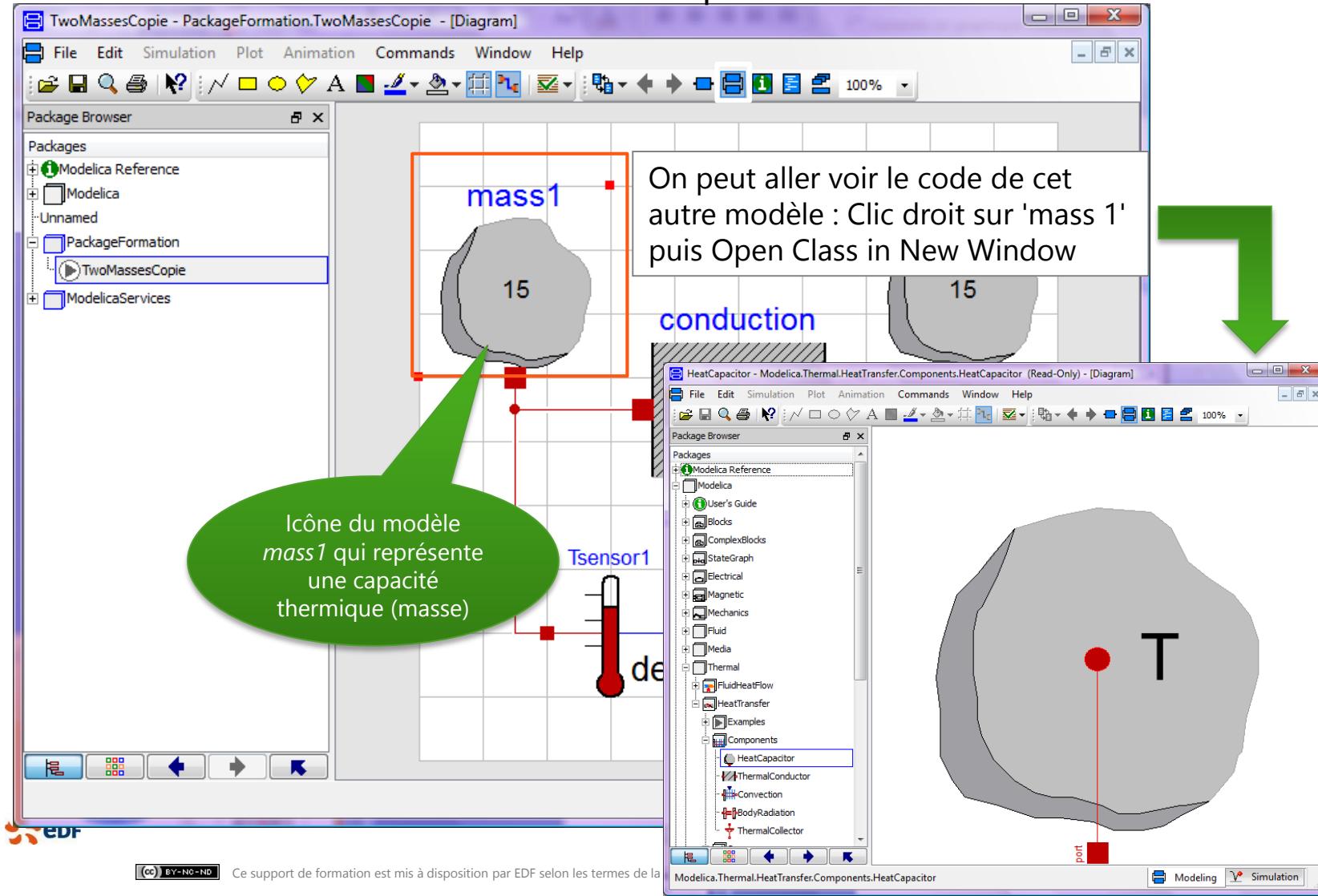
Exemple du modèle TwoMasses



FENÊTRES D'ÉDITION DISPONIBLES

'DIAGRAM' : ASSEMBLAGE DU MODÈLE

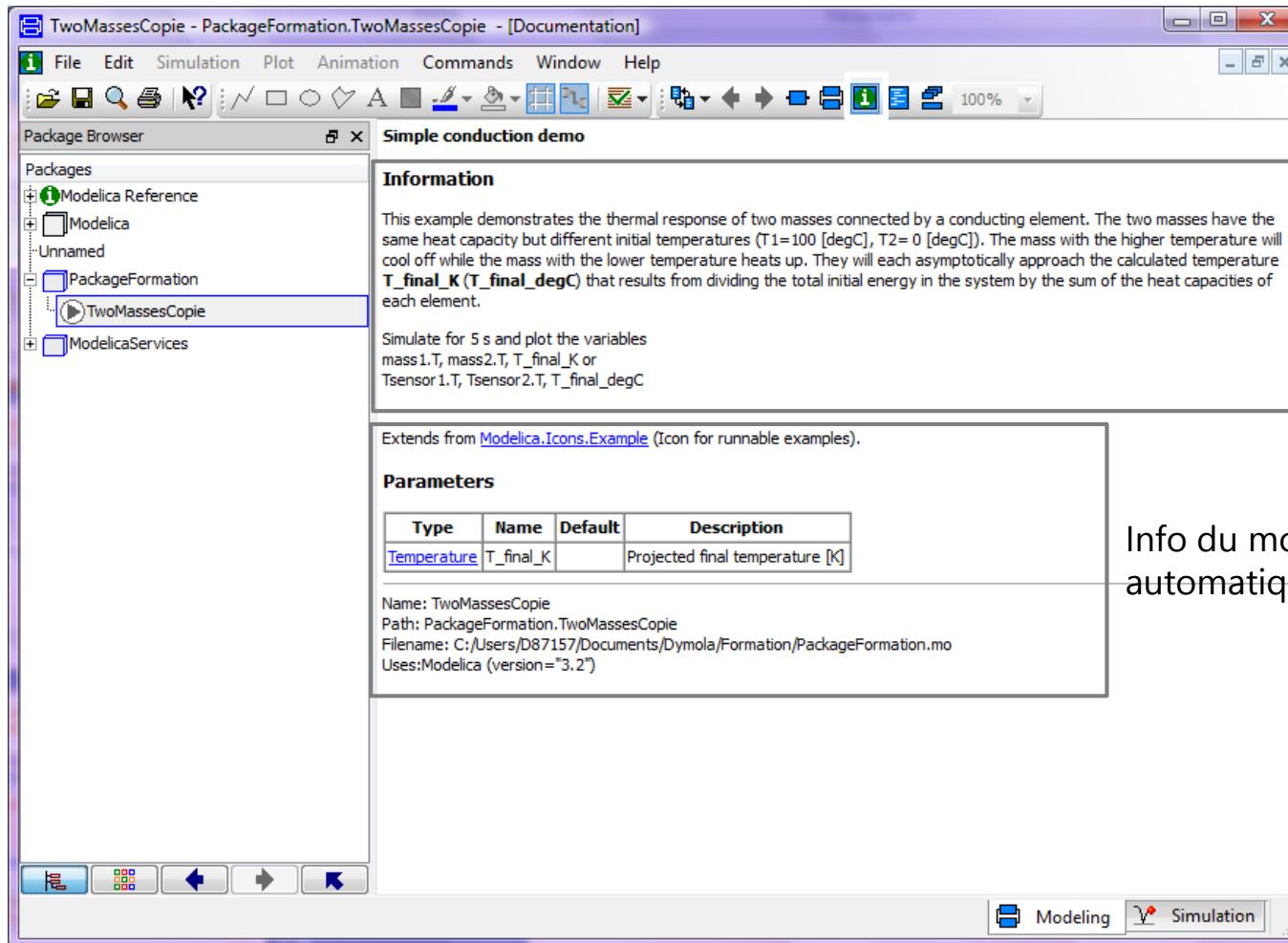
Exemple du modèle TwoMasses



FENÊTRES D'ÉDITION DISPONIBLES

'DOCUMENTATION': INFORMATIONS SUR LE MODÈLE

Exemple du modèle TwoMasses



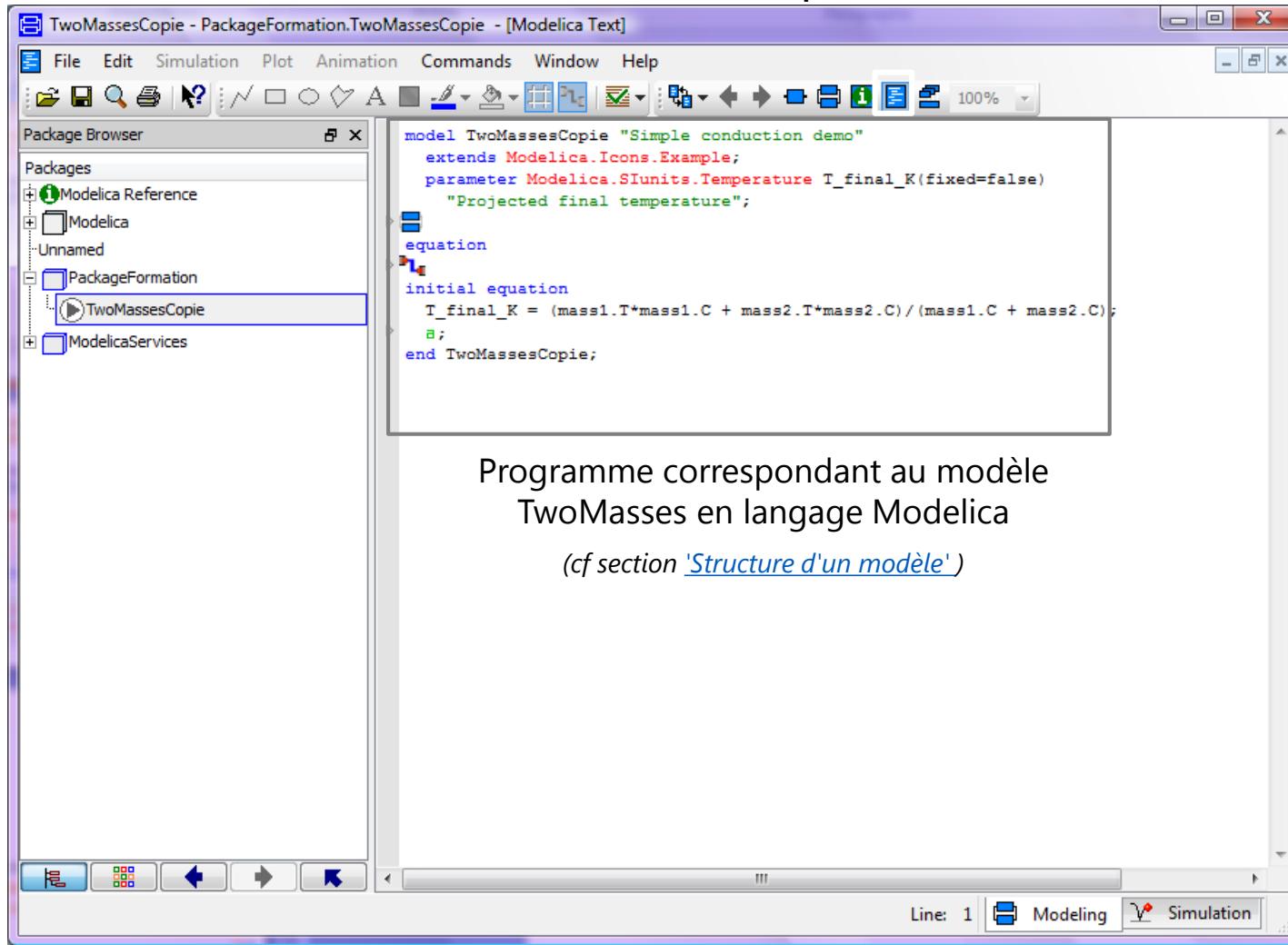
Informations données par le développeur pour aider à la compréhension du modèle

Info du modèle renseignées automatiquement par Dymola

FENÊTRES D'ÉDITION DISPONIBLES

'MODELICA TEXT': CODE EN MODELICA DU MODÈLE

Exemple du modèle TwoMasses



The screenshot shows the Modelica Text editor interface. The title bar reads "TwoMassesCopie - PackageFormation.TwoMassesCopie - [Modelica Text]". The menu bar includes File, Edit, Simulation, Plot, Animation, Commands, Window, and Help. The toolbar contains various icons for file operations and simulation. On the left, the Package Browser shows a tree structure with packages: Modelica Reference, Modelica, Unnamed, PackageFormation, TwoMassesCopie (selected), and ModelicaServices. The main workspace displays the following Modelica code:

```
model TwoMassesCopie "Simple conduction demo"
  extends Modelica.Icons.Example;
  parameter Modelica.SIunits.Temperature T_final_K(fixed=false)
    "Projected final temperature";
  equation
    initial equation
      T_final_K = (mass1.T*mass1.C + mass2.T*mass2.C) / (mass1.C + mass2.C);
    end;
  end TwoMassesCopie;
```

Below the code, a descriptive text reads:

Programme correspondant au modèle
TwoMasses en langage Modelica
(cf section '[Structure d'un modèle](#)')

FENÊTRES D'ÉDITION DISPONIBLES

'USED CLASS': CODE MODELICA RASSEMBLÉ (POUR IMPRESSION)

Exemple du modèle TwoMasses

```
TwoMassesCopie - PackageFormation.TwoMassesCopie - [Used Classes]
File Edit Simulation Plot Animation Commands Window Help
File Edit Simulation Plot Animation Commands Window Help
Used Classes
model TwoMassesCopie "Simple conduction demo"
  extends Modelica.Icons.Example;
  parameter Modelica.SIunits.Temperature T_final_K(fixed=false)
    "Projected final temperature";
  Modelica.Thermal.HeatTransfer.Components.HeatCapacitor mass1(C=15, T(start=
    373.15, fixed=true));
  Modelica.Thermal.HeatTransfer.Components.HeatCapacitor mass2(C=15, T(start=
    273.15, fixed=true));
  Modelica.Thermal.HeatTransfer.Components.ThermalConductor conduction(G=10);
  Modelica.Thermal.HeatTransfer.Celsius.TemperatureSensor Tsensor1;
  Modelica.Thermal.HeatTransfer.Celsius.TemperatureSensor Tsensor2;
equation
  connect(mass1.port, conduction.port_a);
  connect(conduction.port_b, mass2.port);
  connect(mass1.port, Tsensor1.port);
  connect(mass2.port, Tsensor2.port);
initial equation
  T_final_K = (mass1.T*mass1.C + mass2.T*mass2.C) / (mass1.C + mass2.C);
end TwoMassesCopie;

partial package Modelica.Icons.Package "Icon for standard packages"

end Package;

model Modelica.Thermal.HeatTransfer.Components.HeatCapacitor
  "Lumped thermal element storing heat"
  parameter Modelica.SIunits.HeatCapacity C "Heat capacity of element (= cp*m)";
  Modelica.SIunits.Temperature T(start=293.15, displayUnit="degC")
    "Temperature of element";
  Modelica.SIunits.TemperatureSlope der_T(start=0)
    "Time derivative of temperature (= der(T))";
  Interfaces.HeatPort_a port;
equation
  T = port.T;
```

Show the base classes and used classes.

Modeling Simulation

Modèle
TwoMassesCopie

Modèles
utilisés dans
TwoMasses

FENÊTRES D'ÉDITION DISPONIBLES 'USED CLASS' + FLAT MODELICA -> CODE TRÈS RASSEMBLÉ

Exemple du modèle TwoMasses

The screenshot shows the Modelica Studio interface with the 'TwoMasses' model open. The left side features a Package Browser with a tree view of Modelica packages, including 'Modelica Reference', 'Modelica', 'User's Guide', 'Blocks', 'ComplexBlocks', 'StateGraph', 'Electrical', 'Magnetic', 'Mechanics', 'Fluid', 'Media', 'Thermal', 'FluidHeatFlow', and 'HeatTransfer'. Under 'HeatTransfer/Examples', the 'TwoMasses' model is selected and highlighted with a blue border. The main workspace on the right displays the code for the 'TwoMasses' model. The code is color-coded to highlight different components and types. A context menu is visible at the top of the code area, with the option 'Flat Modelica' highlighted.

```
model TwoMasses
  parameter Modelica.SIunits.Temperature T_final_K(fixed = false) = 288.15
    "Projected final temperature";
  parameter Modelica.SIunits.HeatCapacity mass1.C = 15 "Heat capacity of el
  parameter Modelica.SIunits.HeatCapacity mass2.C = 15 "Heat capacity of el
  parameter Modelica.SIunits.ThermalConductance conduction.G = 10
    "Constant thermal conductance of material";

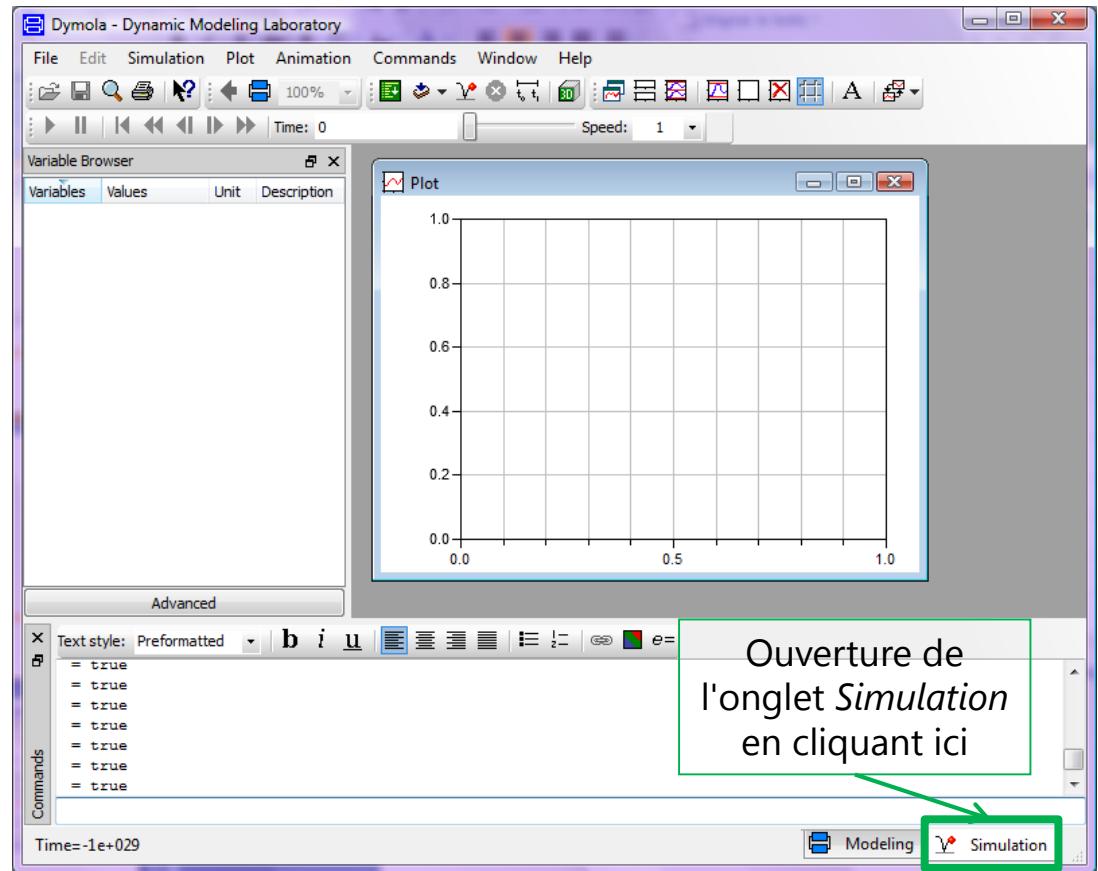
  Modelica.SIunits.Temperature mass1.T(start = 373.15, fixed = true)
    "Temperature of element";
  Modelica.SIunits.TemperatureSlope mass1.der_T(start = 0) "Time derivative
  Modelica.SIunits.Temperature mass1.port.T "Port temperature";
  Modelica.SIunits.HeatFlowRate mass1.port.Q_flow "Heat flow rate (positive
  Modelica.SIunits.Temperature mass2.T(start = 273.15, fixed = true)
    "Temperature of element";
  Modelica.SIunits.TemperatureSlope mass2.der_T(start = 0) "Time derivative
  Modelica.SIunits.Temperature mass2.port.T "Port temperature";
  Modelica.SIunits.HeatFlowRate mass2.port.Q_flow "Heat flow rate (positive
  Modelica.SIunits.HeatFlowRate conduction.Q_flow "Heat flow rate from port
  Modelica.SIunits.TemperatureDifference conduction.dT "port_a.T - port_b.T
  Modelica.SIunits.Temperature conduction.port_a.T "Port temperature";
  Modelica.SIunits.HeatFlowRate conduction.port_a.Q_flow "Heat flow rate (g
  Modelica.SIunits.Temperature conduction.port_b.T "Port temperature";
  Modelica.SIunits.HeatFlowRate conduction.port_b.Q_flow "Heat flow rate (g
  Modelica.Blocks.Interfaces.RealOutput Tsensor1.T;
  Modelica.SIunits.Temperature Tsensor1.port.T "Port temperature";
  Modelica.SIunits.HeatFlowRate Tsensor1.port.Q_flow "Heat flow rate (posit
  Modelica.Blocks.Interfaces.RealOutput Tsensor2.T;
  Modelica.SIunits.Temperature Tsensor2.port.T "Port temperature";
  Modelica.SIunits.HeatFlowRate Tsensor2.port.Q_flow "Heat flow rate (posit

  // Equations and algorithms

  // Component mass1
  // class Modelica.Thermal.HeatTransfer.Components.HeatCapacitor
  equation
```

