



# REUSABLE COMPONENTS

Lecture 1.3

*Modelon*

# OVERVIEW

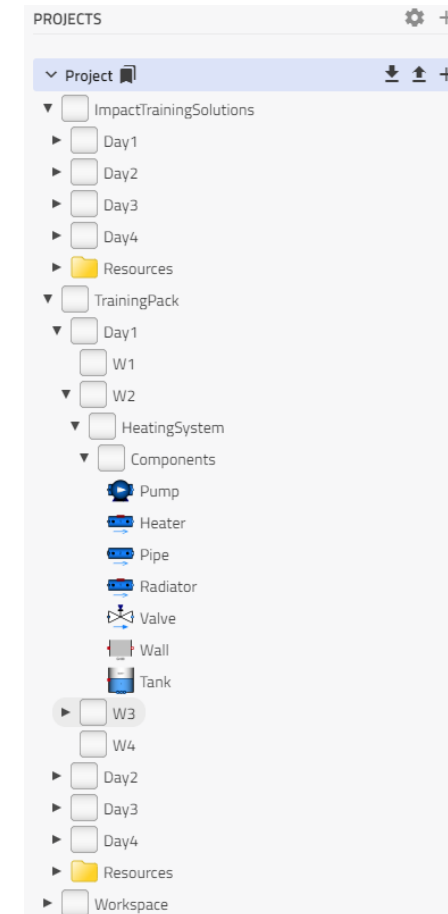
- ✓ Library packages
- ✓ Creating reusable subsystems
- ✓ Connector Interface
- ✓ Parameter Interface
- ✓ Component views
- ✓ Documentation and Icon editor



# LIBRARY PACKAGES

# LIBRARIES

- As you start creating more content its important to organize your work
- Libraries can easily be created and managed in Impact
- Libraries are defined by the modelica class "package"
- A library can contain several hierarchical levels of packages

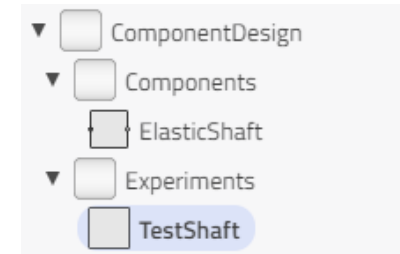
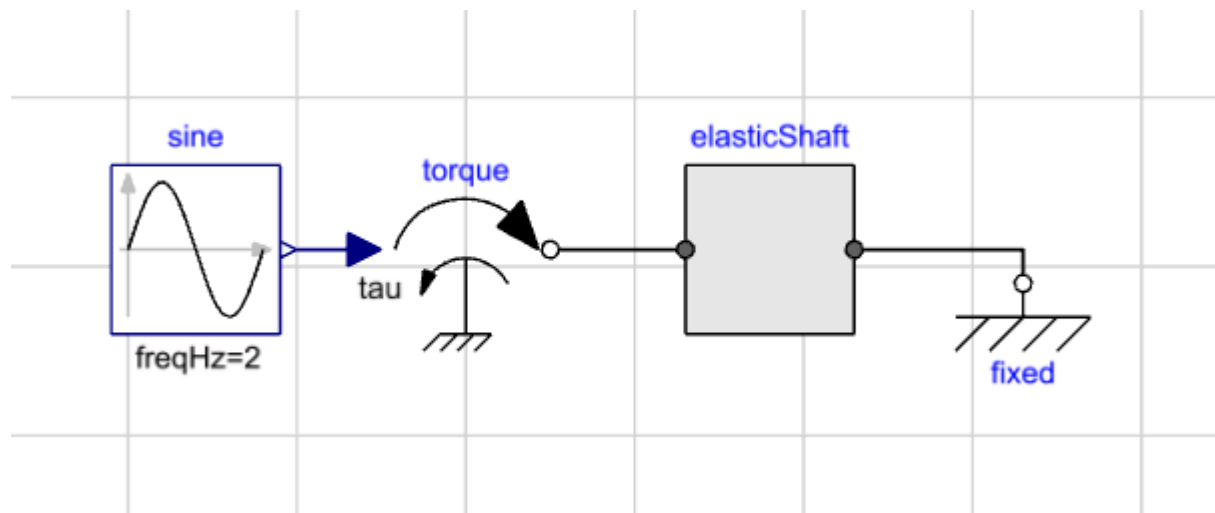




# CREATING REUSABLE COMPONENTS

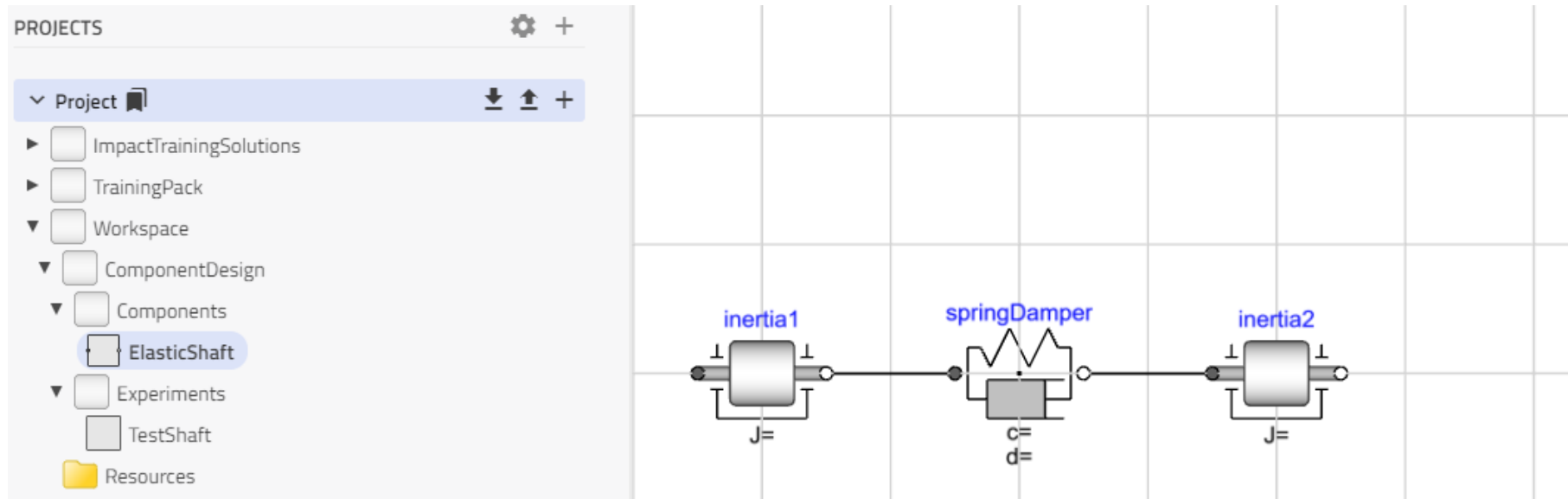
# REUSABLE COMPONENT

- In the following example we want to create an elastic drive shaft.
- Then we want to test this component in a test rig.



