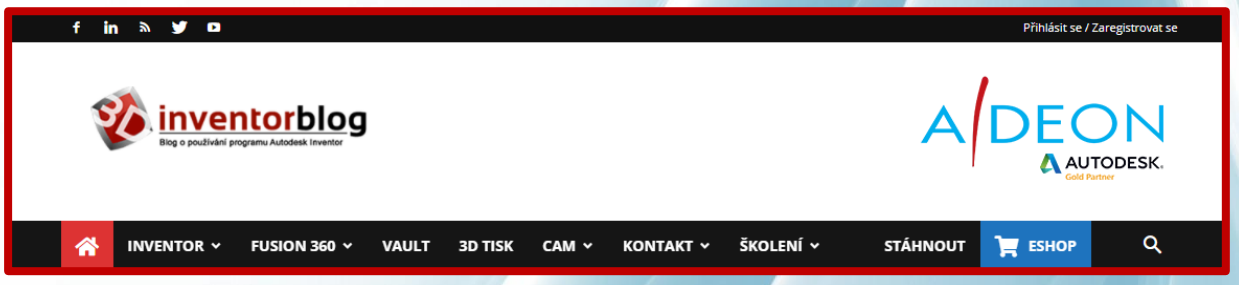
Autodesk iLogic

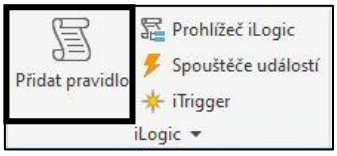
Základní školení



Petr Meduna
petr.meduna@adeon.cz

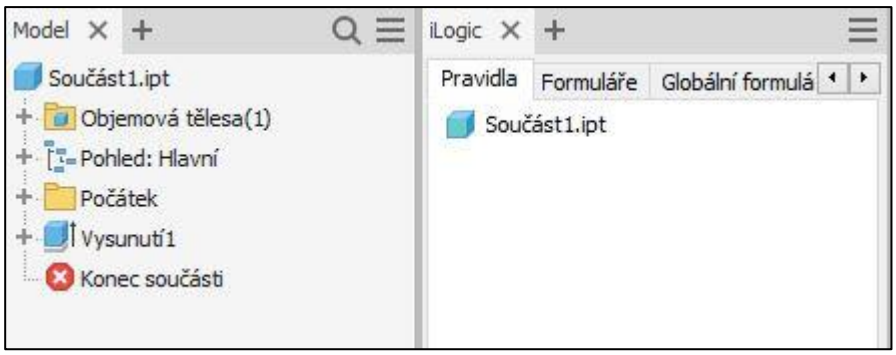
OBSAH

1 Základní práce s iLogicem



1.1 Uživatelské rozhraní

Příkazy iLogicu se vždy nalézají na kartě Správa > iLogic. Aktivací *Prohlížeč iLogic* spustíme nový panel.



Nové pravidlo přidáme klepnutím na *Přidat pravidlo*, nebo pravým klikem do prostoru *prohlížeče iLogic*, který jsme si nyní aktivovali.

Nové pravidlo je třeba pojmenovat.

