WORKSHOP 4.2

Creating a data architecture

Contents

Introduction

In this workshop, you will continue to build the simple car model library. A simple data structure will be introduced, and you will practice managing your library, setting configurations, and managing the data flow.

There is quite a lot of infrastructure to set up before you can start simulating your model, you can skip the Transmission data and Driveline data throughout this tutorial and add it later if you want to get to the finish line quicker.

Creating the packages

1. Create a new package **W2_DataHandling** inside the course by duplicating the whole package **W1_Reconfigurable** created in the previous session (This has already been done for you in the TrainingPack given. You can start working with **W2_DataHandling** package directly).

Now we are going to add a package structure to contain the data for the car model. This structure will also resemble the different parts of the model. The idea is to be able to change one data record to switch between data sets to model say a *V4 engine* or *V8 engine*.

2. Inside this **W2_DataHandling**, create a **Data** sub-package.
3. Inside the *Data* package create the following packages:
    - **Templates** record classes for data structure
    - **EngineData** records containing engine data
    - **TransmissionData** records containing transmission data
    - **DrivelineData** records containing driveline data
    - **ChassisData** records containing chassis data
    - **VehicleData** records containing full vehicle data

    You should now have a library looking like Figure 1.

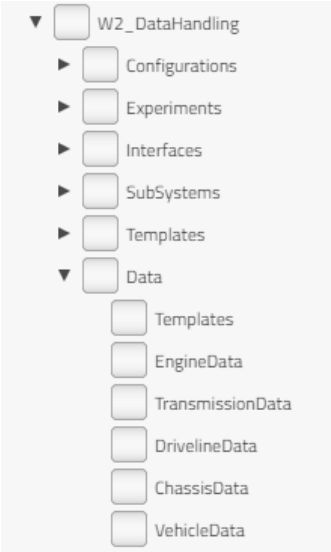


Figure 1 Data package structure

Creating the Records

You will create four records for each sub model of the vehicle. These records will only contain parameter declarations but without any default values. First go back to the *AccelerationTest* model and look at the parameter dialog for each of the four vehicle subcomponents to get an idea of the data needed to describe each subsystem.

1. In the *Data.Templates* package, create four records called *EngineData*, *TransmissionData*, *DrivelineData* and *ChassisData*, containing the needed parameters to describe each subsystem (also see below). Add the keyword partial to all the record classes. The code and record package should look like Figure 2.

```
partial record ChassisData "Empty ChassisDataRecord"
    extends .Modelica.Icons.Record;
    parameter .Modelica.Units.SI.Inertia J_wheel "Wheel inertia";
    parameter .Modelica.Units.SI.Distance R_wheel "Wheel radius";
    parameter .Modelica.Units.SI.Mass m_chassis "mass of the chassis";
end ChassisData;
partial record DrivelineData "Empty DrivelineDataRecord"
    extends .Modelica.Icons.Record;
    parameter .Modelica.Units.SI.Inertia J "Moment of inertia";
end DrivelineData;
partial record EngineData "Empty EngineDataRecord"
    extends .Modelica.Icons.Record;
    parameter .Modelica.Units.SI.Torque max_torque "Torque for full throttle";
end EngineData;
partial record TransmissionData "Empty TransmitionDataRecord"
    extends .Modelica.Icons.Record;
    parameter Real ratio "Transmission ratio (flange_a.phi/flange_b.phi)";
end TransmissionData;
partial record VehicleData "empty complete vehicle data record using replaceable components"
    extends .Modelica.Icons.Record;
    replaceable EngineData engineData constrainedby EngineData annotation(...);
    replaceable TransmissionData transmissionData constrainedby TransmissionData annotation(...);
    replaceable DrivelineData drivelineData constrainedby DrivelineData annotation(...);
    replaceable ChassisData chassisData constrainedby ChassisData annotation(...);
end VehicleData;
```

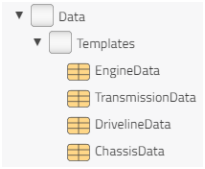


Figure 2 Code and Package structure

Creating the VehicleData Record

The idea of a complete Vehicle data record is to be able to collect all data from sub-records in one place. To be able to easily interchange sub records we need to make them replaceable.