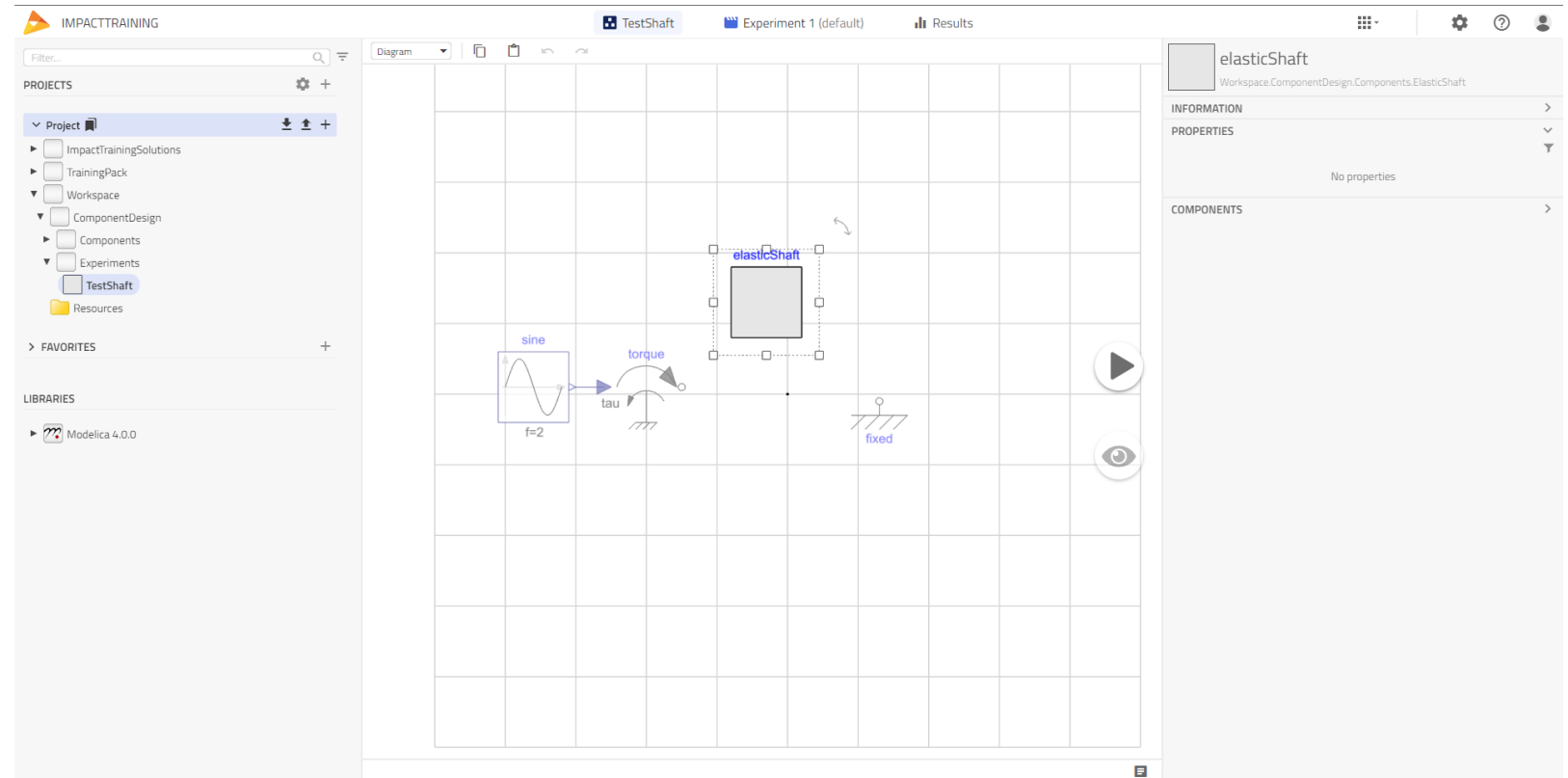
# CREATE SHAFT

- Create a new model -> ElasticShaft
- Drag, drop and connect the needed components



# CREATE SHAFT

- No connectors
- No parameters
- We need to add that!