Dans la vue « Used Class », faire un clic-droit dans le code et sélectionner « Flat Modelica »

Tout le code est rassemblé en un seul modèle équivalent



Onglet 'Simulation' ENVIRONNEMENT DYMOLA / LANGUAGE MODELICA

CONTENU DE L'ONGLET 'SIMULATION'

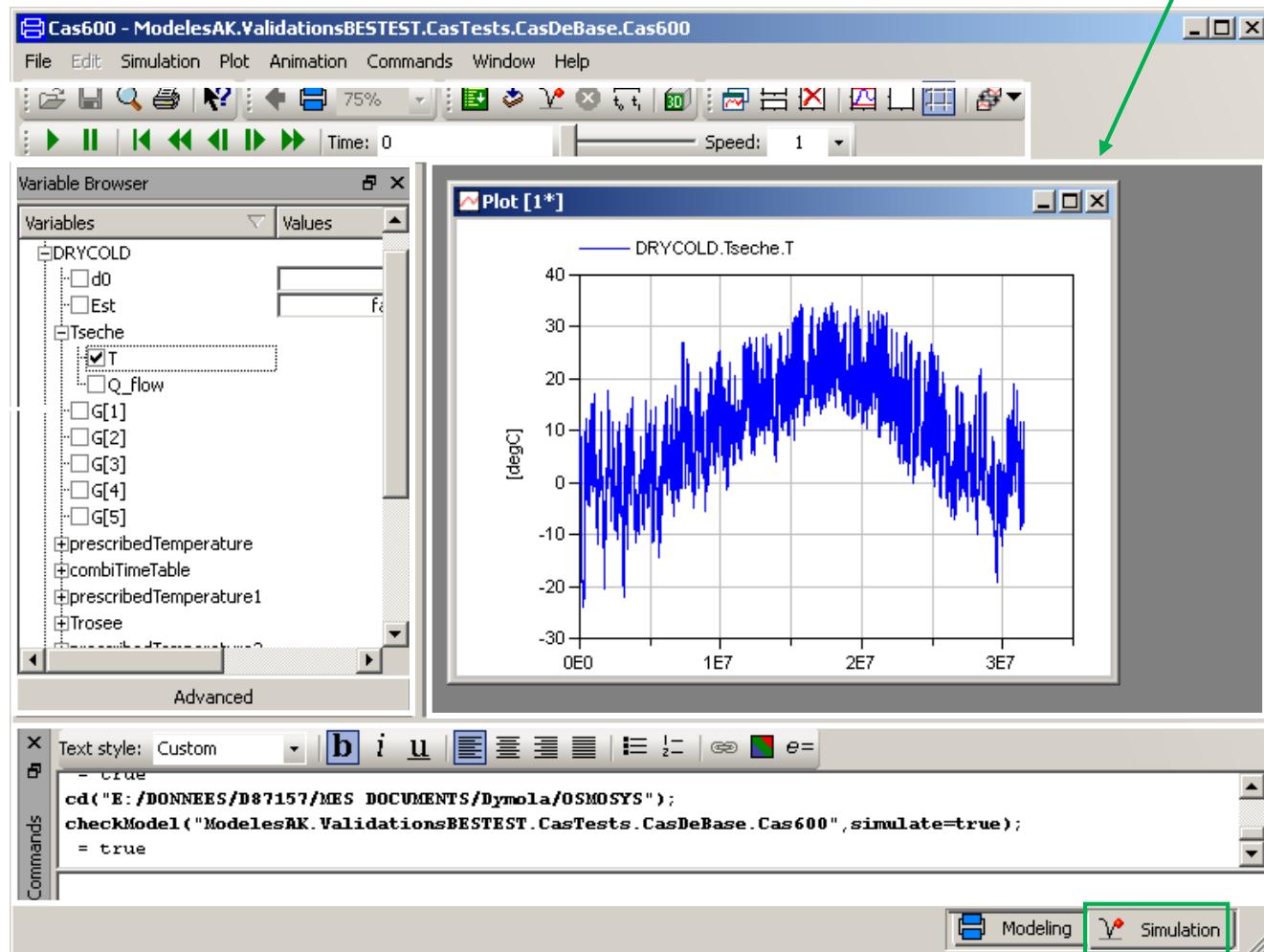
ASPECT GÉNÉRAL

Barre d'outil

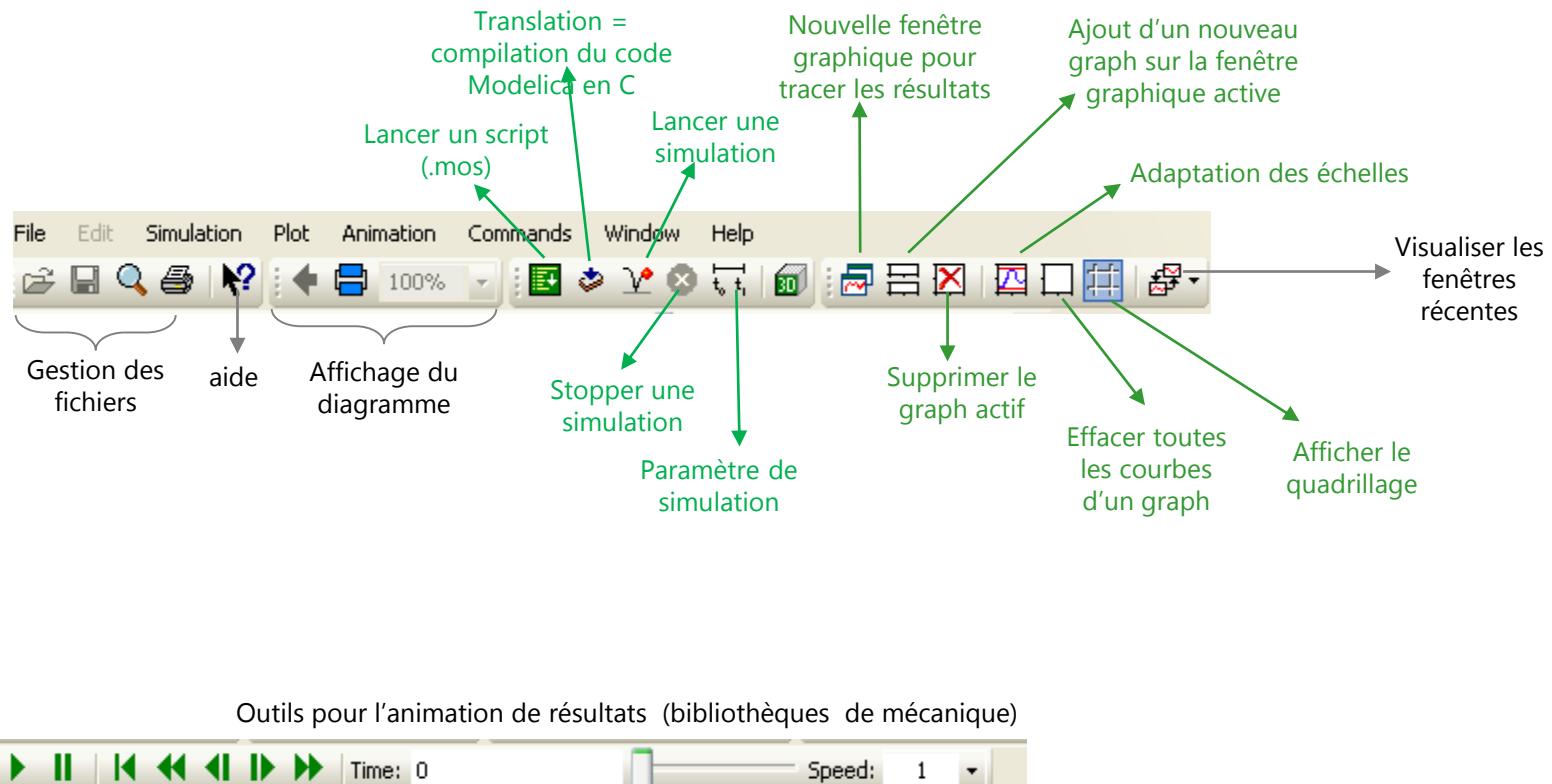
Variables du modèles

Fenêtre de commande

Visualisation des résultats



CONTENU DE L'ONGLET 'SIMULATION' BARRE D'OUTILS

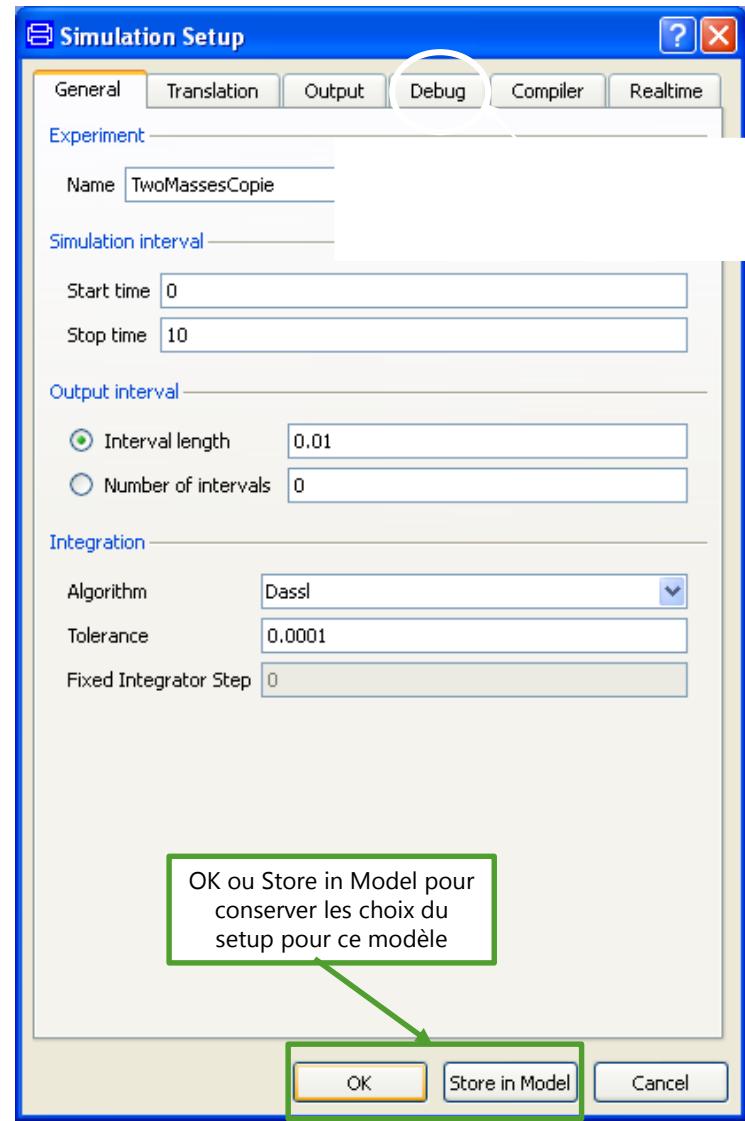
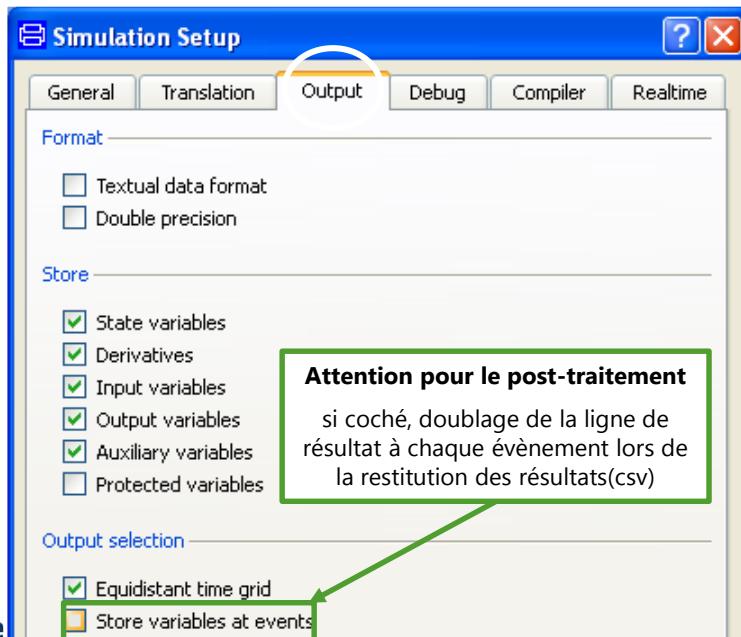


LANCLEMENT D'UNE SIMULATION

MODIFICATION DES PARAMÈTRES DE LA SIMULATION

■ Exemple du modèle TwoMasses

- Ouvrir Setup 
- Changer des paramètres
 - Durée de la simulation = 10s
 - Visualisation toutes les 0,01s
 - Choix du solveur : Dassl