1. In the *Data.Templates* package, create a new *VehicleData* record.

2. Open the source code to *VehicleData* and add the keyword partial.
3. In the diagram layer, drag and drop all the subcomponent records.
   The *VehicleData* record should be look like Figure 3, and the replaceable *EngineData* code should be look like Figure 4.

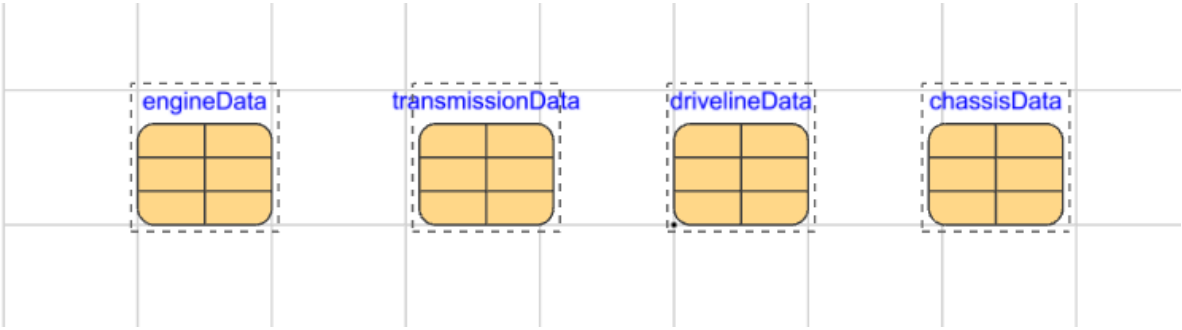


Figure 3 VehicleData record

```
replaceable .Day4.DataHandling.Data.Templates.EngineData engineData
    constrainedby .Day4.DataHandling.Data.Templates.EngineData annotation(...);
```

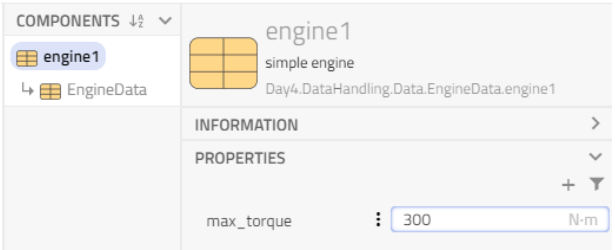
Figure 4 Replaceable enginedata code

Creating the Data Records

Now that the record structure is completed we can start creating the records containing the actual data.

Create two engine variations in the *Data.EngineData* package, one with *max_torque*=300 and one with *max_torque*= 500.

1. Start by extending the *Templates.EngineData* record, name it *engine1* and place it in the *EngineData* package.
2. Go to the *engine1* record and look in the *ComponentBrowser* and fill in the data *max_torque*=300



   Continue to create at least one of each *Transmission*, *Driveline* and *Chassis* data records with the old default values. The code must be like Figure 5.

```
package TransmissionData
  record transmission1 "simple transmission"
    extends .Day4.DataHandling.Data.Templates.TransmissionData(
        ratio=1.0);
  end transmission1;
end TransmissionData;

package DrivelineData
  record driveline1 "simple driveline"
    extends .Day4.DataHandling.Data.Templates.DrivelineData(J=
          0.1);
  end driveline1;
end DrivelineData;

package ChassisData
  record chassis1 "simple chassis"
    extends .Day4.DataHandling.Data.Template.ChassisData(
      J_wheel=1,
      R_wheel=0.3,
      m_chassis=1400);
  end chassis1;
end ChassisData;
```

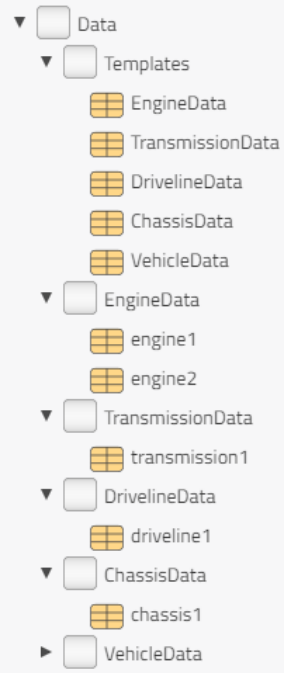


Figure 5 Code for record (*Transmission, Driveline, Chassis*) and full library structure

Now to create a *VehicleData* record all we need to do is to use the sub-records just created.