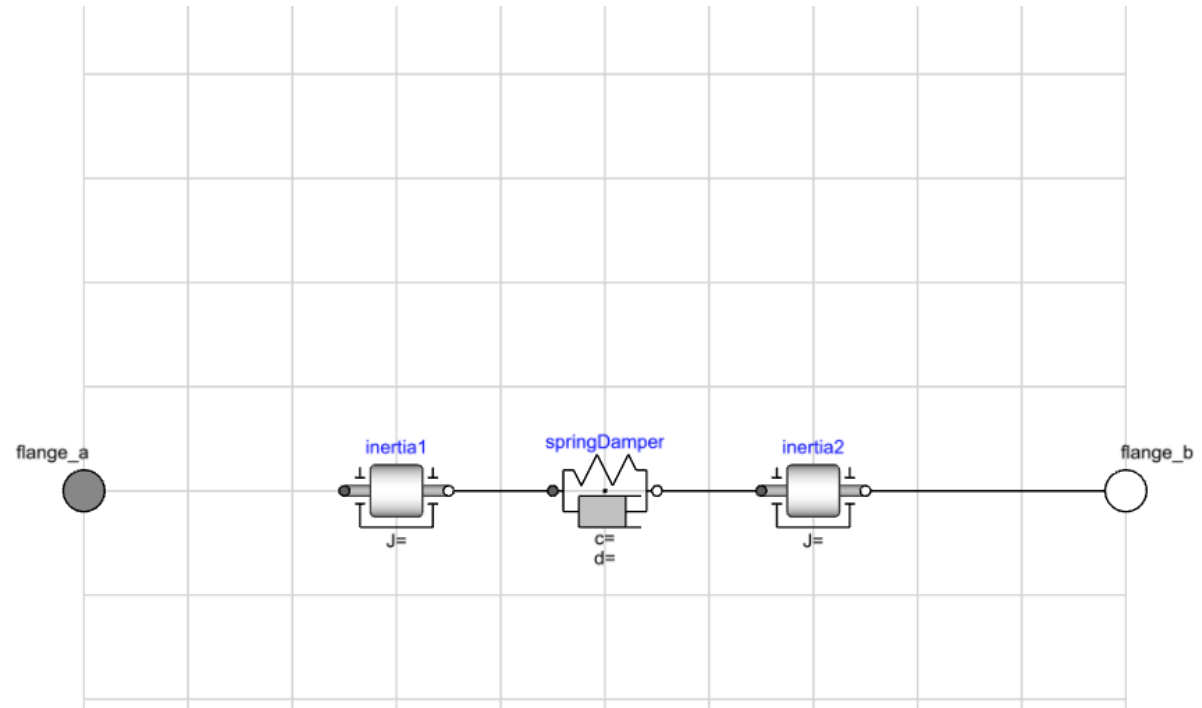
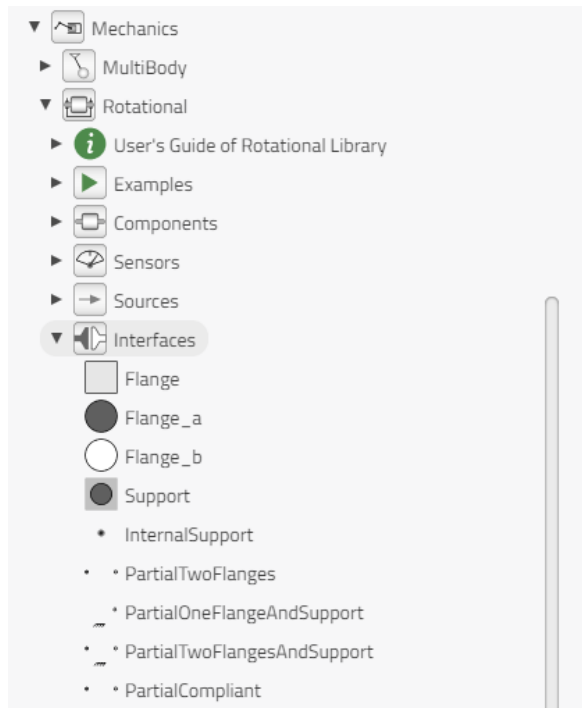




# CONNECTOR INTERFACE

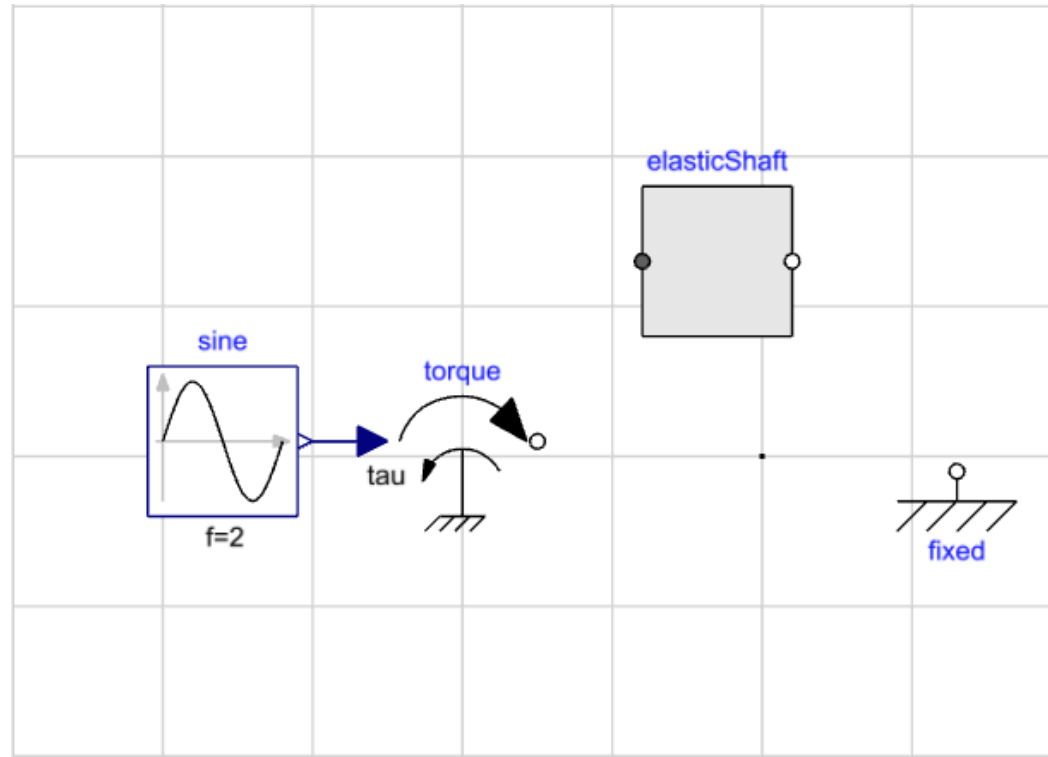
# CONNECTOR INTERFACE

- We find the connectors in the relevant Interface package
- Add and then connect them to the model