LANCLEMENT D'UNE SIMULATION

LANCER UNE SIMULATION ET VISUALISER LES RÉSULTATS

- Exemple du modèle

TwoMasses

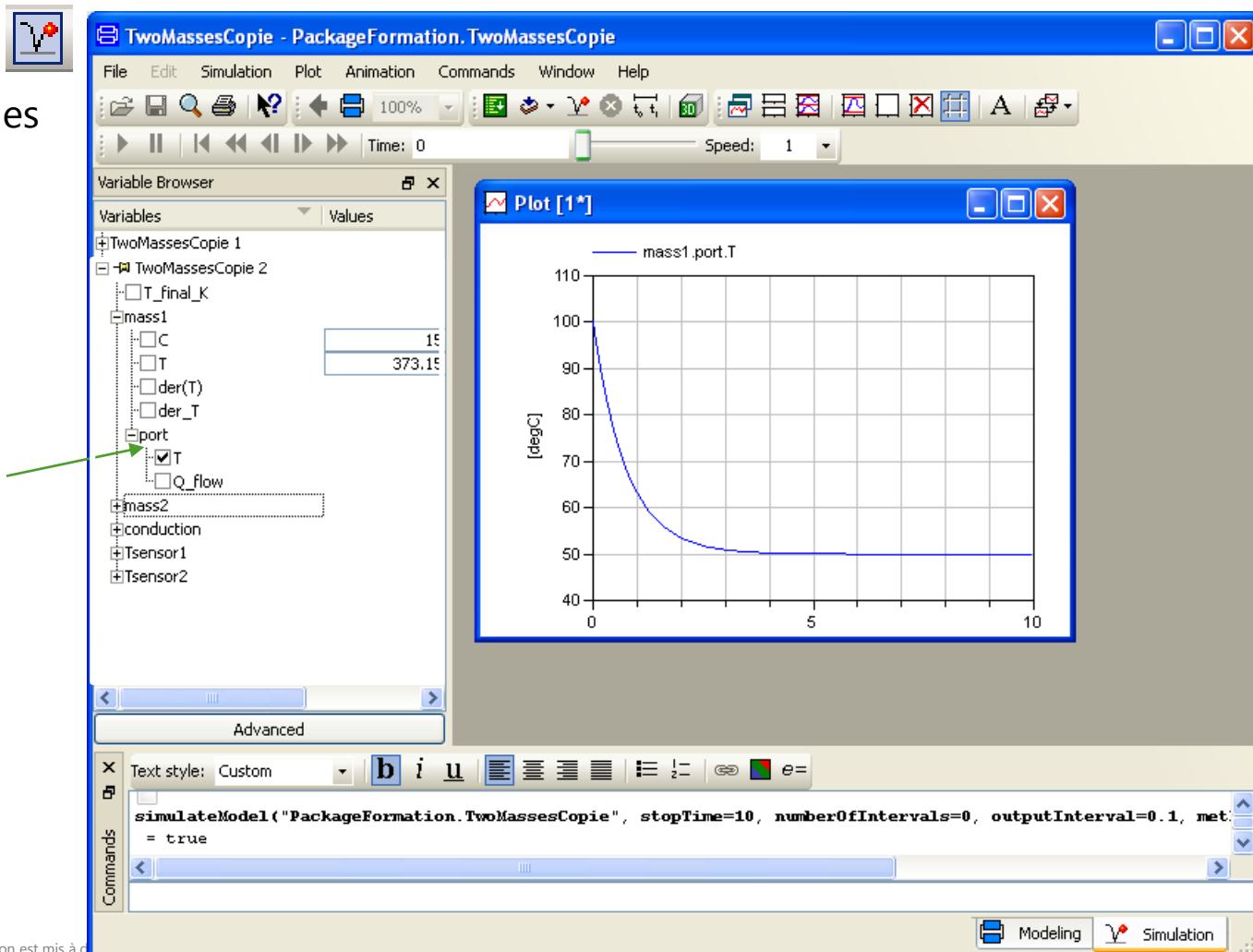
- Lancer la simulation



- Visualiser des variables

- Cocher dans l'arborescence

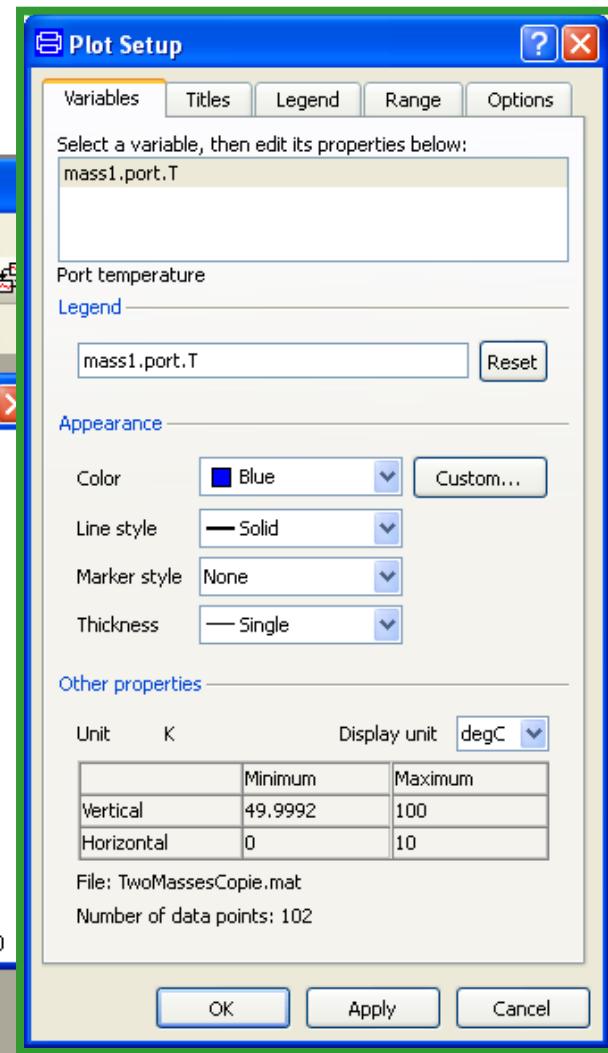
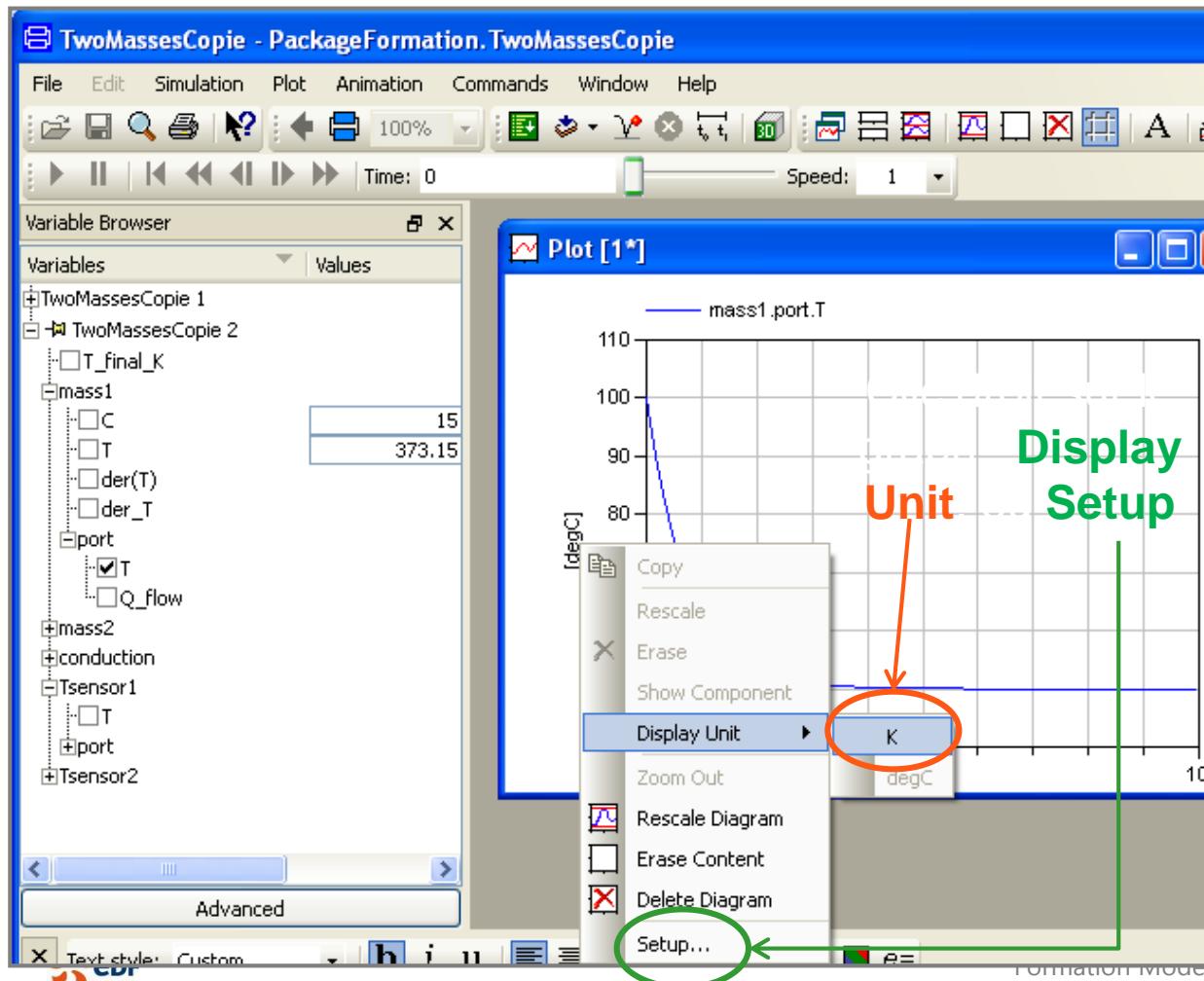
Visualisation de la température T du connecteur *port* de la masse *mass1*



LANCLEMENT D'UNE SIMULATION

VISUALISATION LES RÉSULTATS

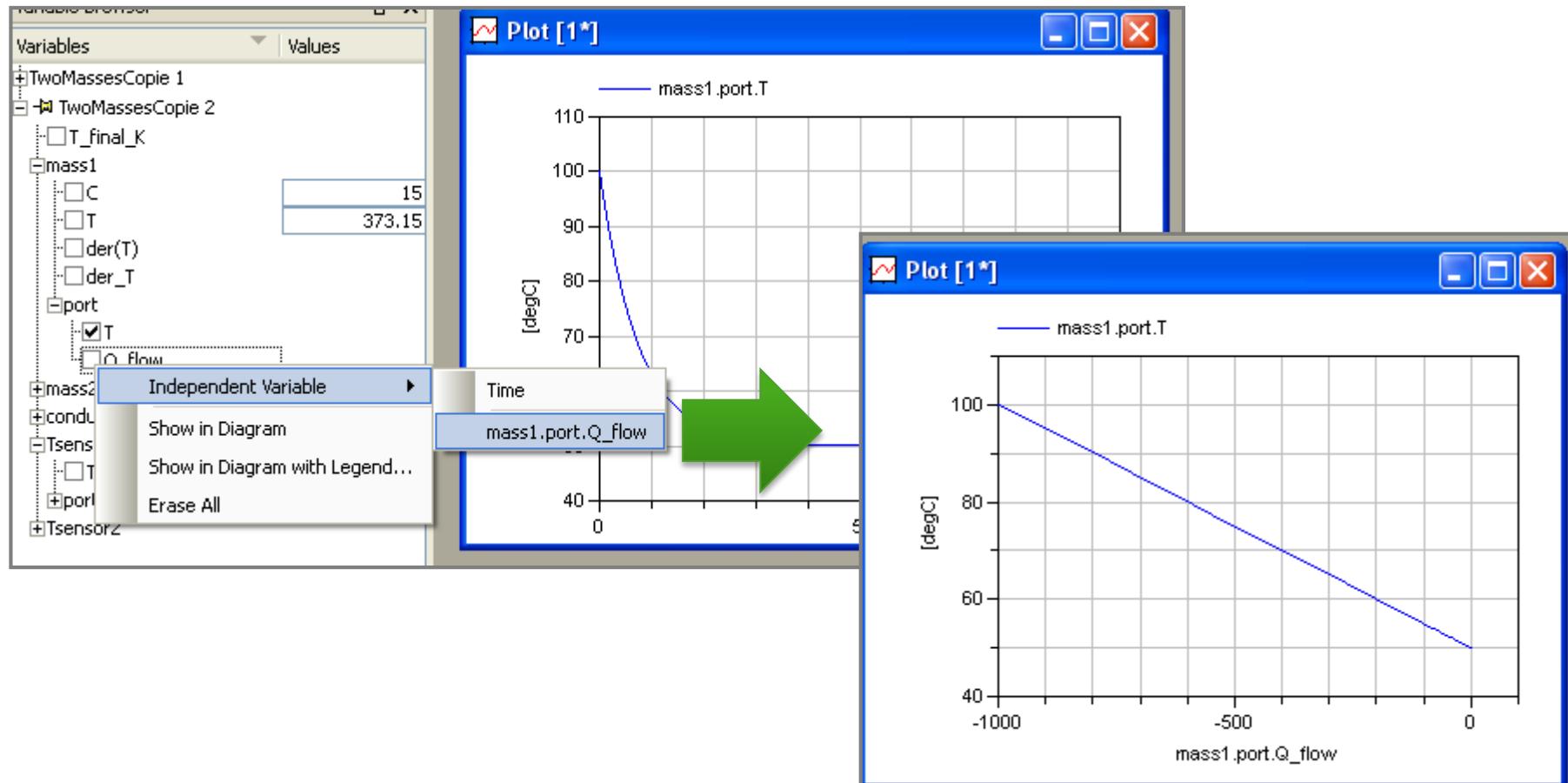
- Changer les unités dans le graph



LANCLEMENT D'UNE SIMULATION

VISUALISATION LES RÉSULTATS

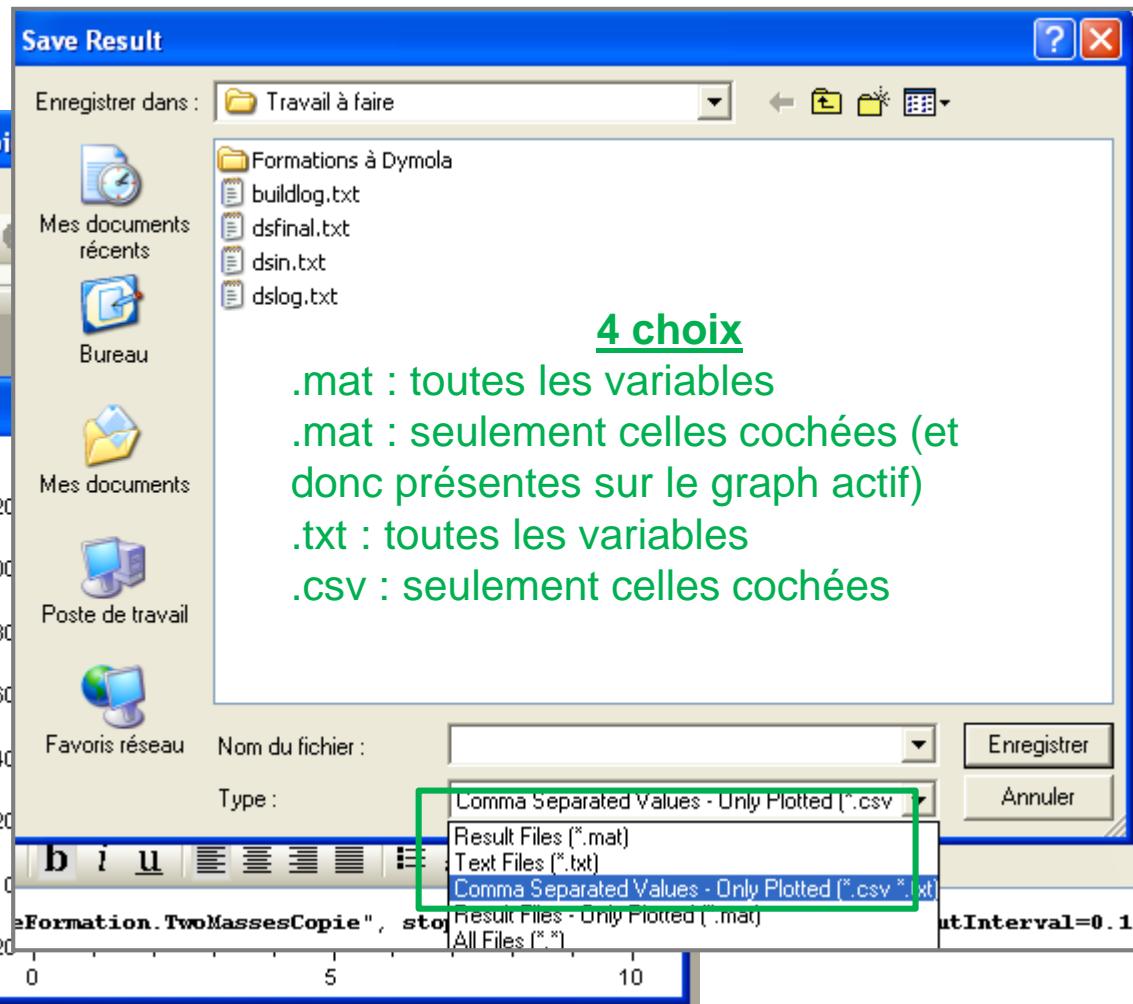
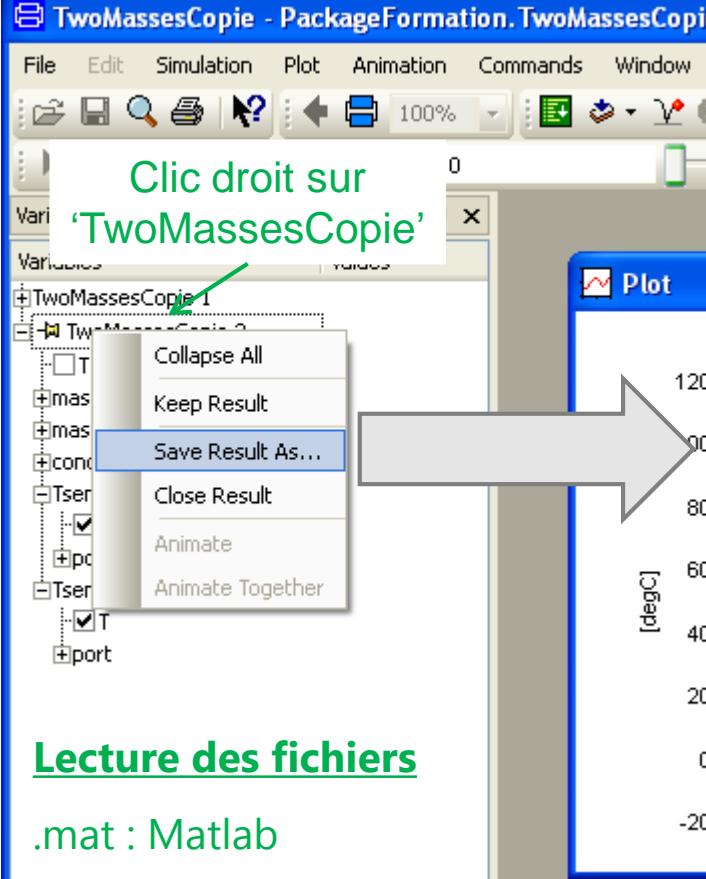
- Tracer 1 variable par rapport à une autre qui n'est pas le temps

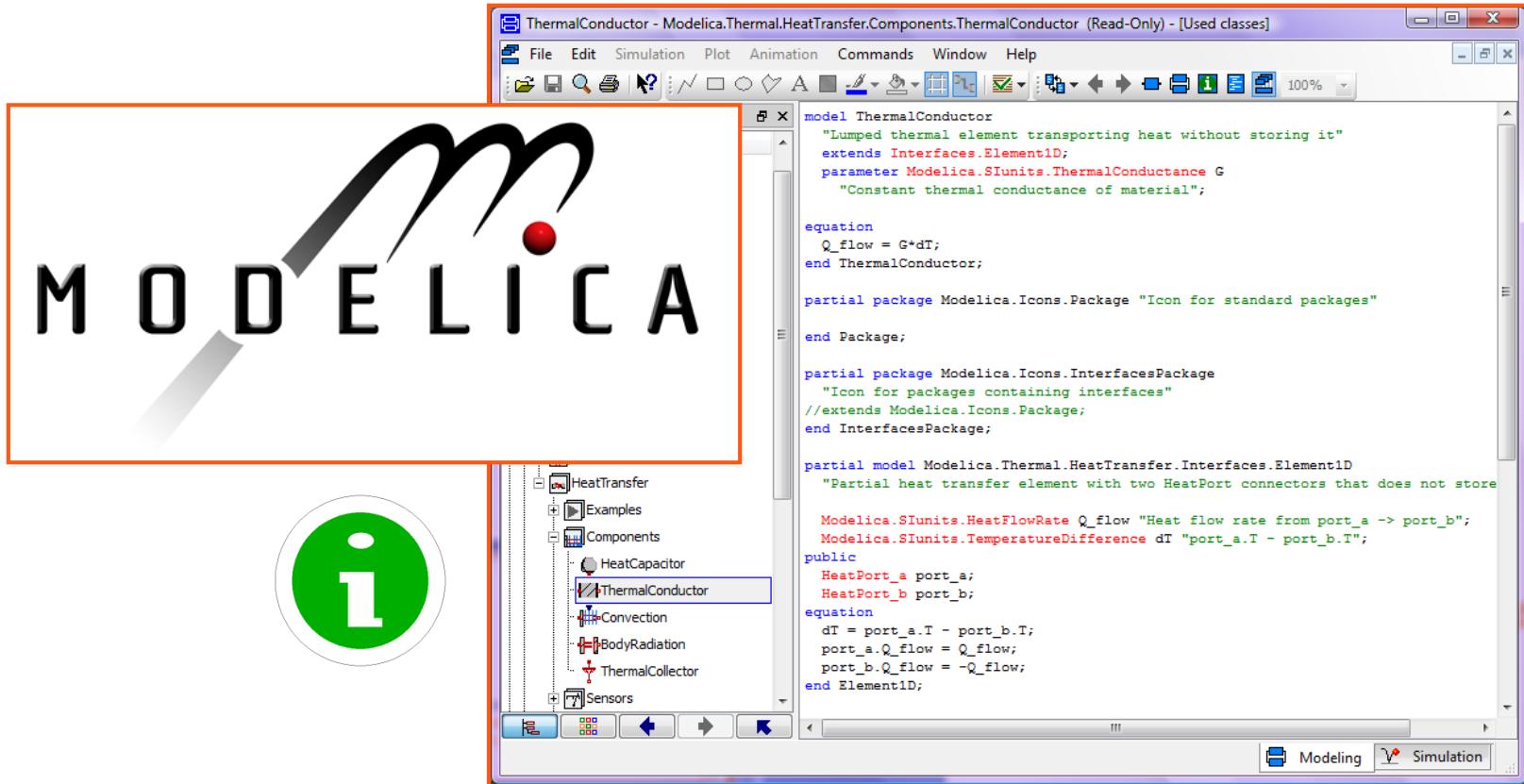


LANCLEMENT D'UNE SIMULATION

VISUALISATION LES RÉSULTATS

□ Export des résultats

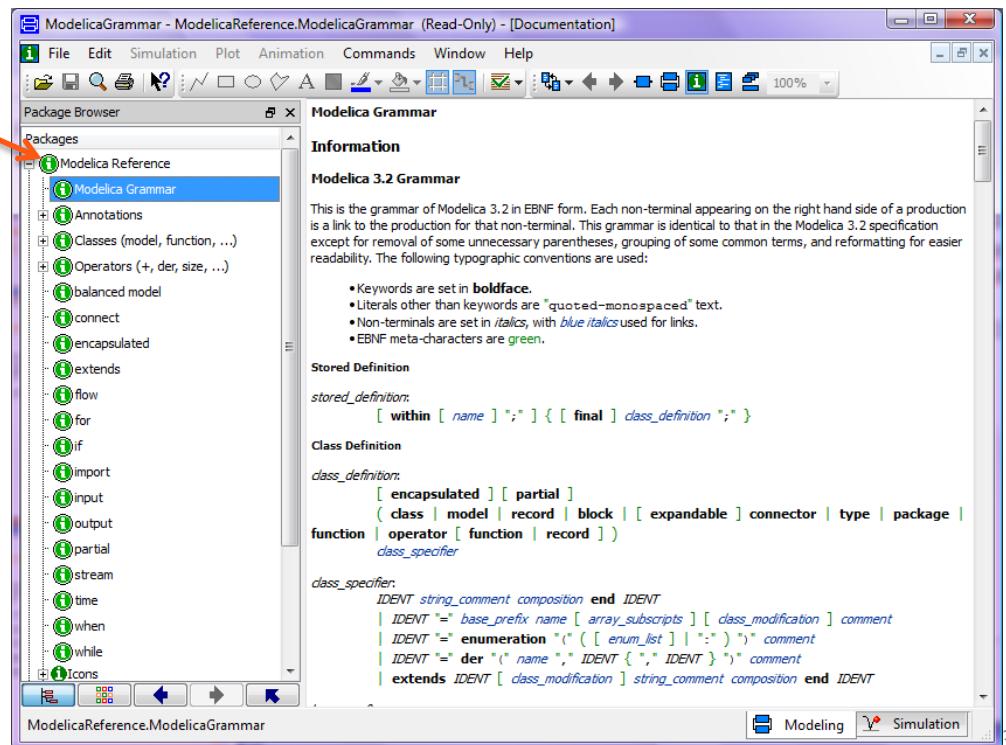




Éléments du langage ENVIRONNEMENT DYMOLA / LANGAGE MODELICA

PRÉAMBULE

- Les points spécifiques du langage Modelica sont disponibles
 - De manière détaillée dans le document pdf "ModelicaSpec" accessible dans Dymola au niveau du menu Help/Documentation/Modelica Language Specification
 - De façon simplifiée dans le package browser de Dymola dans le package ModelicaReference