3. Create a *VehicleData* record, by extending the template. When we redeclare the records it is also necessary to add the replaceable keyword if we want to be able to look into the records in the parameter dialog of the record (see Figure 6).
4. If you used the simplified version of the vehicle record, just extend that record, and then redeclare each sub-record. The replaceable keyword can be added by double clicking to view parameter dialog and switch to *Attributes* tab. Check *Replaceable*.

```
record vehicle1
    extends .Day4.DataHandling.Data.Templates.VehicleData(
        redeclare replaceable .Day4.DataHandling.Data.EngineData.engine2 engineData,
        redeclare replaceable .Day4.DataHandling.Data.TransmissionData.transmission1 transmissionData,
        redeclare replaceable .Day4.DataHandling.Data.DrivelineData.driveline1 drivelineData,
        redeclare replaceable .Day4.DataHandling.Data.ChassisData.chassis1 chassisData);
end vehicle1;
```

Figure 6 Code for vehicle1 record

Adding the Data Record to the template

Go back to the template for the complete vehicle model.

5. Drag the *Data.Templates.VehicleData.vehicle1* into the *Templates.StandardCar*, call the instance name **data** and make it replaceable and a parameter. Rename by double clicking the component and edit the name in the parameter dialog. The template should be look like Figure 7.

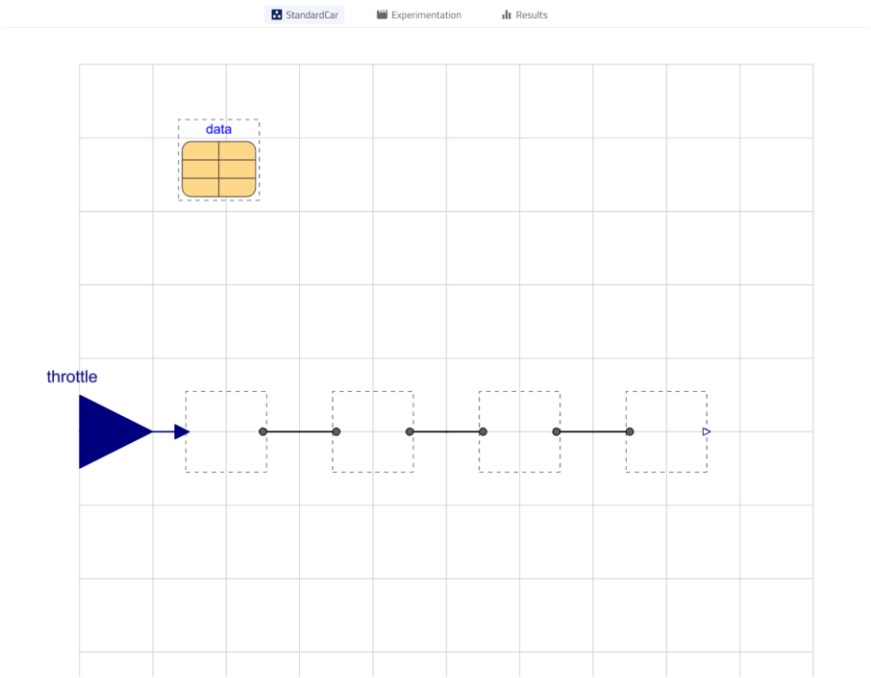
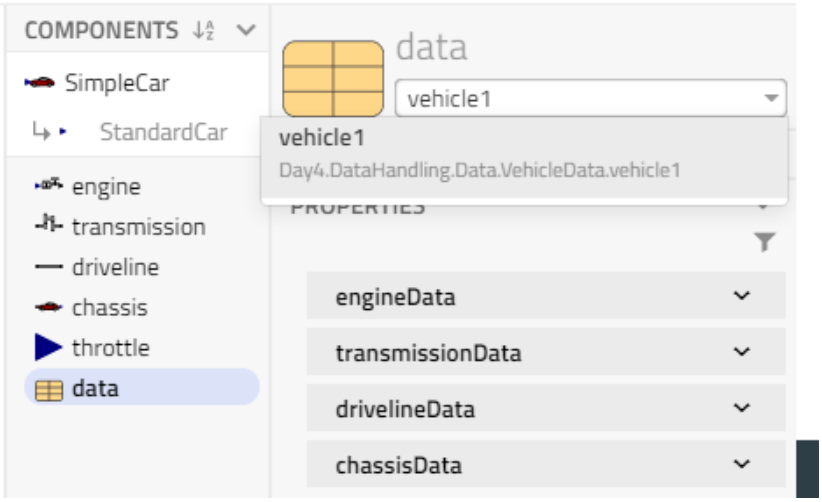