# CONNECTOR INTERFACE

- Now we can see that we have the connectors available when using the class





# PARAMETER INTERFACE

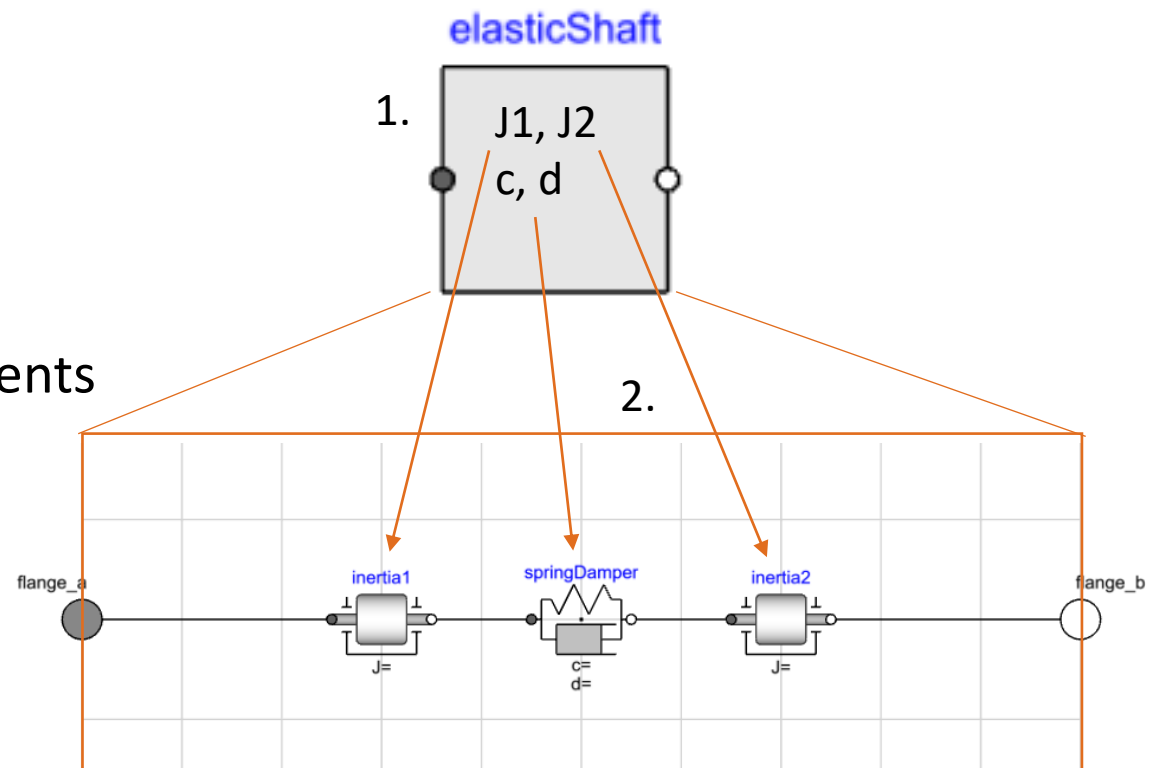
# PARAMETER PROPAGATION

- When we use this model, we would like to be able to set the following properties in ElasticShaft
  - Inertia values,  $J_1$  and  $J_2$
  - Spring and damper values,  $c$  and  $d$ .

This can be done by:

1. Define the parameters in ElasticShaft
2. Use them as modifiers in the sub-components

This is called parameter propagation



# CREATING A NEW PARAMETER

COMONENTS ↓ ↕ ▾  
ElasticShaft  
inertia1  
springDamper  
inertia2  
flange\_a  
flange\_b

ElasticShaft  
...onentDesign.Components.ElasticShaft

INFORMATION >  
PROPERTIES ▾  
+  
Add variable

ADD VARIABLE

Variability: Parameter ▾ Type: Inertia (kg·m<sup>2</sup>) ▾

Name: J1 Expression: 0.1

Description: Inertia 1

Tab: Group:

CANCEL ADD

COMONENTS ↓ ↕ ▾  
ElasticShaft  
inertia1  
springDamper  
inertia2  
flange\_a  
flange\_b

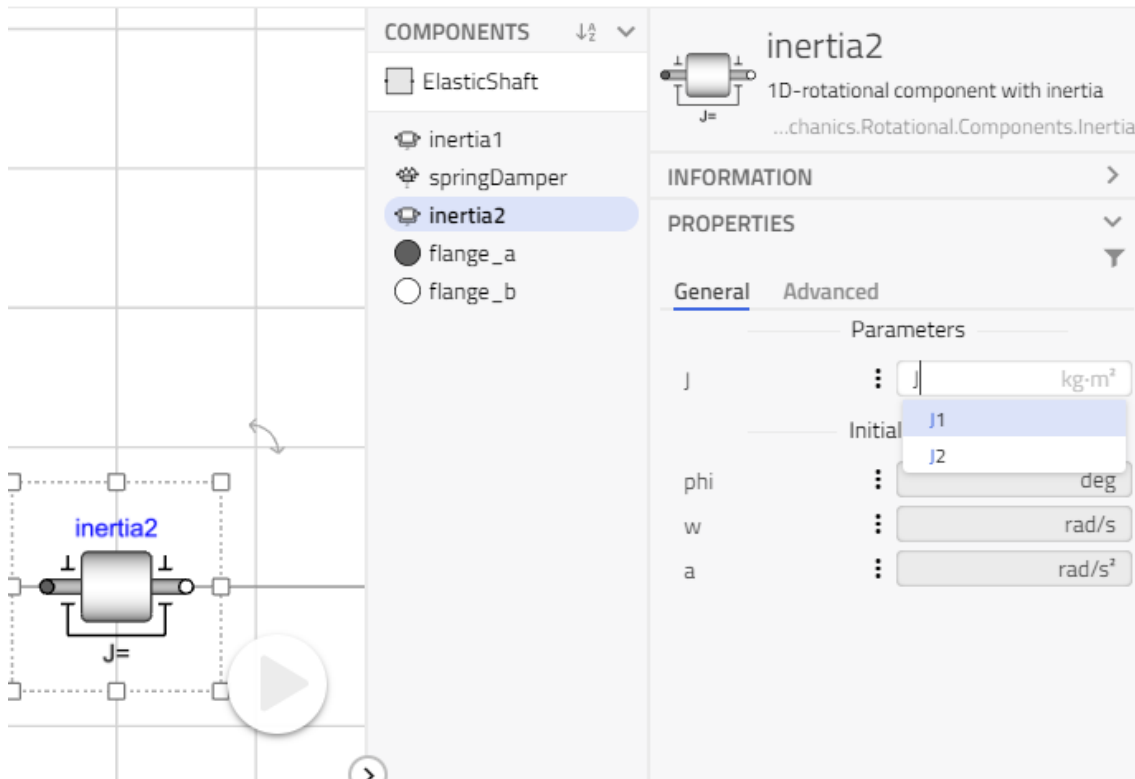
ElasticShaft  
...onentDesign.Components.ElasticShaft