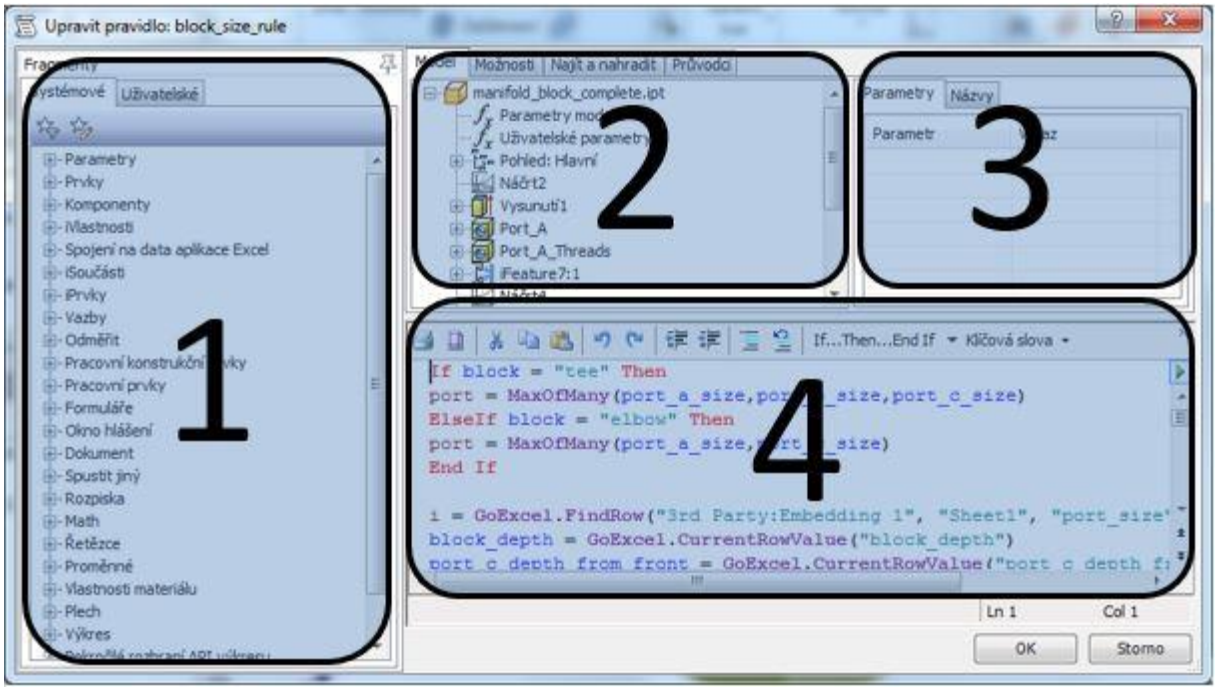
1.2 Dialogové okno pro vytváření pravidel

1) **Fragmenty** – výběr nejdůležitějších a nejpoužívanějších příkazů iLogic z prostředí API Inventoru

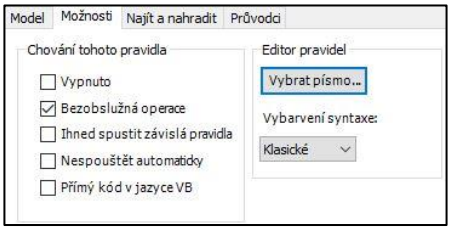
2) **Data modelu** – standardní strom prohlížeče modelu, je zde ke zpřístupnění parametrů, názvů souborů a vazeb*

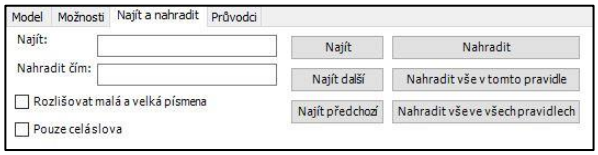
3) **Parametry** – ukazuje jednotlivé parametry, kterými je daná součást definována

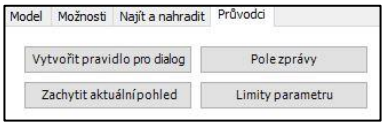
4) **Pracovní část** – sem se zapisuje samotný program, příkazy se barevně odlišují, takže jednoduše lze rozpoznat poznámky, klíčová slova, zobrazený text či samotné příkazy; okno zároveň obsahuje kontrolu syntaxe (zobrazená jako ve Wordu červenou vlnovkou) a kontrolu správnosti programu, kde, pokud je v programu chyba znemožňující jeho spuštění, po stisknutí tlačítka *uložit* vyskočí hláška s popisem chyby a v místě tabulky *parametry* se zobrazí, na kterém řádku se chyba nachází; program disponuje ještě systémovou kontrolou, která případně po spuštění pravidla nahlásí chybu faktickou (špatný argument, chybně napsaná podmínka, nedefinovaná proměnná)



*Vedle dat modelu máme i karty „Možnosti“, „Najít a nahradit“ a „Průvodci“:

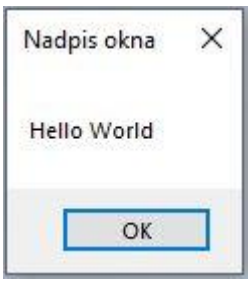






- **Možnosti** – zde lze nastavit vlastnosti pravidla, zda se má ihned spouštět, zda má být vypnuto či zda má vyhodnotit pravidlo jako kód VB, případně i font a zbarvení písma
- **Najít a nahradit** – funkce pro rychlou editaci pravidla, potřebujeme-li nahradit opakující se část kódu (např. proměnnou či parametr) něčím jiným, zedituje zadanou část náhradní hodnotou v celém pravidla nebo ve všech pravidlech
- **Průvodci** – zajišťují reference na soubory DLL s vytvořenými pravidly, vytvářejí okna hlášení, podmínky pro parametry a zachycení aktuálního pohledu