Figure 7 First stage of template StandardCar

Adding the DataRecord to the model

6. Go to the *Configurations.SimpleCar*. The *Configurations.SimpleCar* extends the *StandardCar* template, and each model subsystem has already been configured. Now let's add the data available in the vehicle1 record, by changing the data in the dialog.



7. The SimpleCar model should look like Figure 8.

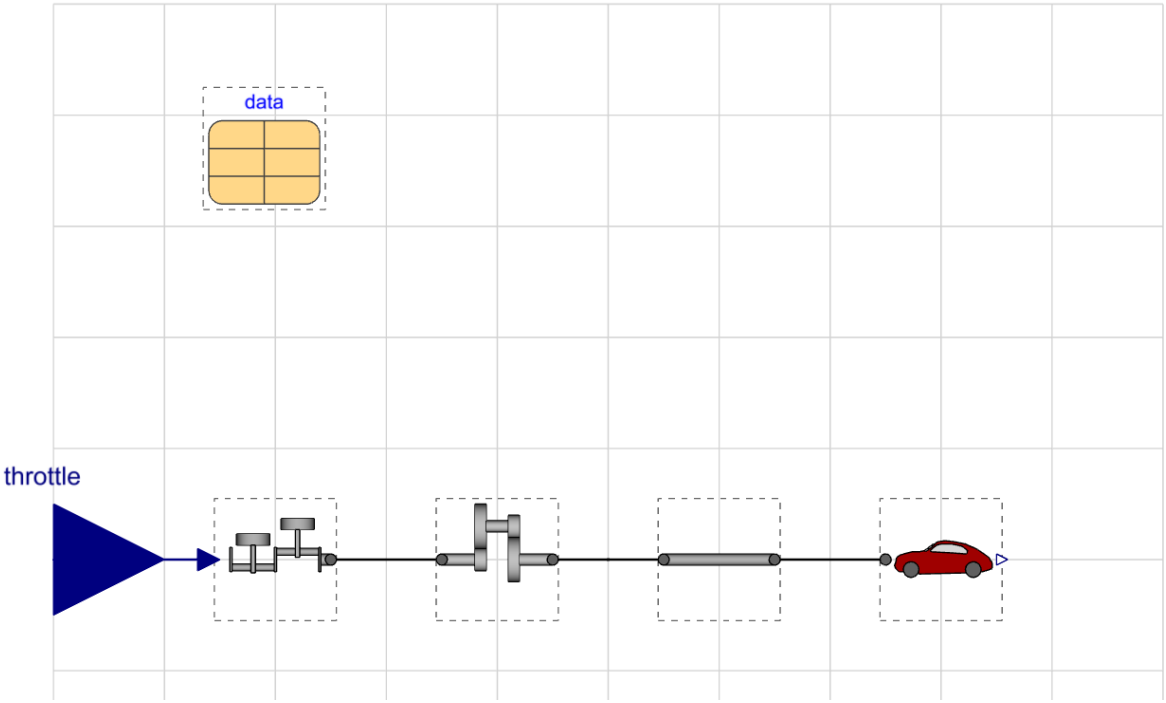


Figure 8 SimpleCar model

8. This record now contains the default values of a simple car. Look at the parameter dialog for the data record. By using the drop-down menu, you can change the *data* records and by opening a specific record you can bring up the parameter dialog of that particular sub record, see the parameter values and edit them as shown in Figure 9.