LES DIFFÉRENTES CLASSES DANS MODELICA

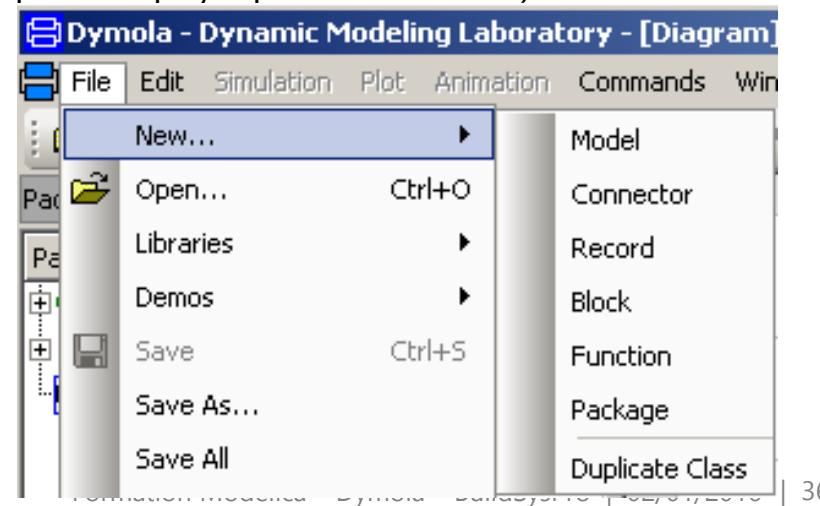
■ 6 types de classes (mot clé 'class' générique)

- **Package** = conteneur de modèles (de n'importe quelle classe) permettant d'organiser en dossiers la bibliothèque
- **Connector** = interface de connexion des modèles physiques pouvant être causal ou acausal
- **Record** = structure de stockage de données uniquement (pas d'équations)
- **Function** = fonction qui effectue un calcul purement mathématique (algorithme séquentiel)
- **Model** = modèle acausal (pour les modèles de composants physiques notamment)
- **Block** = modèle causal (entrées et sorties fixées)



Retour au tutoriel
Création d'un package

*Choix d'une classe
dans Dymola*



COMPOSITION D'UN MODÈLE

« MODEL TEMPLATE » DE BUILDSYSPRO

The screenshot shows the 'ModelTemplate - BuildSysPro.UsersGuide.ModelTemplate - [Modelica Text]' window. The left sidebar is the 'Package Browser' showing the 'BuildSysPro' package structure, including 'User's Guide' and 'Model template with recommendations'. The main area contains the following Modelica code:

```
partial model ModelTemplate
    "Model template with recommendations in terms of documentation and code-writing"
    //---- Declarative section - please comment each variable and parameter ----//
    // Do not write any equation in this section for variables. Otherwise, the variable will be considered as a parameter (constant).

    // Declaration of model parameters using associated units and range of variation if necessary
    parameter Real alpha(min=0, max=1)
        "Alpha coefficient - range of variation between 0 and 1";
    parameter Modelica.SIunits.Length Ls2 "Length";
    parameter BuildSysPro.Utilities.Types.FileNameIn Data=
        Modelica.Utilities.Files.loadResource("modelica://BuildSysPro/Resources/Donnees/Meteos/RT2012/H1a.txt")
        "Path of weather data file";
    parameter Boolean Bool "Parameter showing radio buttons" a;
    parameter Integer Choice "Parameter showing a drop-down menu" a;
    parameter BuildSysPro.Utilities.Records.GenericWall Wall
        "Choice between wall data records" a;

    // Declaration of variables
    Modelica.SIunits.Length D "Distance";

    // Used models

    // Variables or parameters visible only in the model
    protected
        parameter Real beta=alpha*100 "Alpha in the form of a percentage...";
        Real gamma "Intermediate variable not visible by default in the results";

    //---- Section of algorithms & equations ----//
    //--- Algorithms - for classic or sequential calculations, not taking part of the equational
    // Be careful, an algorithm should not be considered as an equation, contrary to equations t
    algorithm
        D:=beta*L; // For instance...

    //--- Equations and connections between models ---
    // The equations form the matrix system solved by the solver - Number of equation = Number of
    equation
        gamma = if Bool then D else 0;

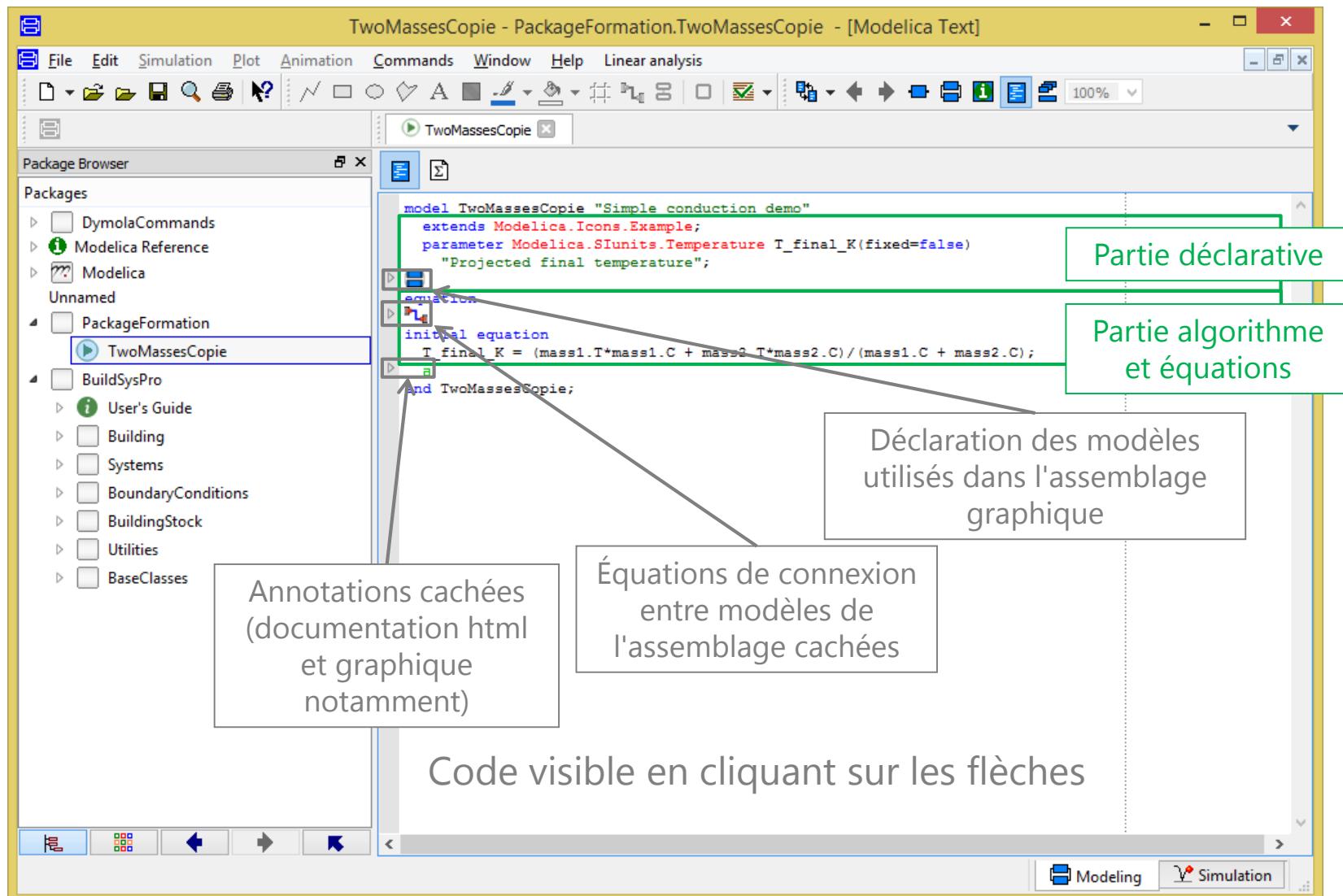
end ModelTemplate;
```

Annotations are present in the code:

- Annotations for parameters: 'a' is highlighted in red boxes at three locations.
- An annotation for the variable 'Wall' is also highlighted in red.
- A large green box encloses the entire code area.
- Callout boxes explain specific parts:
 - 'Déclaration des paramètres et variables du modèle y compris modèles utilisés dans l'assemblage'
 - 'Équations et algorithmes y compris liaisons entre modèles'
 - 'Annotations cachées par défaut (html, aspects graphiques, IHM, ...)' pointing to the 'protected' section.

COMPOSITION D'UN MODÈLE

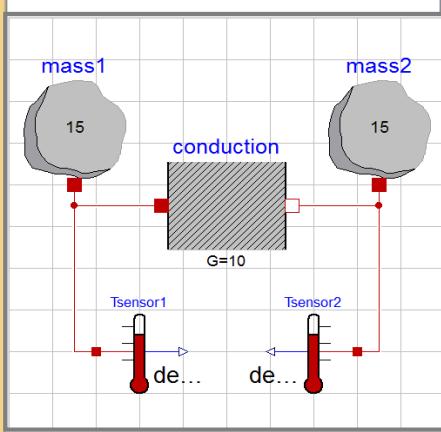
EXEMPLE DU MODÈLE TWOMASSES



COMPOSITION D'UN MODÈLE

VISUALISATION DES MODÈLES UTILISÉS DANS TWOMASSES

Modèles utilisés



Partie déclarative

```
model TwoMassesCopie "Simple conduction demo"
  extends Modelica.Icons.Example;
  parameter Modelica.SIunits.Temperature T_final_K(fixed=false)
    "Projected final temperature";
  Modelica.Thermal.HeatTransfer.Components.HeatCapacitor mass1(C=15, T(start=373.15,
    fixed=true)) a;
  Modelica.Thermal.HeatTransfer.Components.HeatCapacitor mass2(C=15, T(start=273.15,
    fixed=true)) b;
  Modelica.Thermal.HeatTransfer.Components.ThermalConductor conduction(G=10) a;
  Modelica.Thermal.HeatTransfer.Celsius.TemperatureSensor Tsensor1 a;
  Modelica.Thermal.HeatTransfer.Celsius.TemperatureSensor Tsensor2 b;
equation
  initial equation
    T_final_K = (mass1.T+mass1.C + mass2.T+mass2.C) / (mass1.C + mass2.C);
  end TwoMassesCopie;
```

Partie algorithme et équations

- 2 HeatCapacitor (masses)
- 2 TemperatureSensor (capteurs de températures)
- 1 ThermalConductor (élément de conduction)

COMPOSITION D'UN MODÈLE

VISUALISATION DES LIAISONS DE TWOMASSES

TwoMassesCopie - PackageFormation.TwoMassesCopie - [Modelica Text]

File Edit Simulation Plot Animation Commands Window Help Linear analysis

Package Browser

Packages

- DymolaCommands
- Modelica Reference
- Modelica
- Unnamed
- PackageFormation
- TwoMassesCopie**
- BuildSysPro

Équations de liaison

model TwoMassesCopie "Simple conduction demo"
extends Modelica.Icons.Example;
parameter Modelica.SIunits.Temperature T_final_K(fixed=false)
"Projected final temperature";

equation
connect (mass1.port, conduction.port_a) a;
connect (conduction.port_b, mass2.port) a;
connect (mass1.port, Tsensor1.port) a;
connect (mass2.port, Tsensor2.port) a;
initial equation
T_final_K = (mass1.T*mass1.C + mass2.T*mass2.C) / (mass1.C + mass2.C);
a
end TwoMassesCopie;

Partie déclarative

Partie algorithme et équations

1 liaison graphique entre 2 modèles = équation 'connect' entre deux éléments générée automatiquement par Dymola dans le code

COMPOSITION D'UN MODÈLE

VISUALISATION DES ANNOTATIONS DE TWOMASSES

The screenshot shows the Dymola software interface with the following annotations:

- Partie déclarative**: Points to the declaration section of the code, which includes the model definition and parameter declarations.
- Partie algorithme et équations**: Points to the algorithmic part, specifically the equation section where the final temperature is calculated.
- Documentation html**: Points to the documentation block within the code, which contains HTML comments describing the thermal response of two masses.
- Info sur les paramètres de simulation**: Points to the simulation parameters defined in the code, such as StopTime and Interval.
- Annotations**: A large callout pointing to the annotations section at the bottom left, which lists information about what can be found in the annotations.
- On peut y trouver des informations**: A callout pointing to the annotations section, listing three types of information:
 - sur l'aspect graphique (icône et lorsque le modèle est un assemblage, la position des modèles)
 - sur les options de chiffrement
 - sur les bibliothèques externes utilisées

```
model TwoMassesCopie "Simple conduction demo"
  extends Modelica.Icons.Example;
  parameter Modelica.SIunits.Temperature T_final_K(fixed=false)
    "Projected final temperature";
  equation
    initial equation
      T_final_K = (mass1.T*mass1.C + mass2.T*mass2.C)/(mass1.C + mass2.C);
    annotation (Documentation(info="
<HTML>
<p>
This example demonstrates the thermal response of two masses connected by a conducting element. The two masses have the same heat capacity but different initial temperatures (T1=100 [degC], T2= 0 [degC]). The mass with the higher temperature will cool off while the mass with the lower temperature heats up. They will each asymptotically approach the calculated temperature <b>T_final_K</b> (<b>T_final_degC</b>) that results from dividing the total initial energy in the system by the sum of the heat capacities of each element.
</p>
<p>
Simulate for 5 s and plot the variables<br>
mass1.T, mass2.T, T_final_K or <br>
Tsensor1.T, Tsensor2.T, T_final_degC
</p>
</HTML>"),
      experiment(StopTime=1.0, Interval=0.001));
end TwoMassesCopie;
```

COMPOSITION D'UN MODÈLE

COLORATION SYNTAXIQUE SOUS DYMOLA

- Pour faire apparaître la coloration syntaxique du code : Ctrl + L après avoir sélectionné le code (Ctrl + A)
- **Rouge** : les types et fonctions prédéfinies
- **Vert** : les annotations et commentaires
- **Bleu** : les mots clés réservés en langage Modelica
- **Noir** : tout le reste...

COMPOSITION D'UN MODÈLE

SECTION 'EQUATION' VS 'ALGORITHM'

Equation

- Connexions entre les modèles (equations 'connect')
- Équations acausales : l'ordre d'écriture importe peu
 - ⇒ L'équation $U=R*i$ peut se réécrire selon les variables connues
 - $U=R*i$
 - $R=U/i$
 - $i=U/R$
- Équations agrégées ⇒ Système matriciel d'équations résolu par le solveur

Algorithm

- Code séquentiel : Équations résolues dans l'ordre de lecture du code
- Équations n'augmentant pas la taille du système matriciel d'équations à résoudre

TESTS ET BOUCLES

■ If – déclarations conditionnelles



□ Expression classique

- if <condition1> then
 <déclaration1>
- elseif <condition2> then
 <déclaration2>
- else
 <déclaration3>
- end if

□ Expression compacte sur 1 ligne

```
if <condition1> then <déclaration1> else if <condition2> then  
    <déclaration2> else <déclaration3>;
```

Remarque : boucle **if** plus générique que la boucle **when** –
mais attention à la gestion des évènements différente

■ for - boucle de taille donnée



- for <variable d'itération> in <domaine d'itération> loop
 <déclaration1>
 <déclaration2>
 ...
end for

■ while - boucle tant que la condition est remplie



- while <condition> loop
 <déclaration>
end while;

■ when - active des équations quand la condition est remplie

- When <condition1> then
 <déclaration1>
- elsewhen <condition2> then
 <déclaration2>
- end when

Sections possibles

Algorithm



Equation



DÉFINITION DES TYPES

■ Variables

- **Real** = nombre réel (double précision)
- **Integer** = nombre entier
- **String** = chaîne de caractère
- **Boolean** = booléen (vrai ou faux)
- **Enumeration** = liste proposant plusieurs choix

■ Autres types : variables avec unités SI (bibliothèque Modelica.SIUnits)

- Pour définir des grandeurs physiques possédant une unité de mesure spécifique
- 450 grandeurs prédéfinies : ISO 31-1992, ISO 1000 -1992
- Autres attributs associés : min, max, unités affichées
- Exemple : `type CoefficientOfHeatTransfer = Real (final quantity="CoefficientOfHeatTransfer", final unit="W/(m2.K)");`

■ Possibilité de créer des types personnalisés

- Unités personnalisées possibles tant que les exposants sont entiers
- Zones de validité perso, ...

DÉFINITION DES TYPES

ATTRIBUTS POSSIBLES

▪ Principaux préfixes

- 'flow' : définit une variable de type flux dont la connexion impliquera des lois de conservation
 - cf *connecteurs*

□ Préfixes de variabilité dans le temps

- **Variables non modifiables en cours de simulation**

- 'parameter' : instanciable dans l'interface de paramétrage du modèle – une valeur peut être définie
- 'constant' : constante physique ou non - non instanciable via l'interface de paramétrage – une valeur doit obligatoirement être définie

- **Variables discrètes** (non continues) : 'discrete'

□ Préfixes de causalité : 'input' ou 'output'

- Attribut supplémentaire 'final' : prévient toute modification de la variable (unité, valeur, ...)