Příklad 1 – Základ vytváření pravidel



Vytvoříme si jednouživatelský projekt a v něm uložíme díl s názvem S01.ipt. Spustíme prohlížeč iLogic a vytvoříme pravidlo s názvem *Hello World*. Do pracovní části opíšeme následující text:

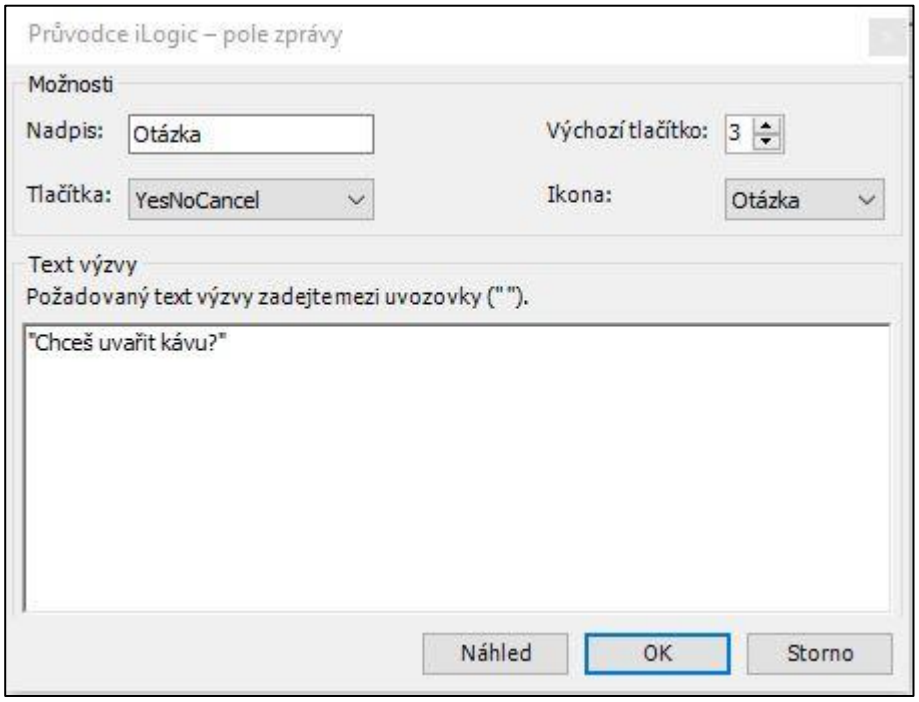
```
MessageBox.Show("Hello World", "Nadpis okna")
```

Nyní potvrďte tlačítkem OK. Pravidlo se nám nyní uložilo a vyskočila nám hláška o provedeném úkonu. Okno editace se nám zavřelo a výsledné pravidlo nyní můžeme najít v panelu iLogic. Pokud bychom chtěli pravidlo spustit znovu nebo upravit, tak na něj musíme kliknout pravým tlačítkem myši a danou úlohu zvolit.

Příklad 2 – Vyskakovací okno přes Průvodce

V Příkladu 1 jsme si ukázali tvorbu jednoduchého pravidla, které po spuštění zobrazí vyskakovací okno. V tomto příkladu si ukážeme, jak takové okno vytvořit přes *Pole zprávy* na kartě *Průvodci*.

Po kliknutí na ikonu se nám zobrazí jednoduchý editační formulář, který si vyplníme dle zadání.



Po kliknutí na tlačítko OK se nám do pracovního prostoru zapíše následující text:

```
i = MessageBox.Show("Chceš uvařit kávu?", "Otázka",
MessageBoxButtons.YesNoCancel, MessageBoxIcon.Question,
MessageBoxDefaultButton.Button3)
```

`MessageBoxButtons.YesNoCancel` je příkaz definující tlačítka pro dialogové okno.
`MessageBoxIcon.Question` definuje typ okna, respektive typ zobrazené ikony u textu.
`MessageBoxDefaultButton.Button3` definuje výchozí tlačítko pro potvrzení dialogového okna.