INFORMATION >  
PROPERTIES ▾  
+ ▾

J1	:	0.1	kg·m <sup>2</sup>	×
J2	:	0.3	kg·m <sup>2</sup>	×
c	:	1000	N·m/rad	×
d	:	1	N·m·s/rad	×

# MODIFY THE SUBCOMPONENTS VALUES

In properties tab:



The screenshot shows the Modelon software interface. On the left, a 3D model of a shaft assembly is shown with a play button overlaid. The main panel is divided into two sections: 'COMPONENTS' and 'PROPERTY TAB'. The 'COMPONENTS' list includes 'ElasticShaft', 'inertia1', 'springDamper', 'inertia2' (selected), 'flange\_a', and 'flange\_b'. The 'PROPERTY TAB' for 'inertia2' is open, showing a 'General' tab with a 'Parameters' section. The parameter 'J' is highlighted, and its value is being edited from 'J' to 'J1' in a dropdown menu. The units are 'kg·m²'. Other parameters listed are 'Initial' (with a dropdown for 'J2'), 'phi' (units 'deg'), 'w' (units 'rad/s'), and 'a' (units 'rad/s²').

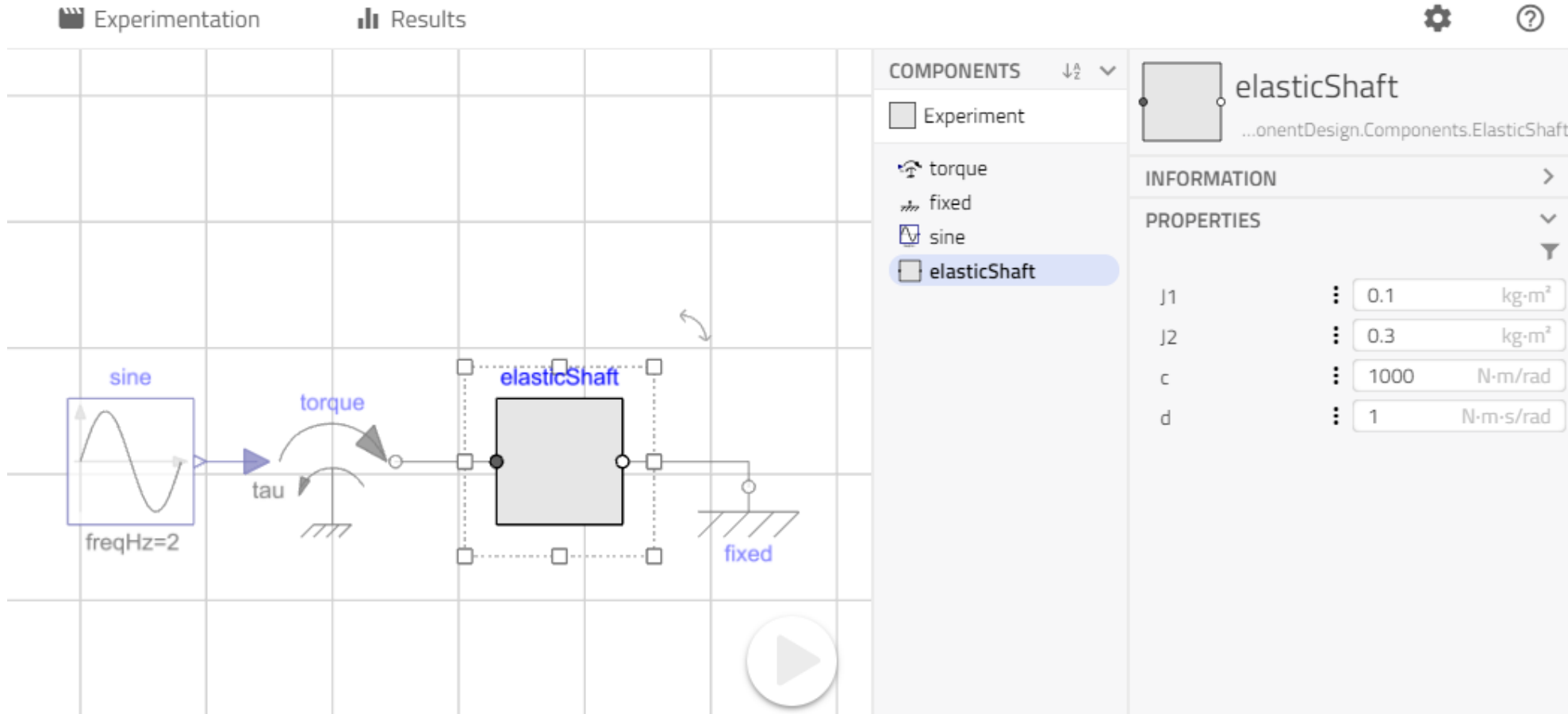
In code editor:

```
model ElasticShaft
  .Modelica.Mechanics.Rotational.Components.Inertia inertia1(J = J1) annotation(...);
  .Modelica.Mechanics.Rotational.Components.SpringDamper springDamper(c=c,d=d) annotation(...);
  .Modelica.Mechanics.Rotational.Components.Inertia inertia2(J = J2) annotation(...);
  .Modelica.Mechanics.Rotational.Interfaces.Flange_a flange_a annotation(...);
  .Modelica.Mechanics.Rotational.Interfaces.Flange_b flange_b annotation(...);
  parameter .Modelica.Units.SI.Inertia J1 = 0.1 "Inertia 1";
  parameter .Modelica.Units.SI.Inertia J2 = 0.3;
  parameter .Modelica.Units.SI.RotationalSpringConstant c = 1000;
  parameter .Modelica.Units.SI.RotationalDampingConstant d = 1;
equation
  connect(inertia1.flange_b, springDamper.flange_a) annotation(...);
  connect(springDamper.flange_b, inertia2.flange_a) annotation(...);
  connect(inertia2.flange_b, flange_b) annotation(...);
  connect(inertia1.flange_a, flange_a) annotation(...);
  annotation(...);
end ElasticShaft;
```

Both ways are equivalent!