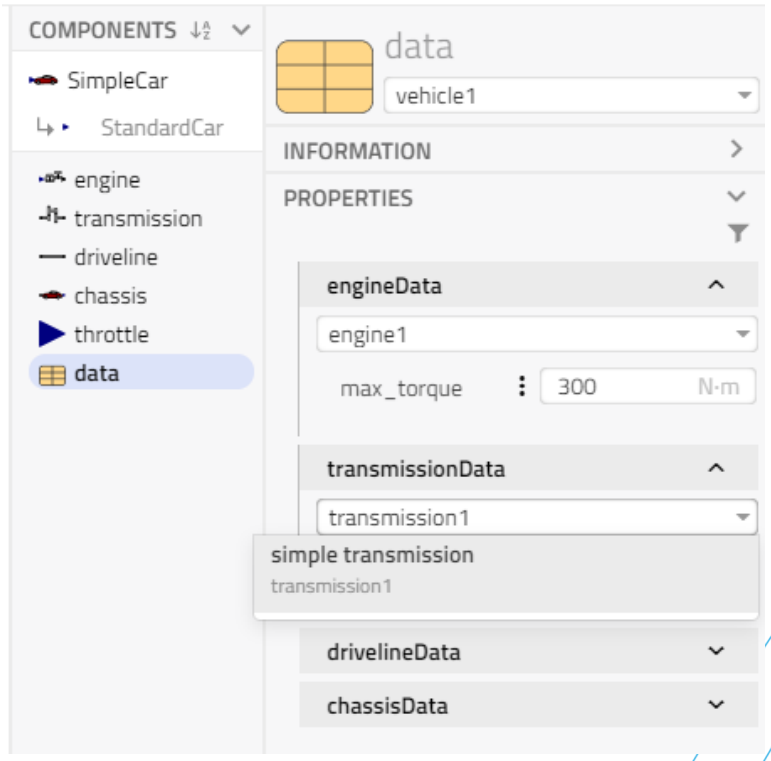


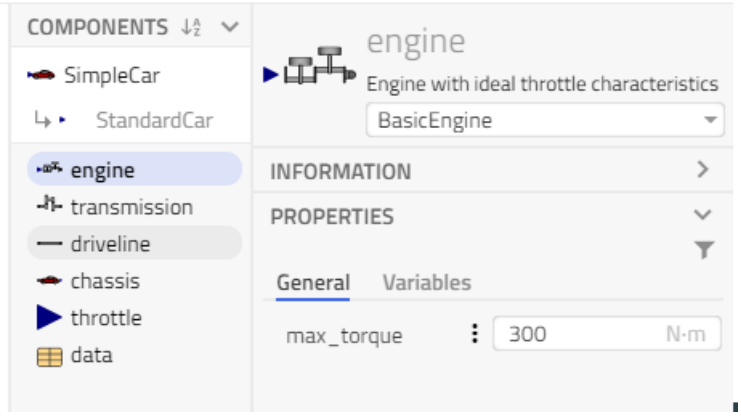
Figure 9 Changing parameter of data record

The parameter data is still only present in the data structure and needs to be propagated into the model structure.

Propagating the data into the model

The car model needs to be updated so it uses the data present in the data records. This will be done by component referencing.

9. Go to the model *Configurations.SimpleCar*. Bring up the parameter dialog for the engine model.



10. Here you see the default value, from the class definition.
11. Associate the **max_torque** parameter to the same parameter in the data record. This can be done by typing directly in the dialog box.