- 'replaceable' : utilisé pour modifier un élément selon une liste de choix (cf *définition des parois*)

▪ Définition de l'accès d'un groupe

- Accès défini pour tous les éléments suivants le mot-clé dans le code

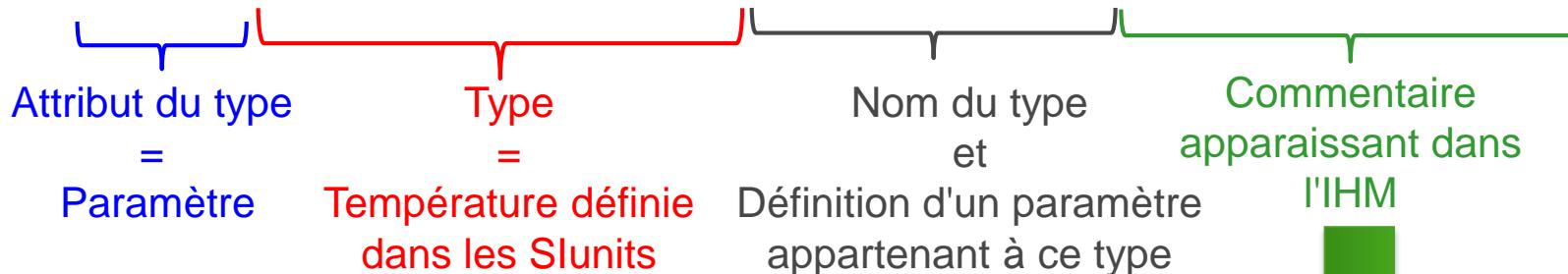
- 'public' : par défaut
- 'protected' : variable interne au modèle, non observable en post-traitement (excepté choix de débogage)

DÉFINITION DES TYPES

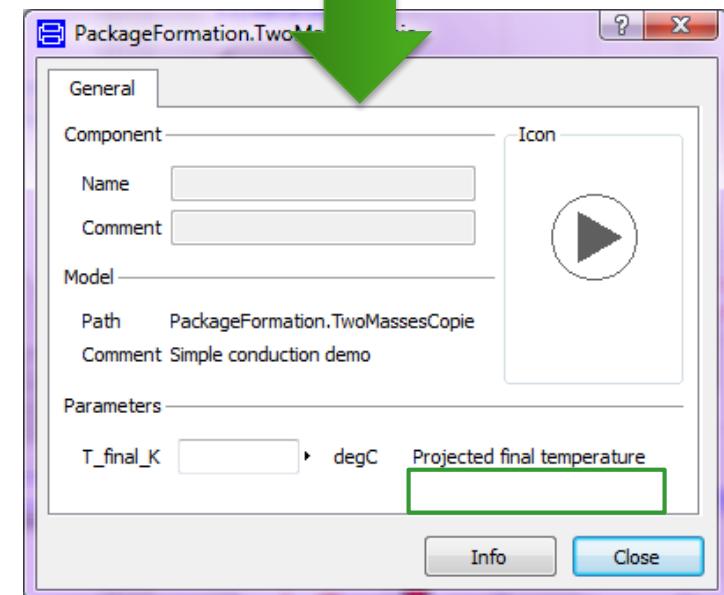
EXAMPLE DE TWOMASSES

- Déclaration des types dans la partie déclarative

- parameter Modelica.SIunits.Temperature T_final_K(fixed=false) "Projected final temperature";



```
model TwoMassesCopie "Simple conduction demo"
  extends Modelica.Icons.Example;
  parameter Modelica.SIunits.Temperature T_final_K(fixed=false)
    "Projected final temperature";
  equation
  initial equation
    T_final_K = (mass1.T*mass1.C + mass2.T*mass2.C) / (mass1.C + mass2.C);
  end;
end TwoMassesCopie;
```



Fenêtre de paramètres du modèle

Formation Modelica – Dymola – BuildSysPro | 02/01/2016 | 47

DÉFINITION D'UN CONNECTEUR

- Type de classe = 'connector'
- Interfaces qui servent à la connexion entre plusieurs modèles
- Peut posséder trois types de grandeurs
 - Variables de flux – de type 'flow' - positif pour un flux entrant dans l'élément
 - Variables de flux de matière en bi-directionnel - 'stream'
 - Variables de type potentiel (sans le mot clé 'flow' ou 'stream')
- Nombre de variables de flux = nombre de variables potentiels

DÉFINITION D'UN CONNECTEUR

LIAISONS ENTRE CONNECTEURS

- Connexion grâce à la fonction 'connect'
 - 'connect' = connexion entre 2 connecteurs seulement
 - Un connecteur peut posséder plusieurs connexions
 - **Génération automatique d'équations**
 - Égalité des potentiels
 - Somme des flux à un nœud égale à zéro
- La connexion peut être **causale** ou **acausale** selon la nature du connecteur
 - Connexion **causale** : variables avec les attributs input ou output
 - Connexion **acausale** : tous les autres connecteurs

DÉFINITION D'UN CONNECTEUR

EXEMPLE DU CONNECTEUR THERMIQUE

- Connecteur en thermique pure (transferts de chaleur)
 - Potentiel = température **T**
 - Flux = flux de chaleur **Q_flow**
- Soit 4 modèles M1 à M4 connectés via leurs connecteurs thermiques respectifs (c1 à c4)

Connexions :

Connect(c1,c2);
Connect(c1,c3);
Connect(c3,c4);



- Les équations automatiques associées sont alors
 - $c1.T = c2.T = c3.T = c4.T$
 - $c1.Q_{\text{flow}} + c2.Q_{\text{flow}} + c3.Q_{\text{flow}} + c4.Q_{\text{flow}} = 0$

DÉFINITION D'UN CONNECTEUR

EXAMPLE DANS TWOMASSES

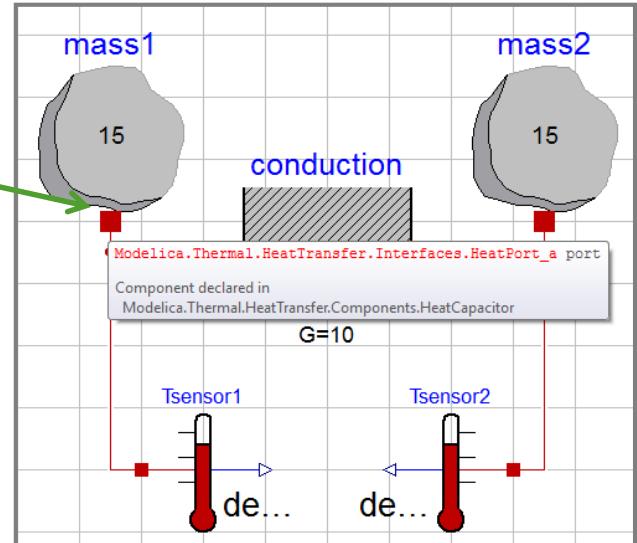
- Sur l'assemblage
- Chercher dans la bibliothèque le

Nature du connecteur visible en passant la souris sur l'élément

chemin du connecteur indiqué

- HeatPort_a est un modèle étendu de HeatPort

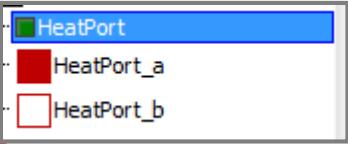
```
connector HeatPort_a
  "Thermal port for 1-dim. heat transfer (filled rectangular icon)"
  extends HeatPort;
  ...
end HeatPort_a;
```



- HeatPort est un connecteur thermique pure sans icône (carré vert par défaut)

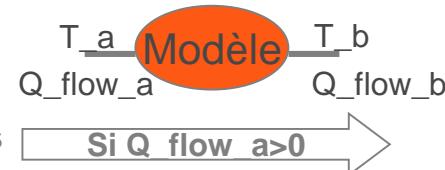
```
partial connector HeatPort "Thermal port for 1-dim. heat transfer"
  Modelica.SIunits.Temperature T "Port temperature";
  flow Modelica.SIunits.HeatFlowRate Q_flow
  "Heat flow rate (positive if flowing from outside into the component"
  ...
end HeatPort;
```

- HeatPort_a a comme seule différence la présence d'un icône rouge



Pourquoi 2 icônes ?

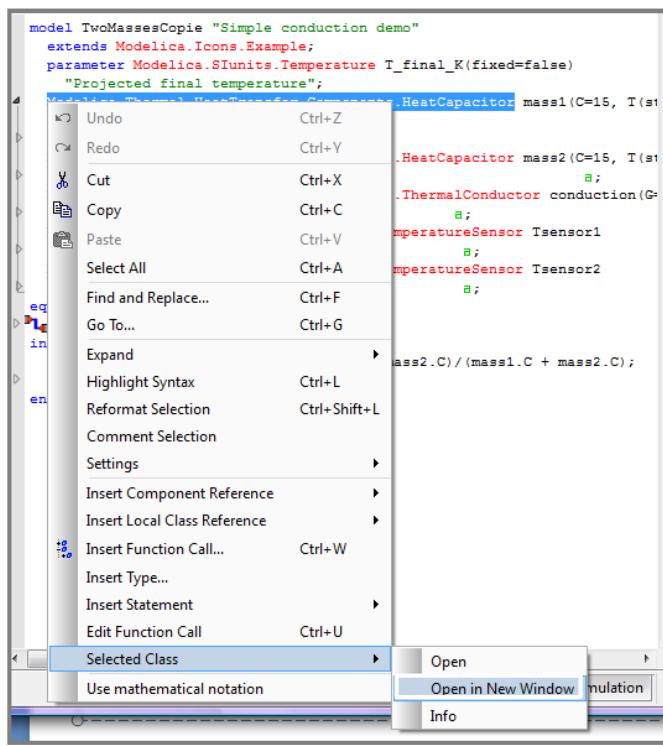
Pour visualiser dans un modèle ayant plusieurs connecteurs la direction du flux entre ces connecteurs



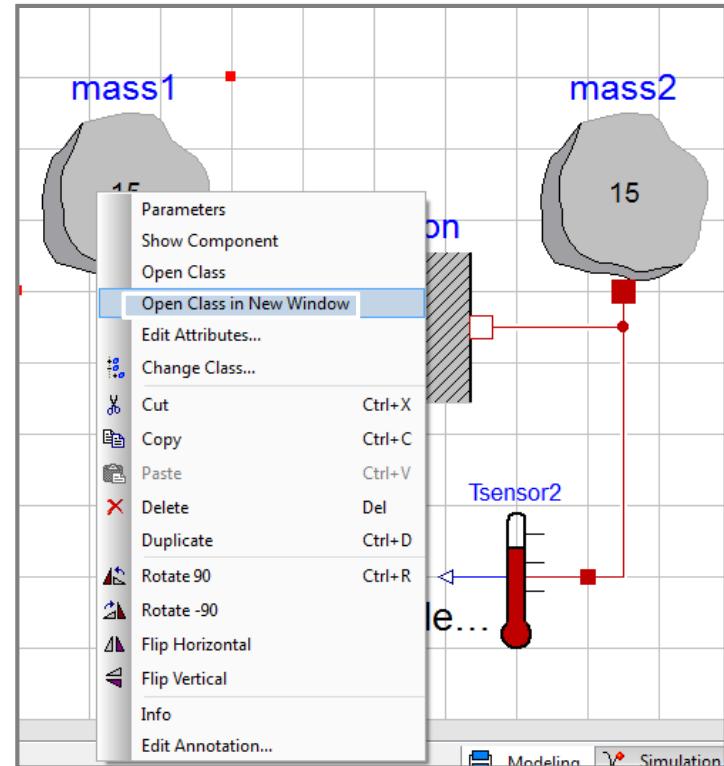
DÉFINITION D'UN CONNECTEUR

EXAMPLE DANS TWOMASSES

- Au lieu de regarder le chemin dans le code : chercher par le code ou l'assemblage en ouvrant successivement tous les éléments jusqu'à atteindre le connecteur
 - Après sélection de l'élément et clic droit sur celui-ci : ouvrir (dans une nouvelle fenêtre)



Dans le code
OU
l'assemblage



L'HÉRITAGE

- Propriété de la programmation orientée objet
 - Un objet possède des propriétés (attributs)
 - Construction d'un objet à partir d'un autre objet (mot clé 'extends')
 - Transmis par héritage : équations, représentation graphique, partie déclarative, ...
 - Ex : Une température aura des unités ($^{\circ}\text{C}$, K), un intervalle (min=0K), ... et tout objet construit à partir de cette température aura les mêmes propriétés

```
type ThermodynamicTemperature = Real (
    final quantity="ThermodynamicTemperature",
    final unit="K",
    min = 0,
    start = 288.15,
    displayUnit="degC")
"Absolute temperature (use type TemperatureDifference for relative temperatures)"
```

▪ Cas des classes définies en 'partial'

- Un modèle partiel ne peut être utilisé que comme modèle parent
 - Il peut lui manquer des équations pour être consistant (nombre inconnus = nombre équations)
 - Il peut ne pas avoir d'icône comme les interfaces (connecteurs) de la bibliothèque Modelica

```
partial connector HeatPort "Thermal port for 1-dim. heat transfer"
  Modelica.SIunits.Temperature T "Port temperature";
  flow Modelica.SIunits.HeatFlowRate Q_flow
    "Heat flow rate (positive if flowing from outside into the component
     ▷
     end HeatPort;
```

GESTION D'ÉVÈNEMENTS

- **2 types d'évènements discrets (instantanés)**
 - Évènement temporel lié aux variables qui évoluent dans le temps
 - Le solveur sait prédire à quel instant aura lieu l'évènement – pas de pénalité de temps de calcul
 - Évènement d'état lié à des variables d'état du modèle sans lien avec le temps
 - Le solveur doit continuellement vérifier si l'évènement peut se produire – pénalise le temps de calcul
- **En pratique, cela se produit dès l'utilisation dans le code**
 - D'opérateurs relationnels (<, >=...),
 - De boucles if ou when
 - De fonctions spécifiques ceil, floor, abs, ...
- **Plusieurs évènements peuvent subvenir en même temps**
 - Attention, il y a alors priorisation des évènements

GESTION D'ÉVÈNEMENTS

PRÉCISION SUR LES ÉVÈNEMENTS D'ÉTAT

- Les équations doivent rester continues et différentiables à tout instant pour permettre leur résolution
 - OR continuité non assurée lors d'opérations conditionnelles (if, when)

- Exemple d'un code menant à une erreur

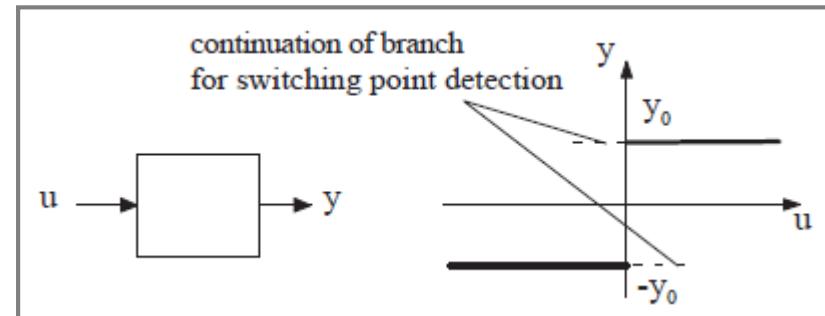
```
y = if u >= 0 then sqrt(u) else 0;
```

Discontinuité en $u=0$ ET fonction racine carrée non définie pour $u<0$

Le solveur peut alors chercher à évaluer \sqrt{u} même si u est négatif car il va osciller autour de l'évènement jusqu'à tomber juste sur cet instant

Code correct :

```
y = if noEvent(u>=0) then sqrt(u) else 0;
```



GESTION D'ÉVÈNEMENTS

CAS PARTICULIER DES BOUCLES IF ET WHEN

Boucle if

- Évaluation des équations en continue

- Les équations font partie du système d'équations global

- Variables calculées de façon continue

Boucle when

- Évaluation des équations **uniquement** aux évènements

- Variables calculées discrètes

- En dehors de l'évènement, les valeurs au pas de temps précédent sont conservées

MANIPULATION DE VECTEURS ET MATRICES

```

model Matrices
    "Exemple d'utilisation des vecteurs et des matrices"
    parameter Real V1[3]={1,2,3} "Définition d'un vecteur";
    parameter Real V2[1,3]=[1,2,3] "Vecteur ligne = matrice (1,:)";

    parameter Real M1[2,3]=[1,2,3;4,5,6] "Définition d'un tableau";
    parameter Real M2[2,3]={{{1,2,3},{4,5,6}}}
        "Définition d'une matrice / équivalent de M1";

    parameter Real V3[:]=1:5 "Génération du vecteur {1,2,3,4,5}";
    parameter Real V4[:]=1:2:7 "Génération du vecteur {1,3,5,7}";

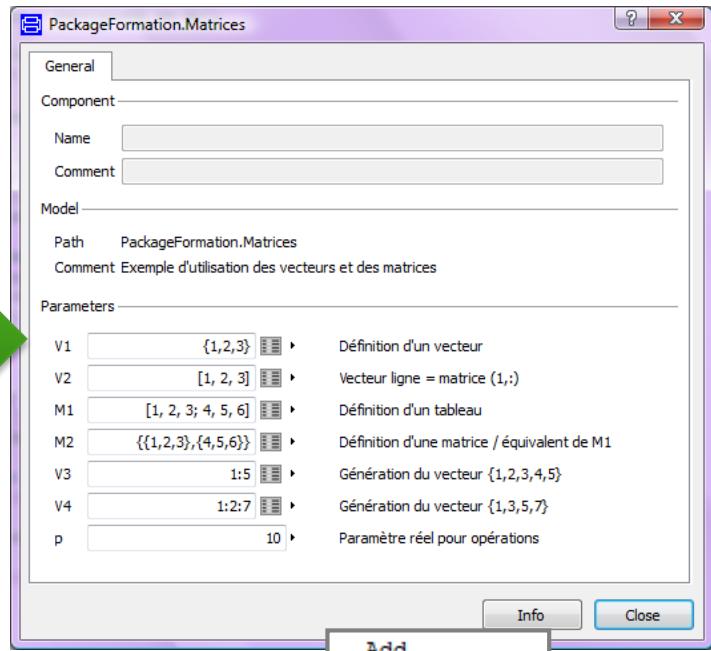
    parameter Real p=10 "Paramètre réel pour opérations";

    // Variables calculées dans la partie 'équation'
    // Attention, Dymola ne gère pas encore les matrices de taille inconnue
    Real Add[2,3] "Addition M1+M2";
    Real Mult[2,3] "Multiplication p * M1";
    Real TranspV1[1,3] "V1'";
    Real TailleM1[2] "Taille de la matrice M1";
    Real Matr1[2,3] "Multiplication M1*M2 éléments par éléments";
    Real Matr2[2,1] "Multiplication M1*V2'";
    Real Extr "Extraction M1[2,3]";
    Real ExtrV[2] "Extraction M1[1:2,2]";

equation
    // Addition / Soustraction
    Add = M1 + M2;
    // Multiplication par un scalaire
    Mult = p * M1;
    // Opérateurs particuliers
    TranspV1 = transpose([V1]);
    TailleM1=size(M1);
    // Multiplication de vecteurs et matrices
    Matr1 = M1.*M2;
    Matr2 = M1 * transpose(V2);
    // Extractions d'éléments
    Extr=M1[2,3];
    ExtrV = M1[1:2,2];
end Matrices;

