Powering Innovation That Drives Human Advancement

Automation cheat slides

Selection Manager

- Clear selection:

```
ExtAPI.SelectionManager.ClearSelection()
```

- Get IDs of entities currently selected in the UI:

```
ExtAPI.SelectionManager.CurrentSelection
```

- Create a new selection and use it:

```
# create a new empty selection
tempSel = ExtAPI.SelectionManager.CreateSelectionInfo(SelectionTypeEnum.GeometryEntities)
# provide list of Ids of entities to select
tempSel.Ids = [4]
# create new meshing method
newMethod = ExtAPI.DataModel.Project.Model.Mesh.AddAutomaticMethod()
# assign location
newMethod.Location=tempSel
```

Selecting objects by name

❑Using GetObjectsByName:

This method will return a list of all the entities with the named passed in argument:

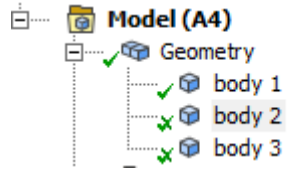
```
ExtAPI.DataModel.GetObjectsByName("Test")# returns a list of all entitites named "Test"
```

❑Selecting a named selection by its name and scope this selection to a force:

```
analysis = ExtAPI.DataModel.Project.Model.Analyses[0] # reference analysis

newForce=analysis.AddForce() # add force

myNS= DataModel.GetObjectsByName("TestSelection")[0] # select first item in the tree with
  name "TestSelection"

NewForce.Location = myNS # scope named selection
```

Geometry

- Loop through children inside geometry branch:

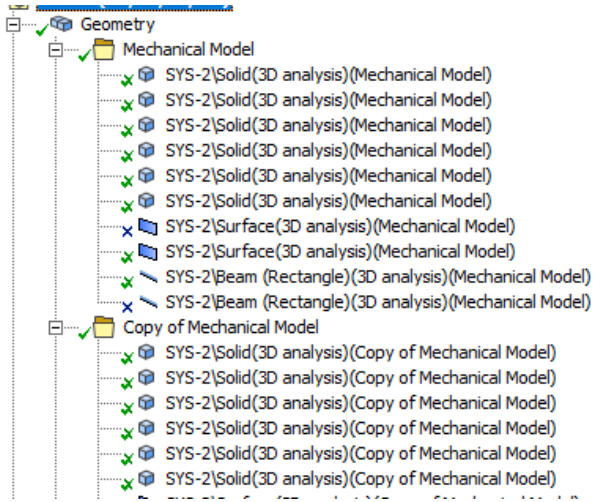



```
for part in ExtAPI.DataModel.Project.Model.Geometry.Children:
        for body in part.Children:
                print body.Name
```

! If the model uses model assembly, this structure is lost as parts and bodies are grouped in folders. It is then better to select bodies through their type (as below)

- Get list of all bodies:

```
geometry = ExtAPI.DataModel.Project.Model.Geometry
listBodies = geometry.GetChildren(DataModelObjectCategory.Body,True)
```



Geo Data

- Loop through assemblies, parts and bodies of GeoData:

```
for Assembly in ExtAPI.DataModel.GeoData.Assemblies:
        for Part in Assembly.Parts:
                for Body in Part.Bodies:
                        for Surface in Body.Faces:
                                print(Surface.Area)
```

Materials

- Material properties defined in Engineering Data can be accessed thanks to the material module:

```
# import materials module
import materials
# get to first material in the Mechanical tree
mat = ExtAPI.DataModel.Project.Model.Materials.Children[0]
print(mat.Name)
# get engineering data material properties for this material
matED = mat.GetEngineeringDataMaterial()
# get and print list of material properties
listMatProp = materials.GetListMaterialProperties(matED)
print(listMatProp)
# get and print Elasticity properties (if it exists)
elasticity = materials.GetMaterialPropertyByName(matED,"Elasticity")
print(elasticity)
```

Coordinate systems

- Create a coordinate system:

```
# create coordinate system:

CS = ExtAPI.DataModel.Project.Model.CoordinateSystems.AddCoordinateSystem()

# modify CS name

CS.Name="New CS"

# change CS type

CS.CoordinateSystemType=CoordinateSystemTypeEnum.Cylindrical
```

Connections

- Define a contact region between two faces:

```
# create contact:
Connections=ExtAPI.DataModel.Project.Model.Connections
newContact=Connections.AddContactRegion()
# modify contact type and define friction coefficient:
newContact.ContactType=ContactType.Frictional
newContact.FrictionCoefficient=0.2
ExtAPI.DataModel.Tree.Refresh() # refresh tree to see modifications
# select contact and target faces:
sourceSel=ExtAPI.SelectionManager.CreateSelectionInfo(SelectionTypeEnum.GeometryEntities) # create empty
selection
sourceSel.Ids=[137] # define Ids of faces to be included
newContact.SourceLocation=sourceSel # apply selection to contact side
targetSel=ExtAPI.SelectionManager.CreateSelectionInfo(SelectionTypeEnum.GeometryEntities)
targetSel.Ids=[164] # define Ids of faces to be included
newContact.TargetLocation= targetSel # apply selection to target side
```