# USING THE FINISHED MODEL

Experimentation Results



The screenshot displays a software interface for modeling an elastic shaft. The main workspace shows a schematic diagram with the following components and connections:

- sine**: A sine wave input block with the parameter  $\text{freqHz}=2$ .
- torque**: A torque input block labeled  $\tau$ .
- elasticShaft**: The central component being modeled.
- fixed**: A fixed boundary condition at the end of the shaft.

The right-hand side of the interface shows the **elasticShaft** component's configuration:

**COMPONENTS**

- Experiment
- torque
- fixed
- sine
- elasticShaft** (selected)

**elasticShaft**  
...onentDesign.Components.ElasticShaft

**INFORMATION** >

**PROPERTIES** v

J1	:	0.1	kg·m <sup>2</sup>
J2	:	0.3	kg·m <sup>2</sup>
c	:	1000	N·m/rad
d	:	1	N·m·s/rad

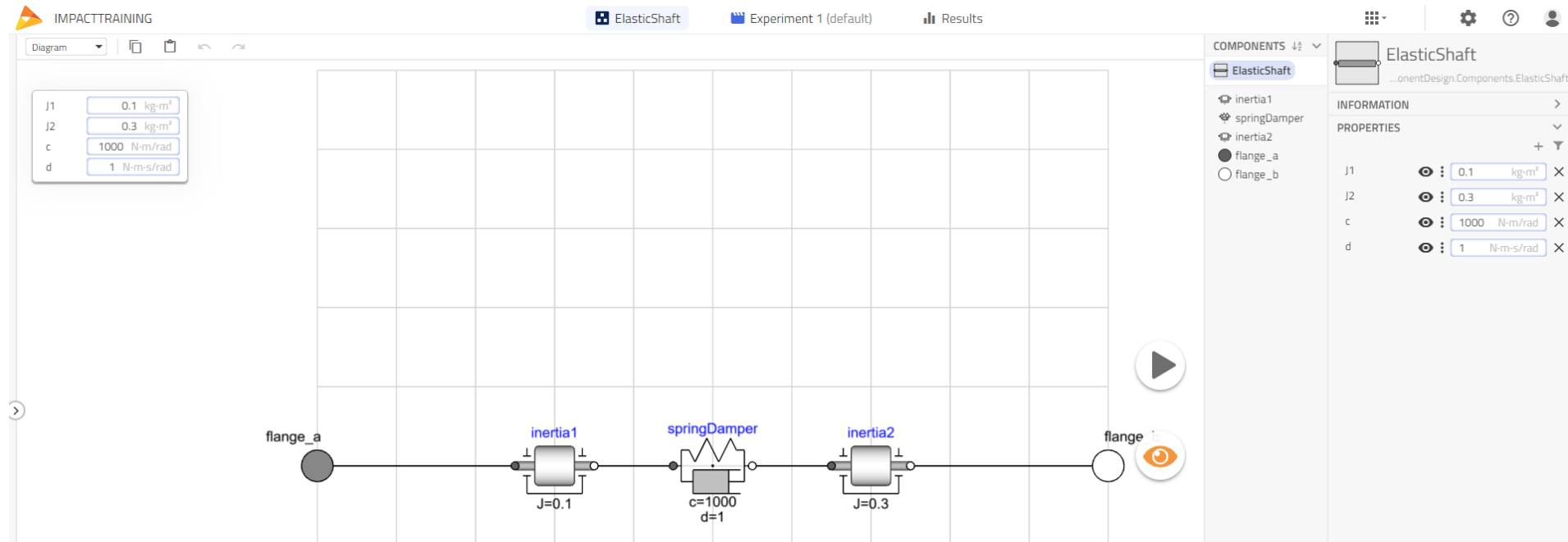




# COMPONENT VIEWS

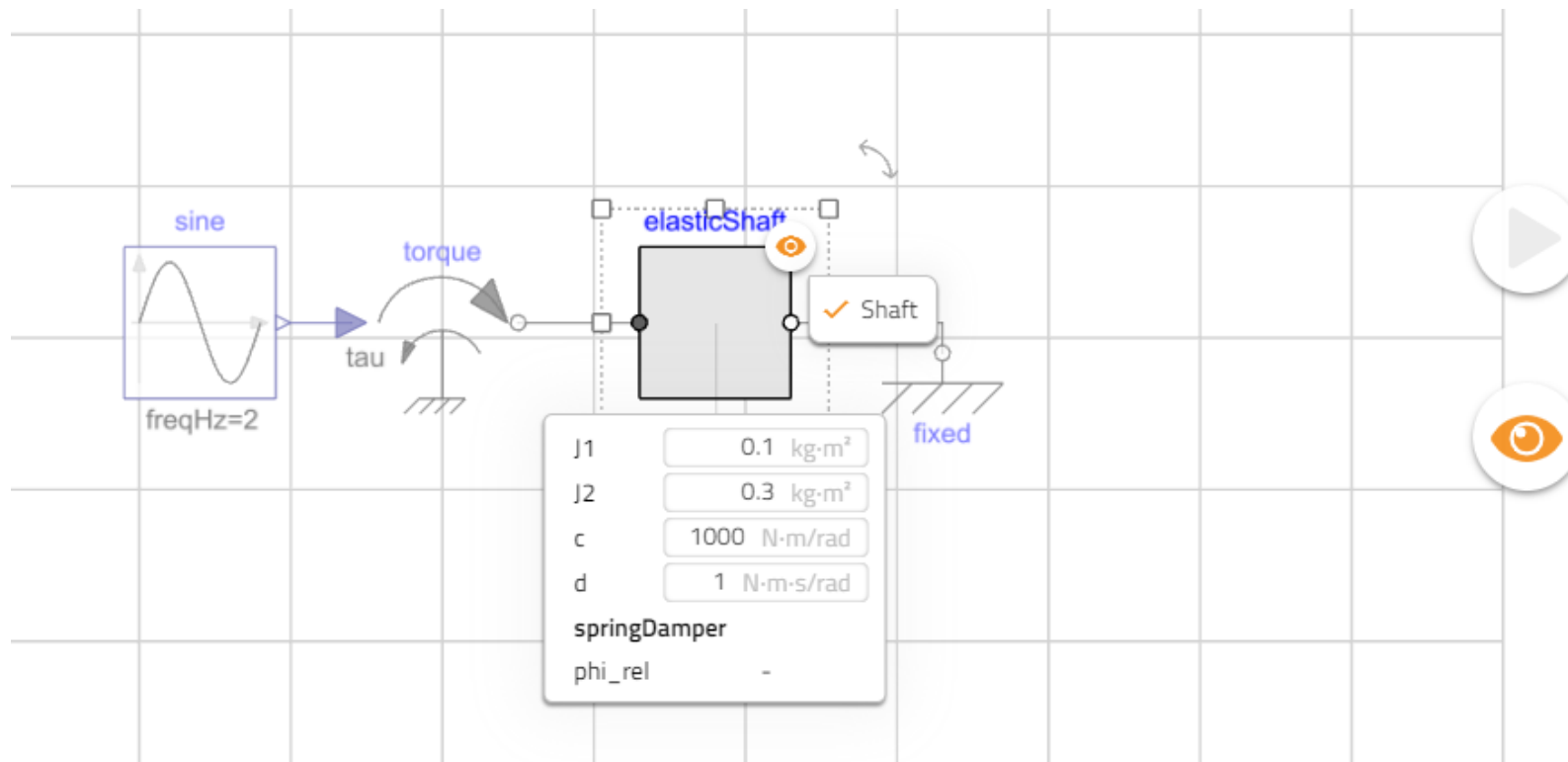
# COMPONENT VIEW

- A view defined inside a component model, can be reused in a system model



# COMPONENT VIEW

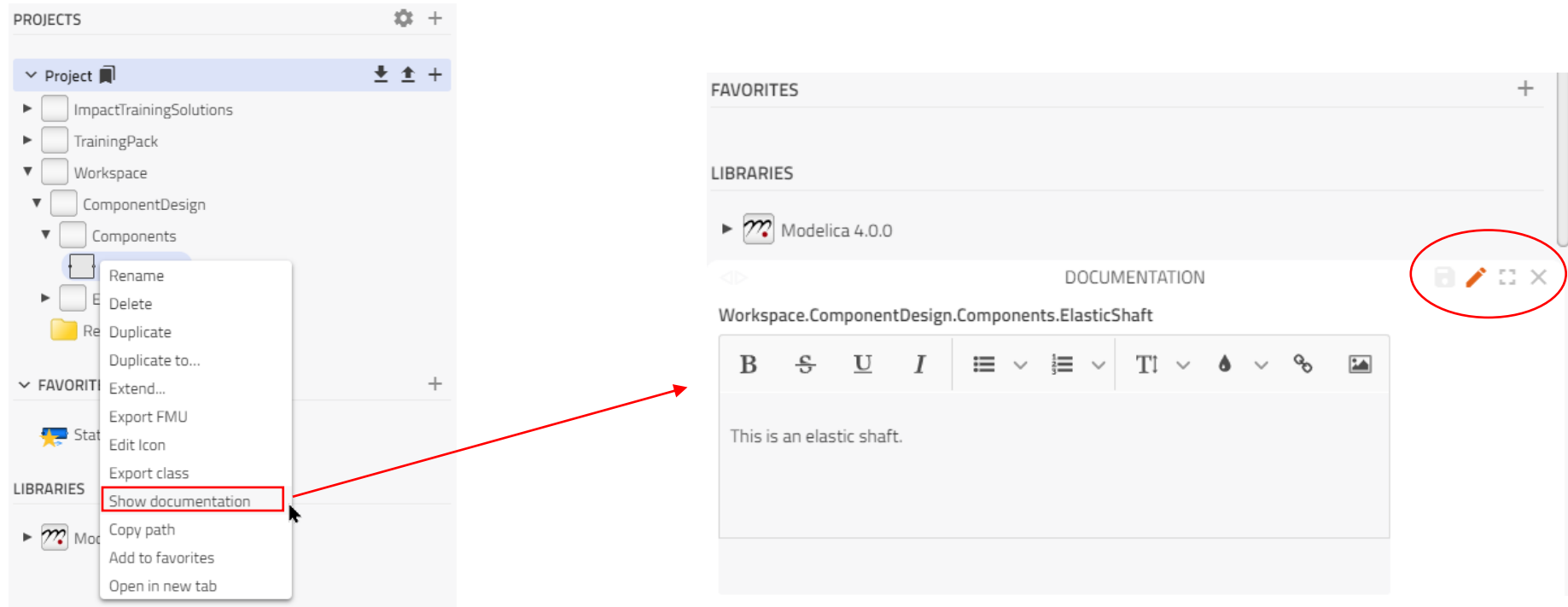
- The view can be activated by clicking  on the specific instance.