```

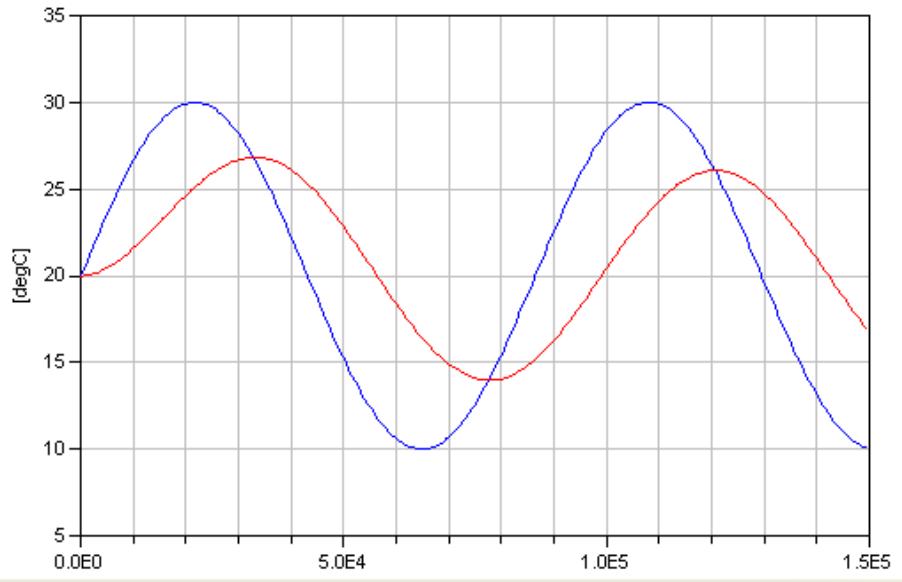
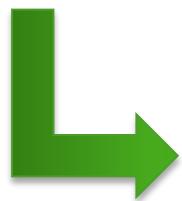
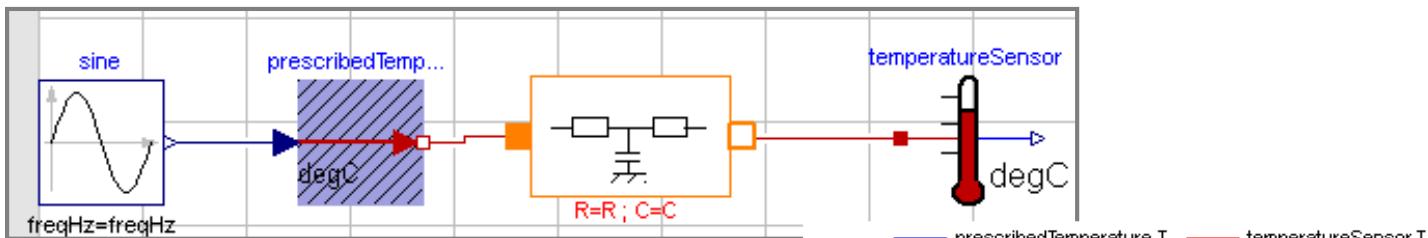
Fenêtre 'paramètres'



$V_1 = \{1, 2, 3\}$
 $V_2 = [1 \ 2 \ 3]$
 $M_1 = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$
 $M_2 = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$
 $V_3 = \{1, 2, 3, 4, 5\}$
 $V_4 = \{1, 3, 5, 7\}$
 $p = 10$

Résultats

Add	$\begin{bmatrix} 2 & 4 & 6 \\ 8 & 10 & 12 \end{bmatrix}$
Mult	$\begin{bmatrix} 10 & 20 & 30 \\ 40 & 50 & 60 \end{bmatrix}$
TranspV1	$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$
TailleM1	$\{2.0, 3.0\}$
Matr1	$\begin{bmatrix} 1.0 & 4.0 & 9.0 \\ 16.0 & 25.0 & 36.0 \end{bmatrix}$
Matr2	$\begin{bmatrix} 14.0 \\ 32.0 \end{bmatrix}$
Extr	$= 6$
ExtrV	$\{2, 5\}$

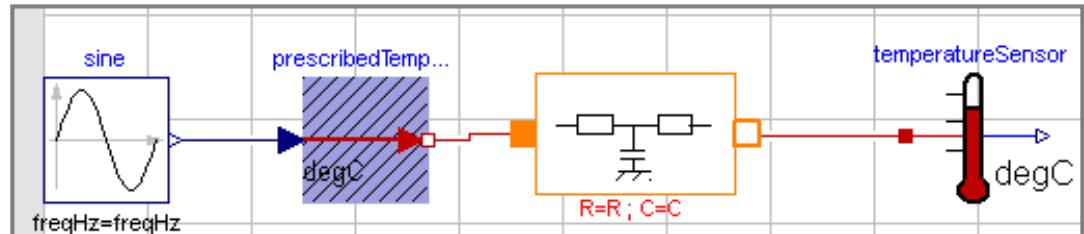


Exercices d'application ENVIRONNEMENT DYMOLA / LANGAGE MODELICA

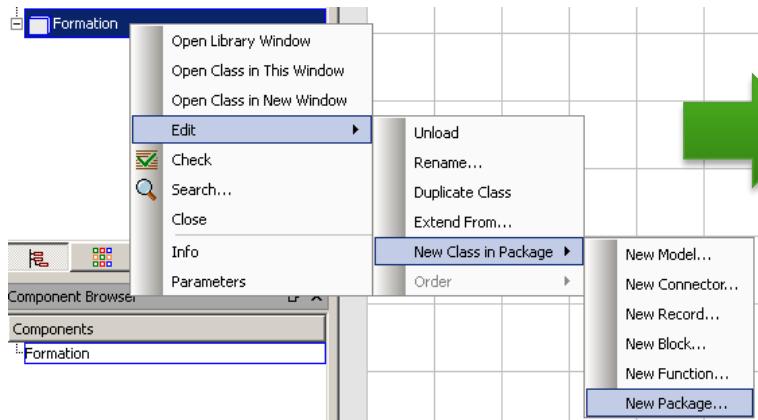
PAROI R2C1

■ Objectifs

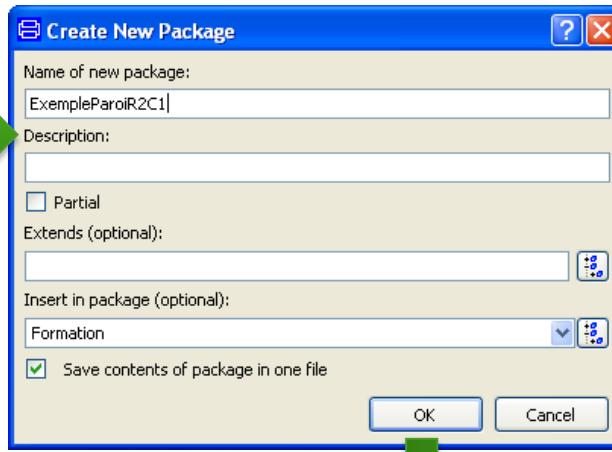
- Créer entièrement une paroi R2C1
- Simuler son comportement à une excitation sinusoïdale



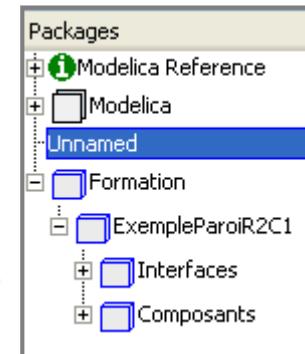
■ 1. Créer des nouveaux packages pour héberger tous les modèles



Clic droit sur le package
Formation



Répéter l'opération pour
avoir l'arborescence
suivante



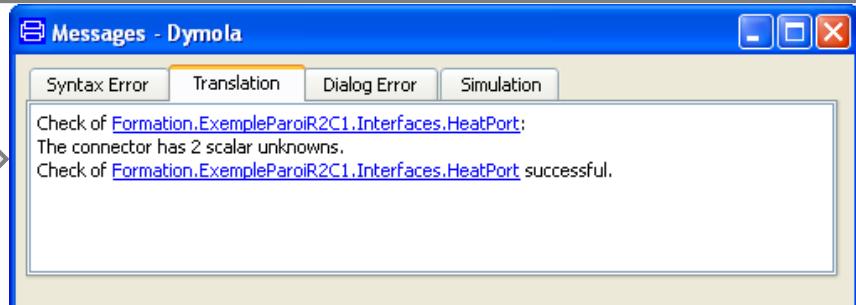
PAROI R2C1

- 2.Créer les connecteurs thermiques dans le package Interfaces
 - Rque : ils existent dans la bibliothèque Modelica, ici on va les créer à partir de zéro pour comprendre les mécanismes
 - Rappel : Connecteur thermique
 - **Potentiel** = température **T**
 - **Flux** = flux de chaleur **Q_flow**
 - On crée 1 connecteur partiel *HeatPort* qui sera ensuite décliné avec 2 icônes différentes

Ecrire le code suivant dans l'onglet 'Modelica text'

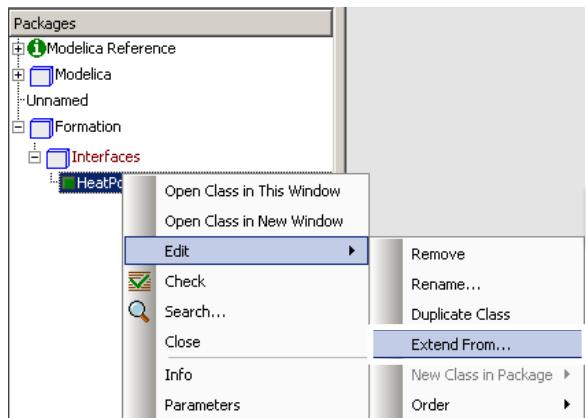
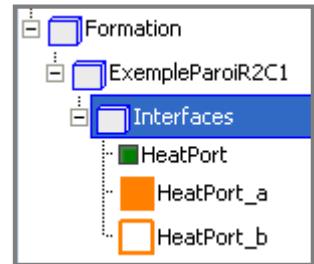
```
partial connector HeatPort "Port thermique"
  Modelica.SIunits.Temperature T "Port température";
  flow Modelica.SIunits.HeatFlowRate Q_flow "flux de chaleur (positif si reçu par le système,
négatif si cédé par le système)";
end HeatPort;
```

Vérifier qu'il n'y a pas
d'erreur de syntaxe avec
le 'check'

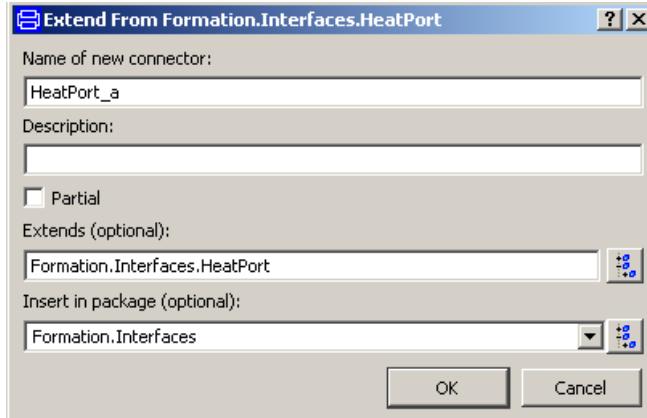


PAROI R2C1

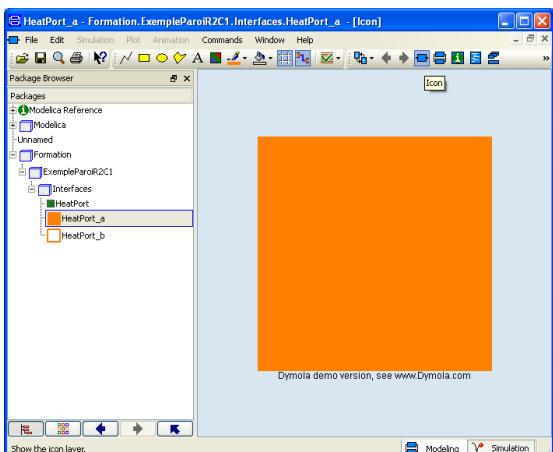
- On décline *HeatPort* en 2 icônes différentes grâce à l'option 'extends'



Clic droit sur le connecteur *HeatPort*



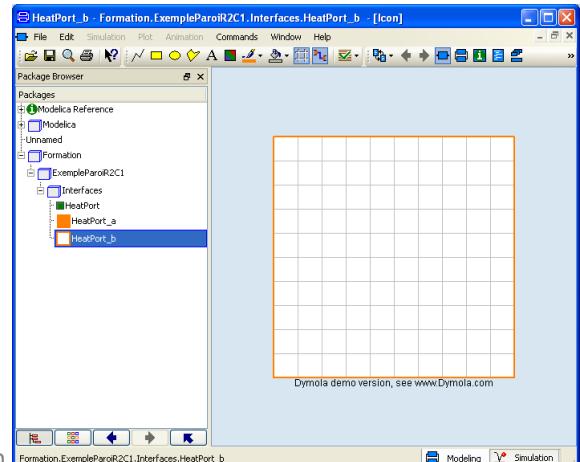
Ouvrir l'onglet « Icon » et y dessiner des icônes différentes



Formation

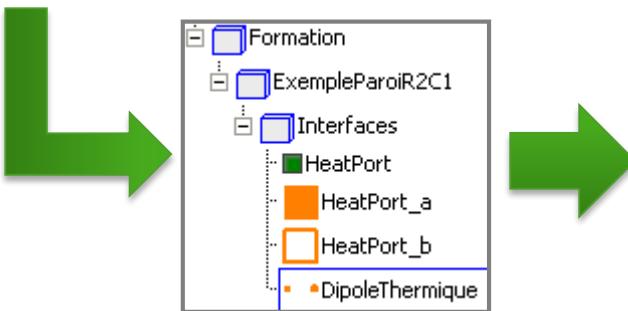
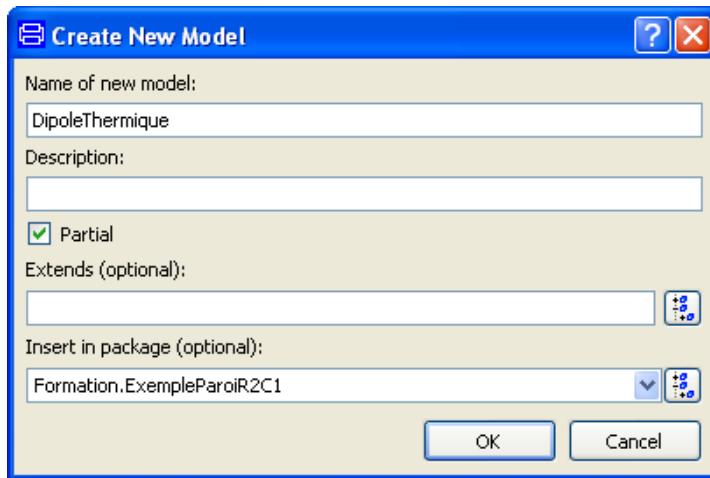
Code généré automatiquement

```
connector HeatPort_a  
    extends HeatPort  
end HeatPort_a;
```

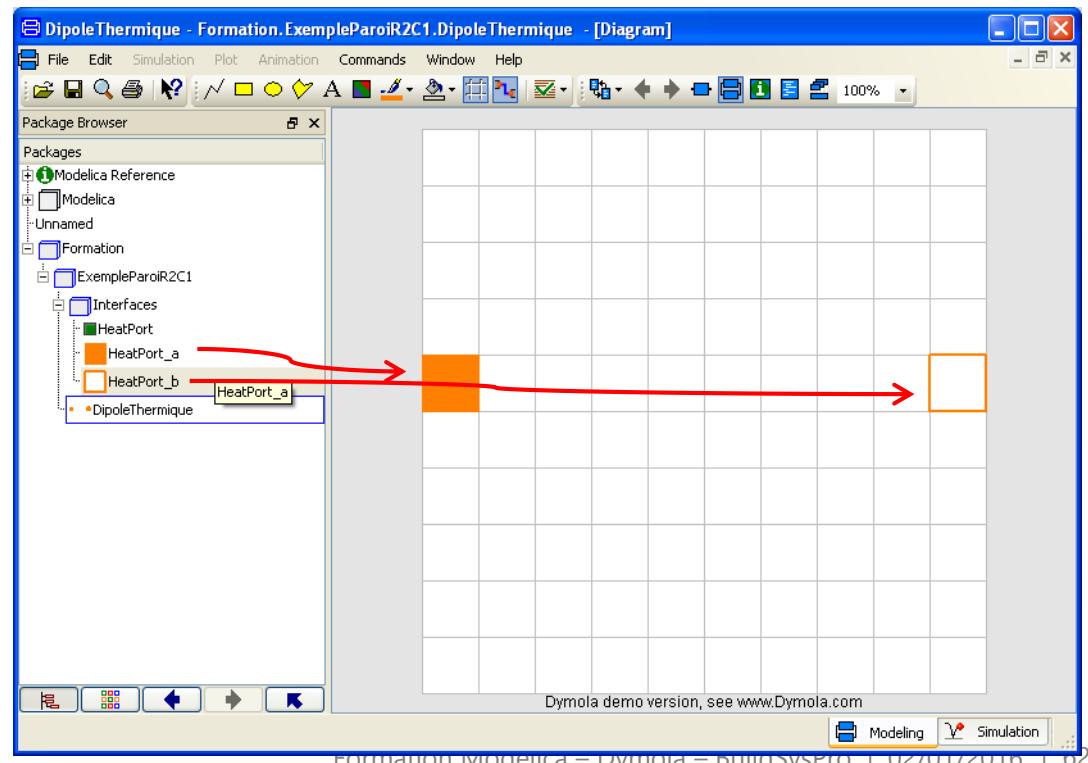


PAROI R2C1

- 3. Créer un dipôle thermique qui est une base des modèles thermiques (conduction, convection, rayonnement)



Glisser-déposer les HeatPort_a et HeatPort_b dans le diagramme (onglet 'Diagram')



PAROI R2C1

Code automatique généré par Dymola

The screenshot shows the Dymola interface with the following components:

- Top Bar:** File, Edit, Simulation, Plot, Animation, Commands, Window, Help.
- Toolbar:** Standard icons for file operations, search, and zoom.
- Package Browser:** Shows the project structure:
 - Packages: Modelica Reference, Modelica, Unnamed, Formation, ExempleParoiR2C1, Interfaces.
 - ExempleParoiR2C1 contains: Interfaces, HeatPort_a, HeatPort_b, DipoleThermique.
- Central Editor:** Displays the generated Modelica code for the DipoleThermique model.

```
partial model DipoleThermique
  Interfaces.HeatPort_a heatPort_a;
  Interfaces.HeatPort_b heatPort_b;
end DipoleThermique;
```

A large green arrow points from the Package Browser to the code editor with the text "Ajout des équations et variables".

```
partial model DipoleThermique
  Modelica.SIunits.HeatFlowRate Q_flow "Flux de chaleur de port_a -> port_b";
  Modelica.SIunits.TemperatureDifference dT "port_a.T - port_b.T";

  Interfaces.HeatPort_a heatPort_a;
  Interfaces.HeatPort_b heatPort_b;

  equation
    dT = heatPort_a.T - heatPort_b.T;
    heatPort_a.Q_flow = Q_flow;
    heatPort_b.Q_flow = -Q_flow;
end DipoleThermique;
```
- Messages Window:** Shows the build status:

Check of Formation.ExempleParoiR2C1.Interfaces.DipoleThermique:
The model has 6 scalar unknowns and 5 scalar equations.
Check of Formation.ExempleParoiR2C1.Interfaces.DipoleThermique successful.
- Right Panel:** **Modèle dipôle**