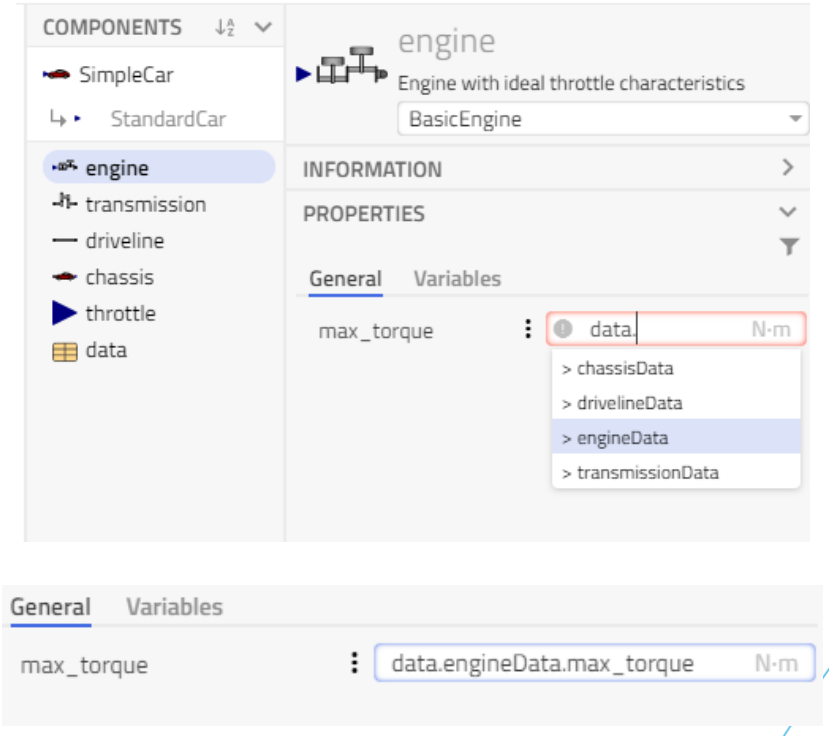


Figure 10 Propagating engine data

12. Do the same thing for the parameters in the *Transmission*, *Driveline* and *Chassis* components. See the Figure 11.

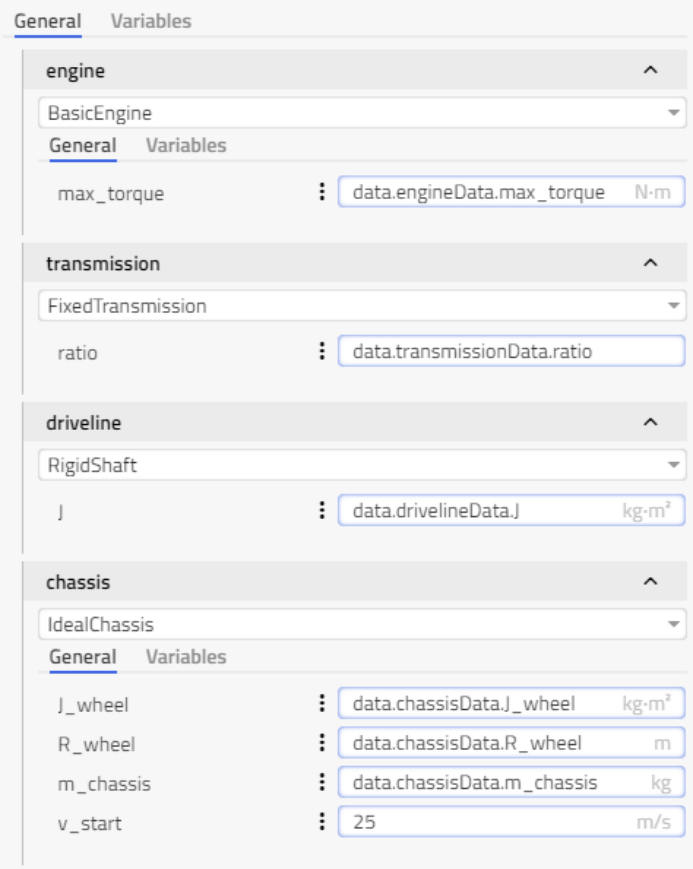


Figure 11 Parameter for other three components

Note that this data referencing only needs to be done once when you create your model. Then you can either duplicate or extend it and reconfigure the data to create model variants.

13. Just to try it out, simulate the model for 10s and test modifying the model by editing the data record and compare the results. There is an experiment set up for the car already, *W2_DataHandling.Experiments.ThrottleTest.*