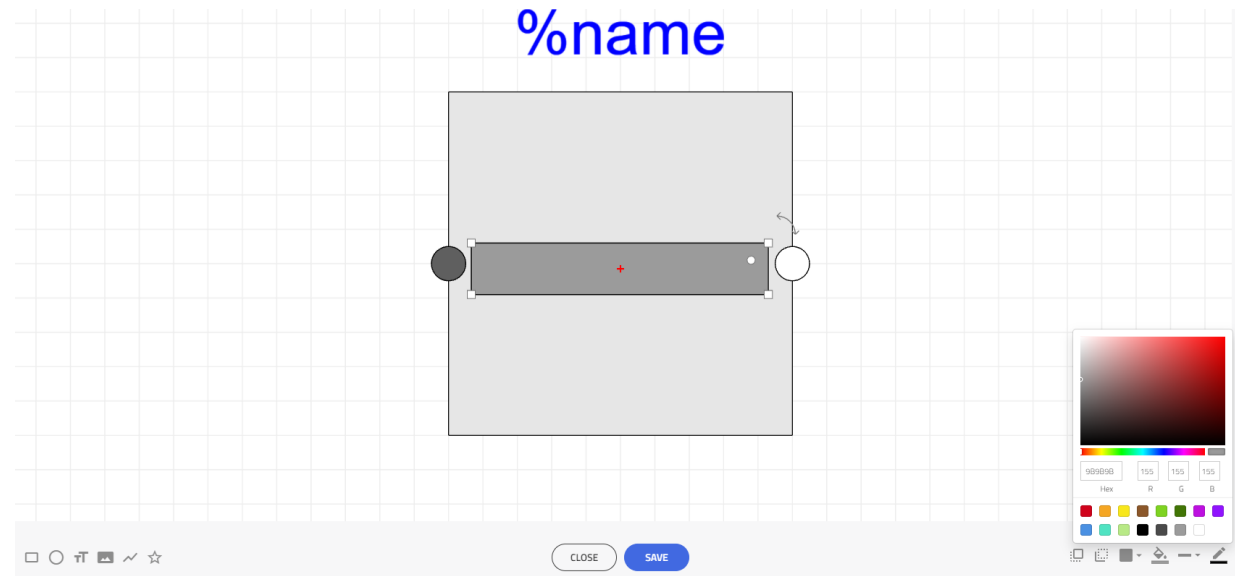
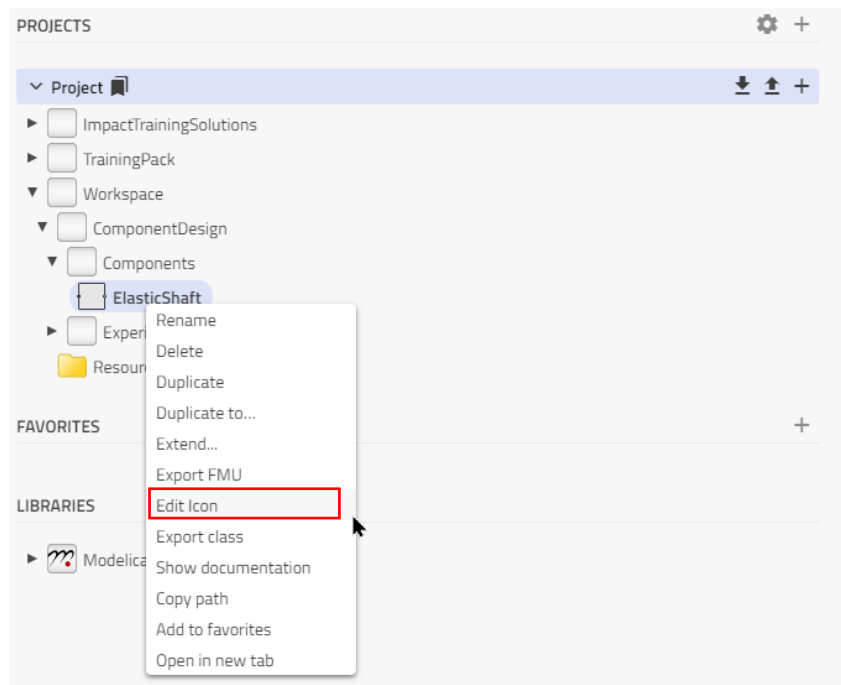
# ICON EDITOR AND DOCUMENTATION EDITOR

# DOCUMENTATION EDITOR

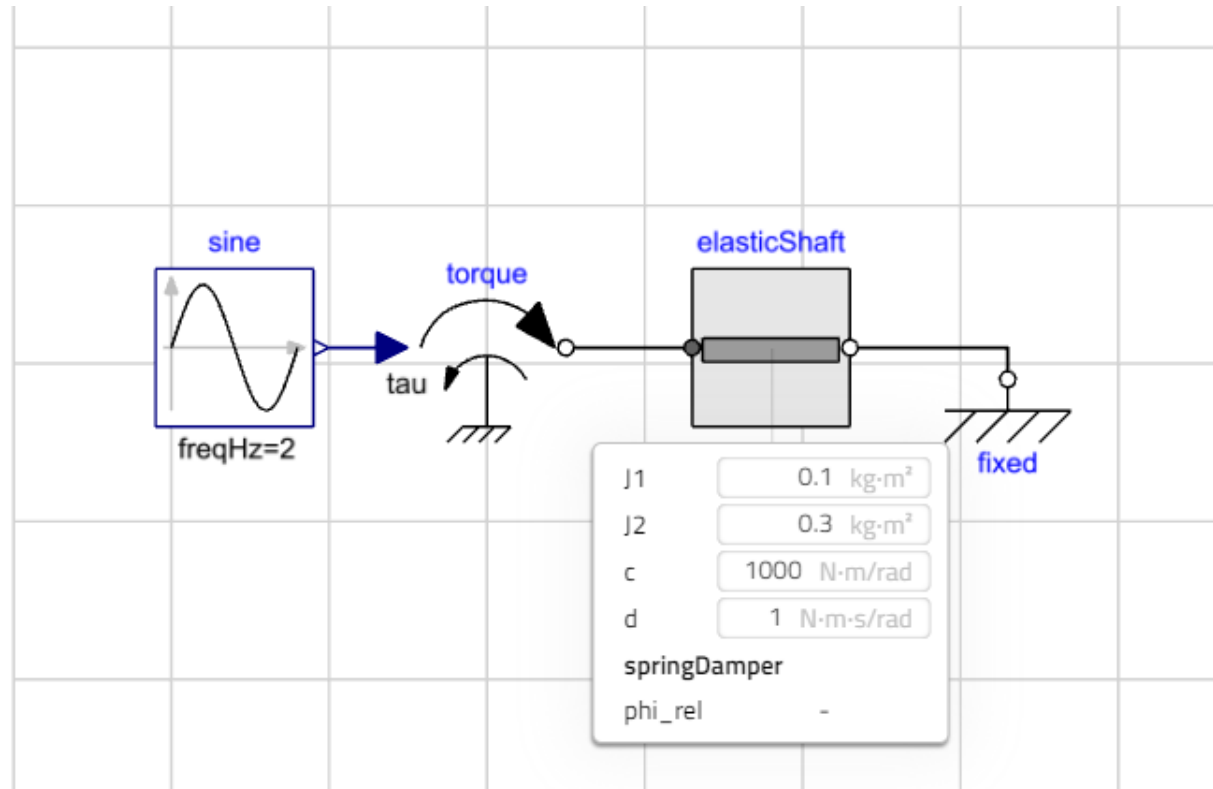


# ICON EDITOR

- Draw an icon using simple primitives, or import an image.



# COMPONENT READY



# WORKSHOP 1.3

In this workshop you will:

- Create a component interface
  - Add connectors
  - Add and propagate parameters
- Test the component in a rig
- Add an Icon and Documentation