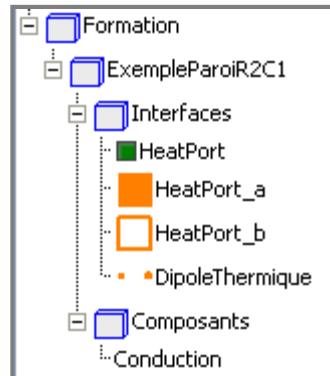
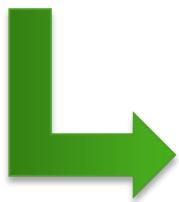
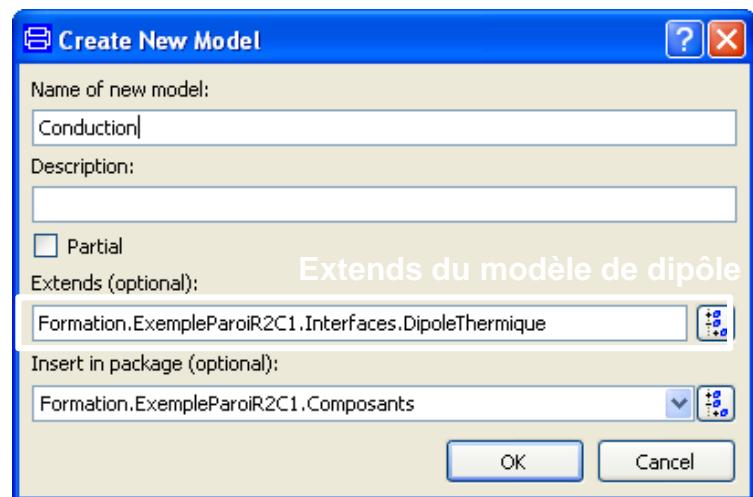
Variables : Q_flow et dT internes au modèle

Relations entre ces 2 variables et les connecteurs à donner dans la partie 'équation'

Check : il manque 1 équation pour que le modèle soit complet

PAROI R2C1

- 4. Créer un modèle conductif représentant la partie résistive (R) de la paroi R2C1



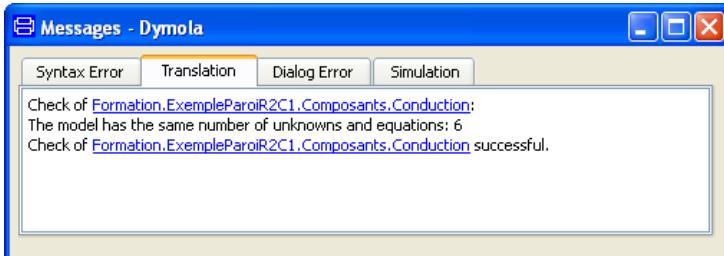
Code automatique généré par Dymola

```
model Conduction
  extends Interfaces.DipoleThermique;
end Conduction;
```

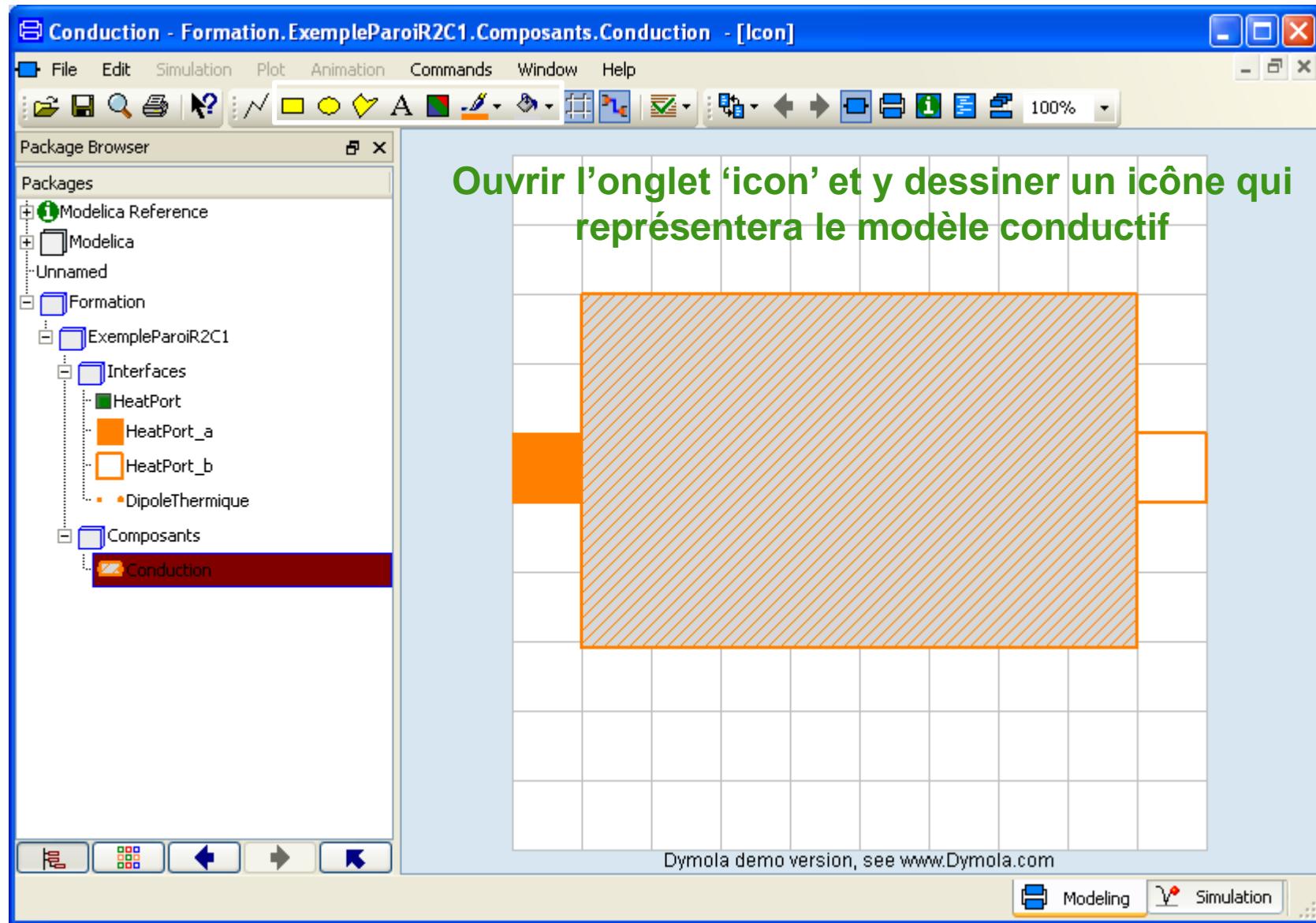
Ajout de paramètres et équations

```
model Conduction
  extends Interfaces.DipoleThermique;
parameter Modelica.SIunits.ThermalConductance G "Conductance";
equation
  Q_flow = G*dT;
  a;
end Conduction;
```

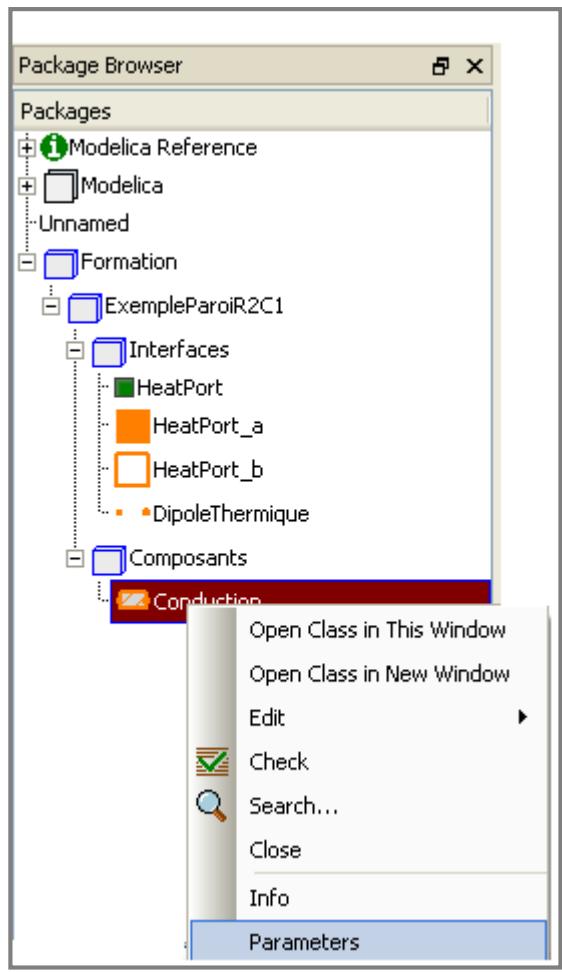
Check : il y a désormais le même nombre d'équations que d'inconnues



PAROI R2C1

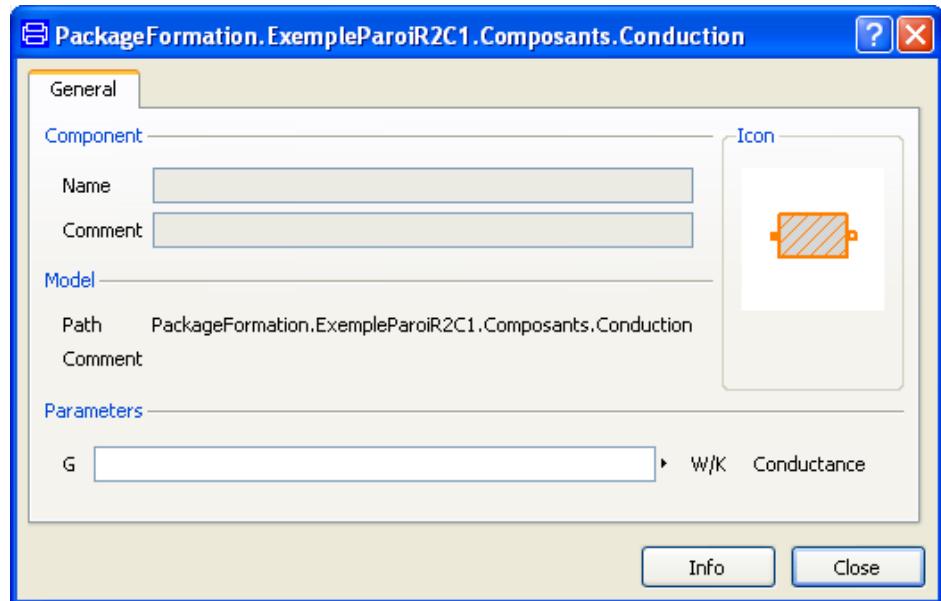


PAROI R2C1



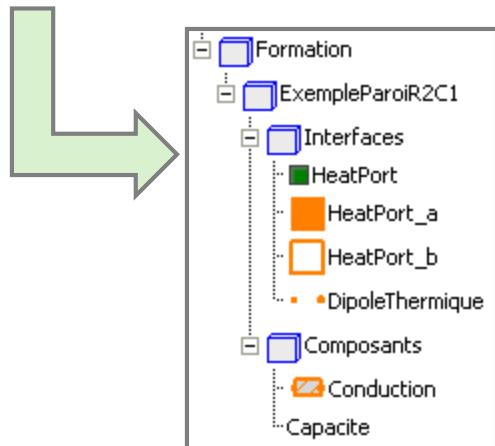
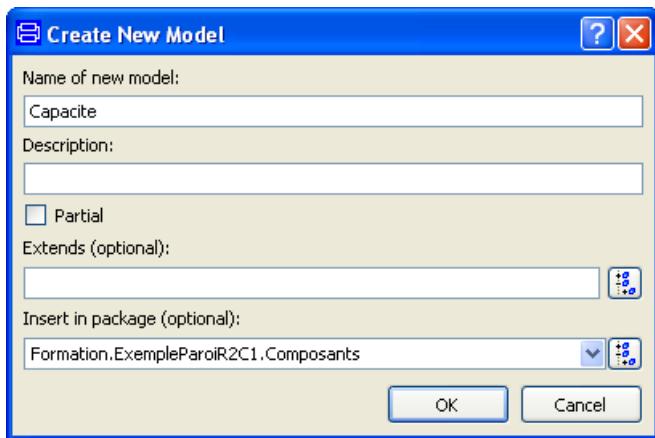
```
model Conduction
  extends Interfaces.DipoleThermique;
parameter Modelica.SIunits.ThermalConductance G "Conductance";
equation
  Q_flow = G*dT;
  ...
end Conduction;
```

Paramètres
du modèle



PAROI R2C1

- 5. Créer un modèle de capacité thermique représentant la partie capacitive (C) de la paroi R2C1



Ajout d'un connecteur HeatPort_a par glisser-déplacer dans l'onglet 'Diagram'

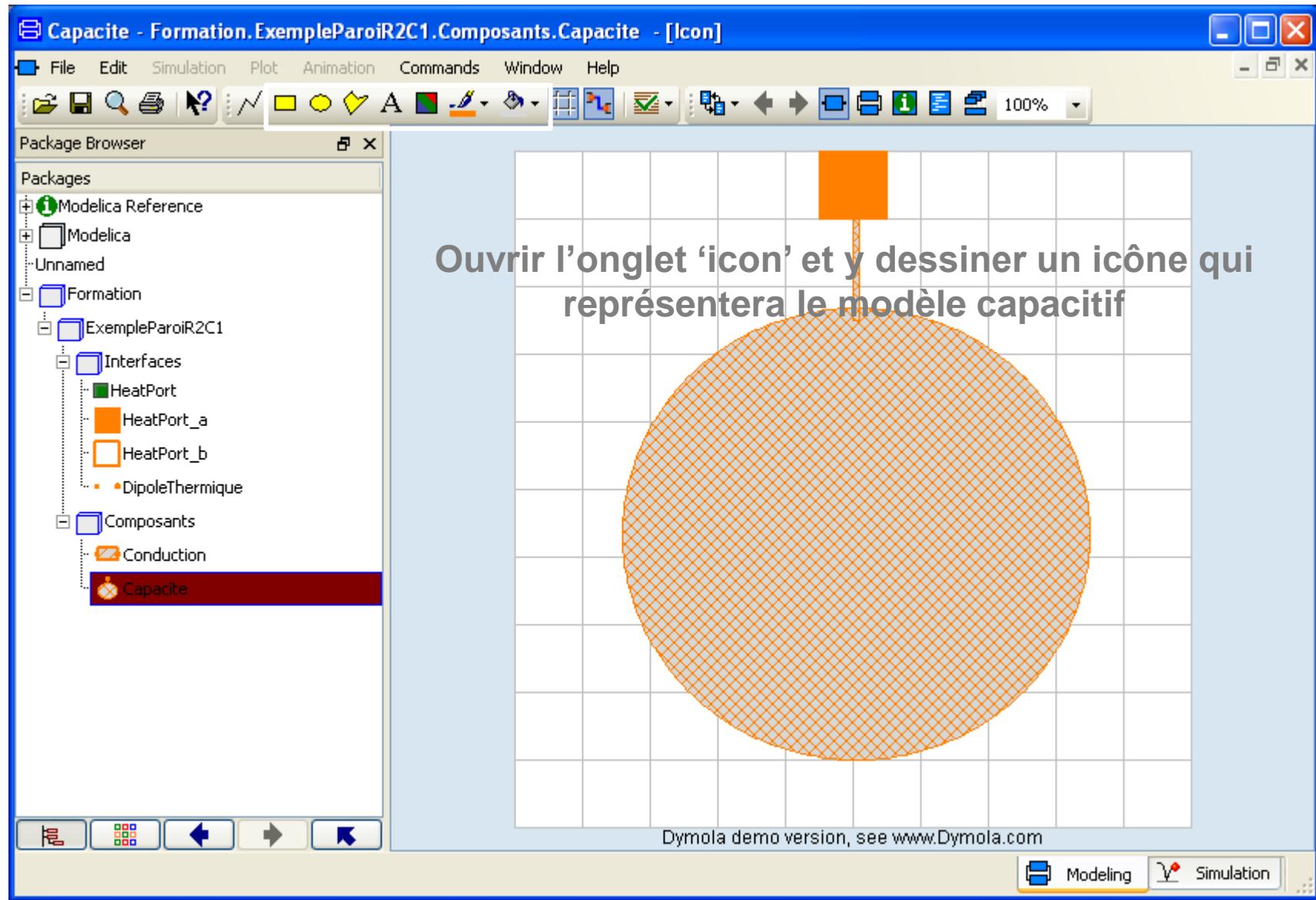
Ajout des paramètres et équations dans l'onglet 'Modelica Text'

```
model Capacite
  parameter Modelica.SIunits.HeatCapacity C "Capacité thermique (= cp*m)" ;
  parameter Modelica.SIunits.Temperature Tinit(displayUnit="degC")=293.15
    "Température initiale du noeud";
  Modelica.SIunits.Temperature T(start=Tinit, displayUnit="degC")
    "Température du noeud";
  Interfaces.HeatPort_a heatPort_a
  ;

equation
  T = heatPort_a.T;
  C*der(T) = heatPort_a.Q_flow;
  ;
end Capacite;
```

Vérifier le modèle – commande Check

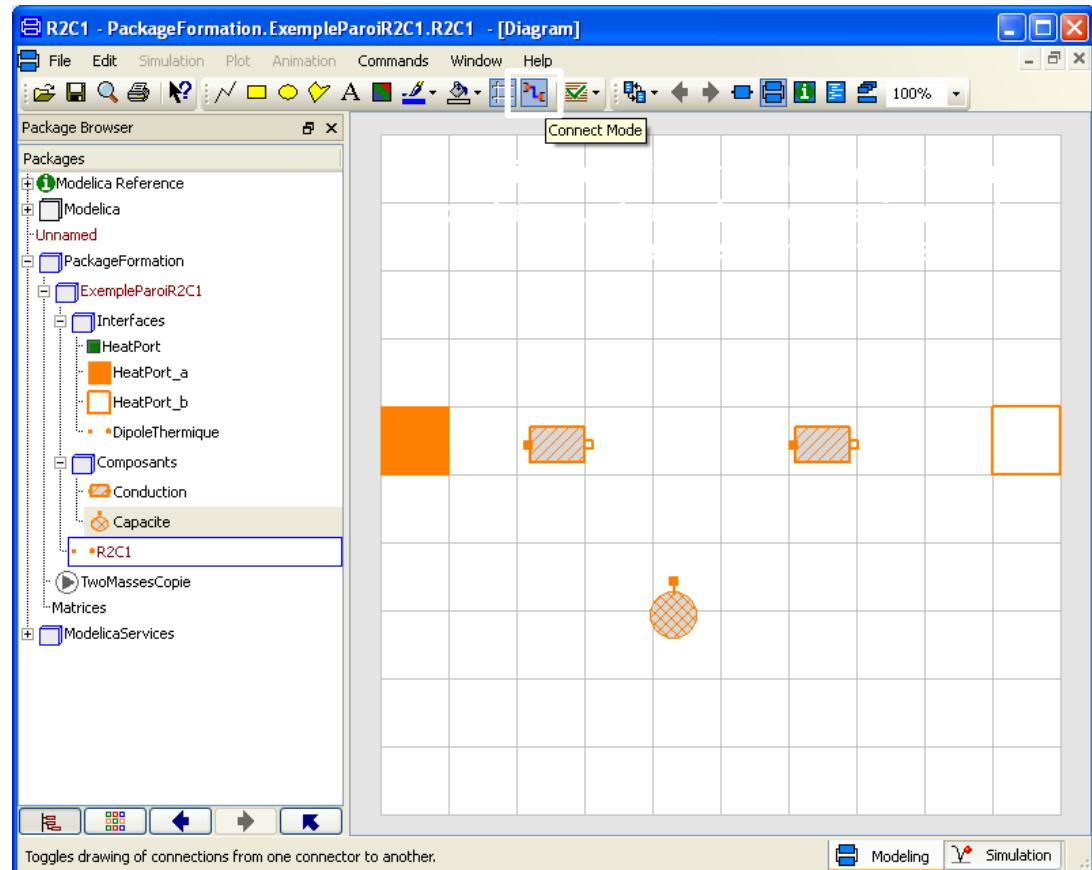
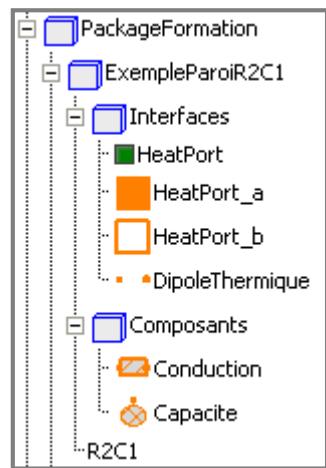
PAROI R2C1