Mesher

- Define a mesh method on a body:

```
# reference mesher

mesher = ExtAPI.DataModel.Project.Model.Mesh

# create body selection
tempSel = ExtAPI.SelectionManager.CreateSelectionInfo(SelectionTypeEnum.GeometryEntities)

tempSel.Ids = [4]

# insert and define new meshing method
newMethod = mesher.AddAutomaticMethod()
newMethod.Location=tempSel
newMethod.Method=MethodType.HexDominant
```

Mesh Data

- Access information on an element:

```
# reference mesh data

meshData = ExtAPI.DataModel.Project.Model.Analyses[0].MeshData

# select one element and display information
elem=meshData.Elements[0]

elem.Centroid # position of element

elem.Type # type of element
```

Named Selection - Worksheet

- Create a worksheet named selection:

```
model = ExtAPI.DataModel.Project.Model # reference model
newNS = model.AddNamedSelection() # add named selection
newNS.Name = "New named selection" # change name
newNS.ScopingMethod = GeometryDefineByType.Worksheet # set definition to worksheet
objId = newNS.ObjectId # reference ID of named selection object for future use

# add a new selection criteria (new line in worksheet)
DataModel.GetObjectById(objId).GenerationCriteria.Add(None)

# make first row of worksheet active to modify it

DataModel.GetObjectById(objId).GenerationCriteria[0].Active = True

# define selection type
DataModel.GetObjectById(objId).GenerationCriteria[0].EntityType = SelectionType.MeshNode

# define selection criterion

DataModel.GetObjectById(objId).GenerationCriteria[0].Criterion = SelectionCriterionType.LocationX

# define operator

DataModel.GetObjectById(objId).GenerationCriteria[0].Operator = SelectionOperatorType.LessThanOrEqual

# set value

DataModel.GetObjectById(objId).GenerationCriteria[0].Value = Quantity("1 [mm]")
```

Analysis settings

- Access information on analysis settings:

```
# analysis settings is always Children[0] of the analysis
settings = ExtAPI.DataModel.Project.Model.Analyses[0].Children[0]
```

- Define autotime stepping:

```
settings.Activate() # activate Analysis Settings Object
settings.CurrentStepNumber=1 # select active step
settings.AutomaticTimeStepping=AutomaticTimeStepping.On # activate algorithm
settings.DefineBy=TimeStepDefineByType.Substeps # define by substeps
settings.InitialSubsteps=10 # define values
settings.MinimumSubsteps=5
settings.MaximumSubsteps=25
```

Boundary conditions

❑ Define a variable pressure boundary condition:

```
analysis=ExtAPI.DataModel.Project.Model.Analyses[0] # refer to analysis
newPress=analysis.AddPressure() # add pressure
# define entry values as tabular data
newPress.Magnitude.Inputs[0].DiscreteValues=[Quantity('0.5[s]'),Quantity('1[s]')]
# define (output) pressure values as tabular data
newPress.Magnitude.Output.DiscreteValues=[Quantity('10[MPa]'),Quantity('20[MPa]')]
```

❑ Define loads by components (instead of vector):

```
analysis=ExtAPI.DataModel.Project.Model.Analyses[0] # refer to analysis
newForce=analysis.AddForce() # add force
newForce.DefineBy = LoadDefineBy.Components # define by components
newForce.ZComponent.Output.DiscreteValues = [Quantity("1 [N]")] # define Z value
newForce.YComponent.Output.DiscreteValues = [Quantity("1 [N]")] # define Y value
```

Solution

- Create a user-defined result :

```
# store link to solution
solu=ExtAPI.DataModel.Project.Model.Analyses[0].Solution
# add UDR
newUDR = solu.AddUserDefinedResult()
# define expression
newUDR.Expression="UX"
# change display time
newUDR.DisplayTime = Quantity('1 [sec]')
# evaluate
newUDR.EvaluateAllResults()
```

Result Data

- Retrieve all components of stress tensor on one element:

```
model=ExtAPI.DataModel.Project.Model # refer to model
reader = model.Analyses[0].GetResultsData() # get results data of first analysis in the tree
StressResults = reader.GetResult('S') # obtain stress results
StressElement1 = StressResults.GetElementValues(1) # retrieve results on element n°1
```

- Retrieve stress values in one direction:

```
model=ExtAPI.DataModel.Project.Model # refer to model
reader = model.Analyses[0].GetResultsData() # get results data of first analysis in the tree
StressResults = reader.GetResult('S') # obtain stress results
StressResults.SelectComponents(["X"]) # select direction
StressXElement1=StressResults.GetElementValues(1) # retrieve results on element n°1
```


Ansys