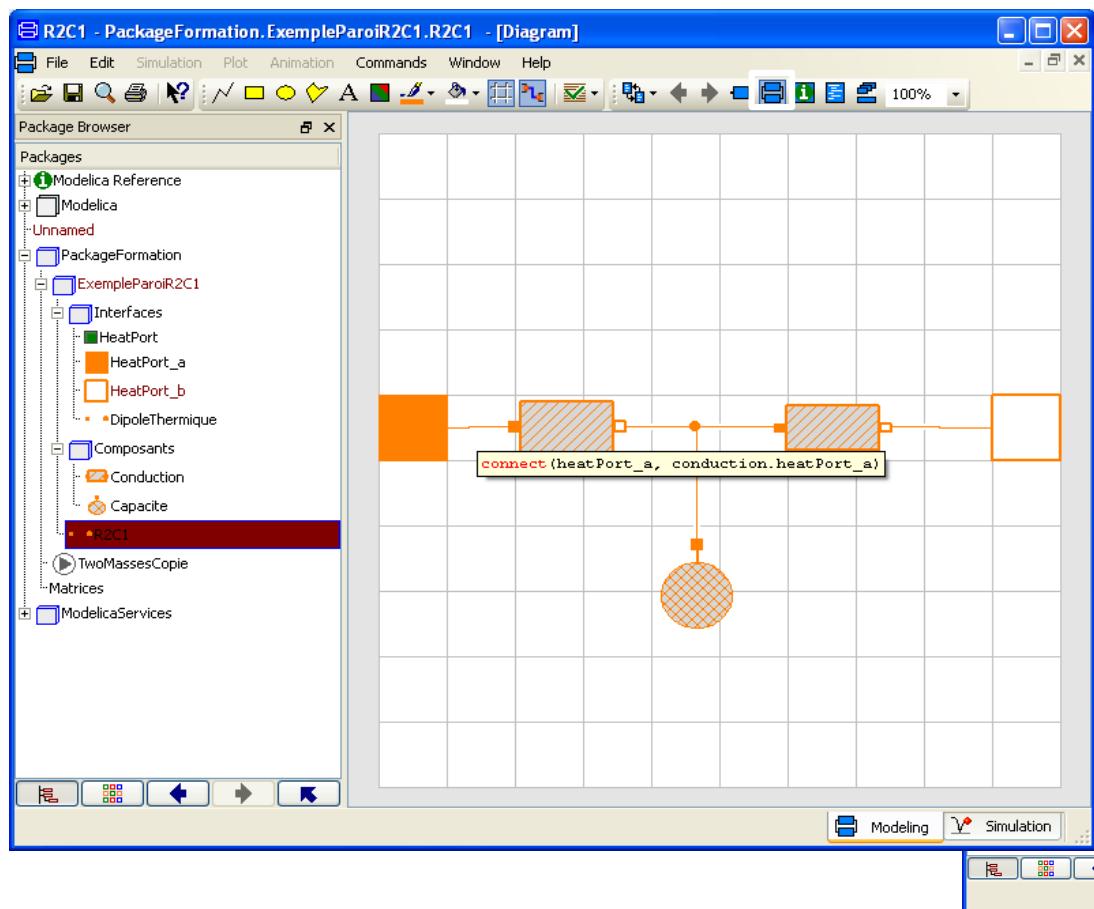
PAROI R2C1

▪ 6. Assembler les 2 résistances et la capacité

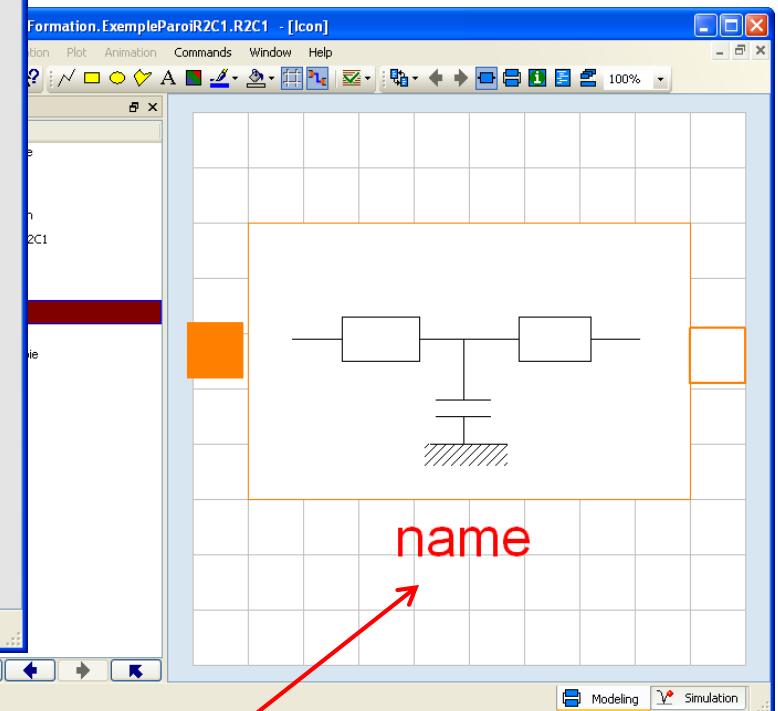
Ajout des éléments par glisser-déplacer dans l'onglet 'Diagram' : 2 connecteurs, 2 résistances, 1 capacité



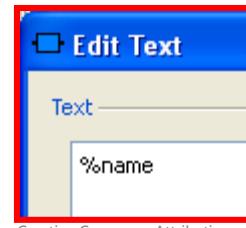
PAROI R2C1



Créer une icône pour ce modèle de paroi R2C1



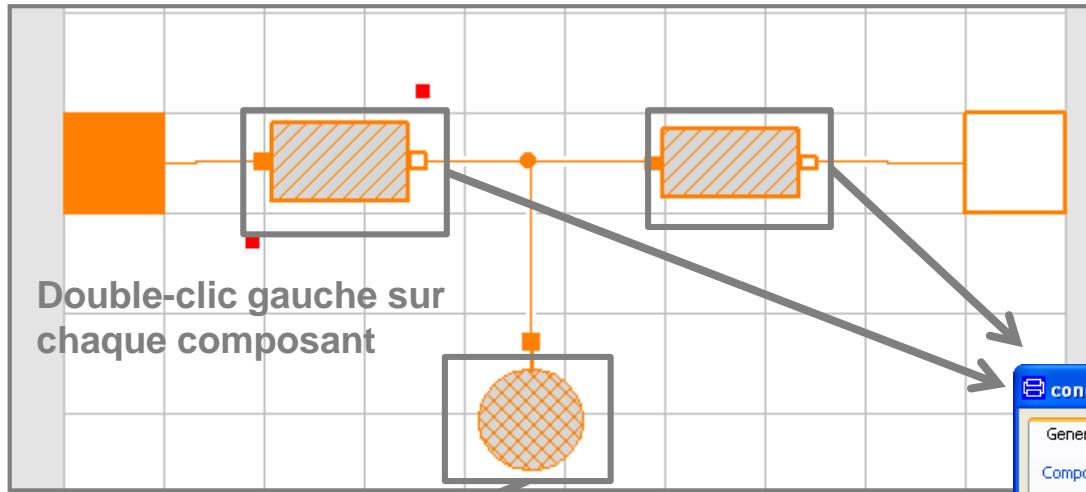
Lors de l'utilisation de ce modèle, il sera indiqué visuellement le nom de l'élément



PAROI R2C1

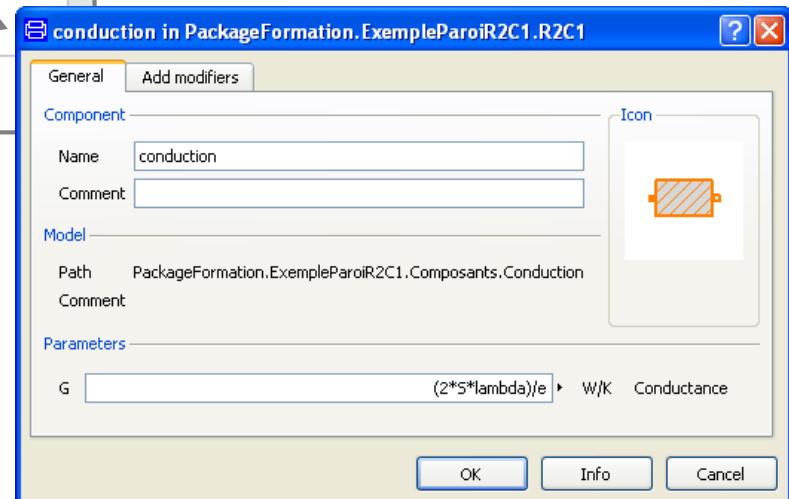
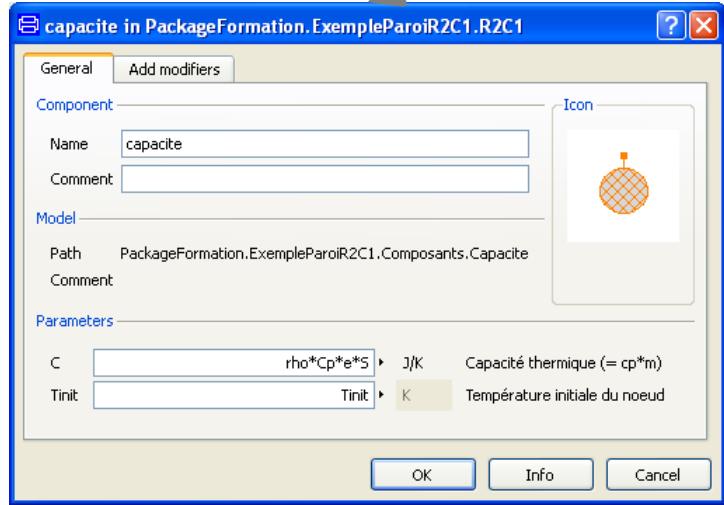
$$\rho \cdot cp \cdot \frac{dT}{dt} = \lambda \cdot \Delta T$$

Dans le code : paramétrer le modèle en remontant les paramètres des modèles RC



Flux dans chacune des résistances

$$Q = \frac{2 \cdot \lambda \cdot S}{e} \cdot \Delta T$$



Flux dans la capacité thermique

$$Q = \rho \cdot cp \cdot e \cdot S \frac{dT}{dt}$$

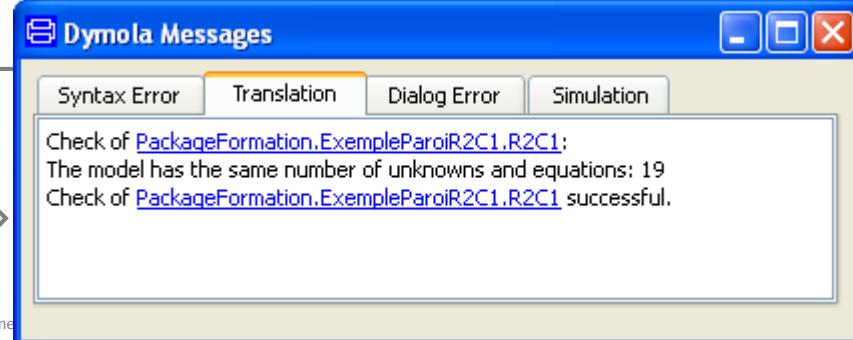
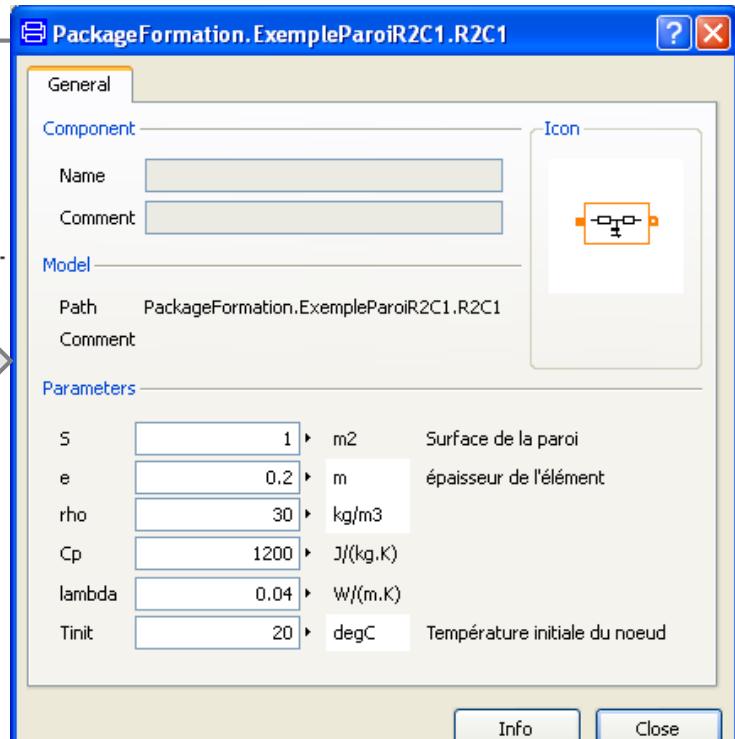
PAROI R2C1

Dans le code : paramétrer le modèle en remontant les paramètres des modèles RC

```
model R2C1
parameter Modelica.SIunits.Area S=1 "Surface de la paroi";
parameter Modelica.SIunits.Length e=0.2 "épaisseur de l'élément";
parameter Modelica.SIunits.Density rho=30;
parameter Modelica.SIunits.SpecificHeatCapacity Cp=1200;
parameter Modelica.SIunits.ThermalConductivity lambda=0.04;
parameter Modelica.SIunits.Temperature Tinit(displayUnit="degC")=293.
    "Température initiale du noeud";

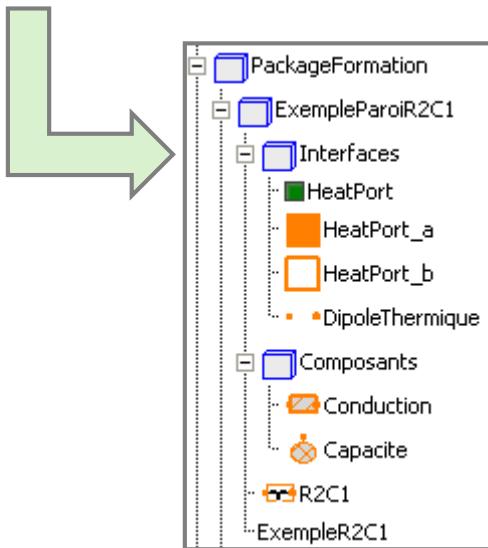
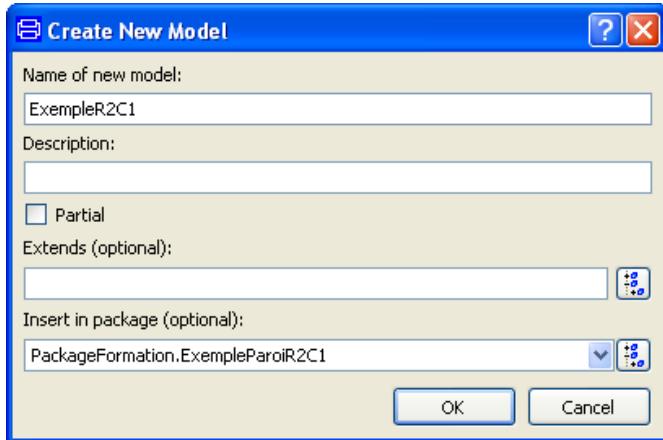
Composants.Conduction conduction(G=(2*S*lambda)/e)
    a;
Composants.Conduction conductionl(G=(2*S*lambda)/e)
    a;
Composants.Capacite capacite(C=rho*Cp*e*S, Tinit=Tinit)
    a;
Interfaces.HeatPort_a heatPort_a
    a;
Interfaces.HeatPort_b heatPort_b
    a;
equation
    connect(heatPort_a, conduction.heatPort_a) a;
    connect(heatPort_b, conductionl.heatPort_b) a;
    connect(capacite.heatPort_a, conduction.heatPort_b) a;
    connect(capacite.heatPort_a, conductionl.heatPort_a) a;
    a;
end R2C1;
```

Paramètres
du modèle



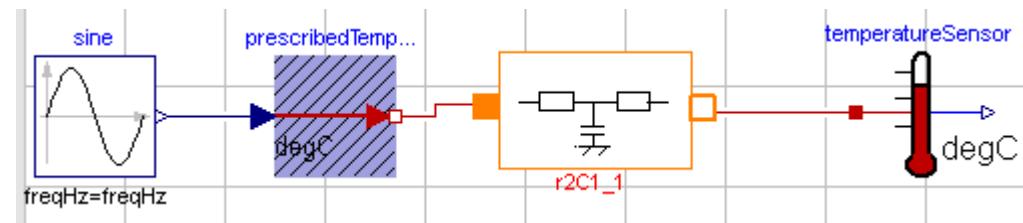
PAROI R2C1

■ 7. Simuler le modèle R2C1 avec une sollicitation sinusoïdale

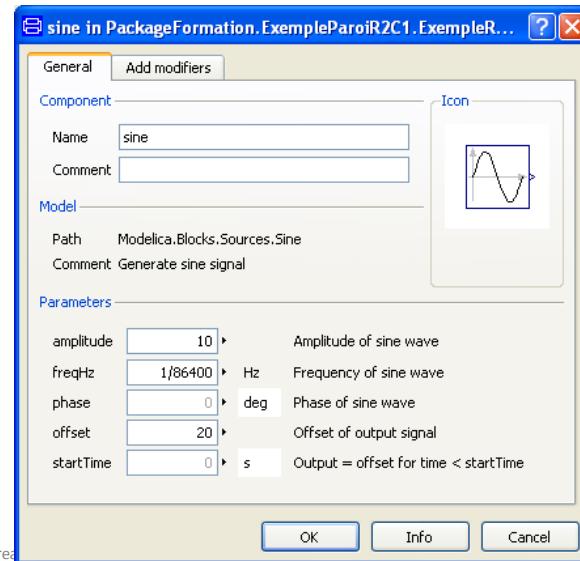


Ajout des éléments par glisser-déplacer dans l'onglet 'Diagram'

Chercher les modèles dans la bibliothèque Modelica



Paramétrier le modèle 'sine' afin que la simulation tourne



PAROI R2C1

Aller dans l'onglet 'Simulation' et configurer les paramètres de la simulation

Visualiser la température en entrée et sortie de la paroi R2C1