Lze si všimnout, že celé okno je definováno pro proměnnou i, tedy se předpokládá, že s oknem budeme dále pracovat, například ho zahrneme do nějaké podmínky. V tomto případě lze proměnnou smazat.

1.3 Rozlišení textu v pravidlech

```
Promenna        – lokální parametr pravidla
d0              – existující parametr (uživatelský či modelový)
12,531          – číslo
If              – řídící struktury (podmínky)
'text           – jednořádková poznámka
"Otázka"        – textový řetězec
'''Poznámka na  - víceřádková poznámka
''' několik řádků
Parameter       - systémem rozpoznaná syntaxe
```

2 Proměnné

Proměnná je jakýkoliv námi vytvořený název v programu, který nese informaci a ze kterého buď tuto informaci čteme, nebo ji do něj ukládáme. Existují tři typy proměnných:

- Parametry
- iVlastnosti
- lokální proměnné

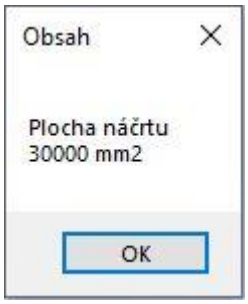
2.1 Parametry

Parametry, se kterými pracujeme v iLogicu, jsou tytéž, které definujeme při vytváření modelu a které se nacházejí v tabulce parametrů. Jedná se v podstatě o souhrn velikostí daných prvků v modelu (vysunutí, rotace, tažení, zkosení,…). Tedy tam, kde zapisujeme nějaké číslo, se vždy vytvoří parametr.
Rozpoznané parametry zapsané do pracovní části dialogového okna jsou označeny modrou barvou, pakliže tomu tak není, daný parametr neexistuje a výraz program bere jako lokální proměnnou.

Příklad 3 – Ukázka práce s parametry

Vytvoříme obdélníkový náčrt součásti, kde si určíme rozměry tvaru jako sirka = 100 a delka = 300. Vytvoříme si nové pravidlo s názvem „obsah“ a zapíšeme následující řádky:

```
sirka = 100 mm
delka = 300 mm
z = (sirka * delka)
MessageBox.Show("Plocha náčrtu" & vbLf & z & " mm2", "Obsah")
```



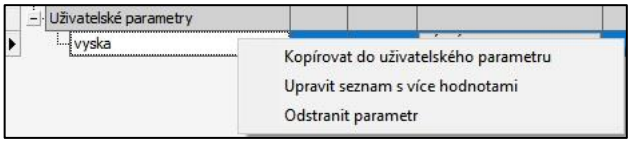
Pravidlo nám nyní ve vyskakovacím okně vypíše výsledek provedené operace, v našem případě součin dvou hodnot. Obdobně lze takto provádět s parametry různé matematické operace.

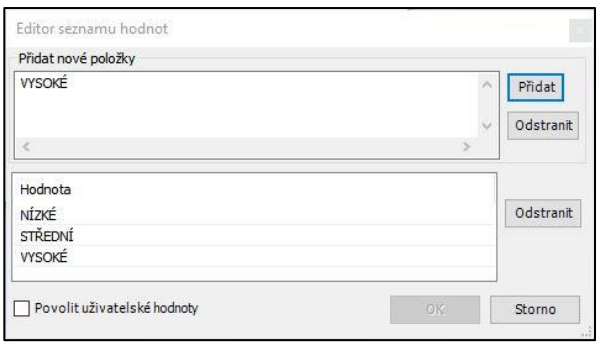
Multiparametr

S příchodem iLogicu do Inventoru vznikly i takzvané multiparametry, tedy takové, do nichž lze zapsat více hodnot a tyto hodnoty pomocí podmínek a dalších klíčových slov vyvolávat. Na následujícím příkladu si ukážeme, jak takový parametr vytvořit.

Příklad 4 – Vytvoření multiparametru

Na kartě Správa v tabulce parametrů vytvoříme textový parametr s názvem „vyska“ a potvrdíme. Klikneme pravým a v seznamu vybereme „Upravit seznam s více hodnotami“. Následně v editačním okně zapíšeme, jaké hodnoty chceme parametru přiřadit. Do horního pole „Přidat nové položky“ přidáme hodnotu „NÍZKÉ“ a kliknutím na tlačítko „Přidat“ se nám hodnota přiřadí k parametru. Obdobným způsobem přiřadíme k parametru i zbývající hodnoty „STŘEDNÍ“ a „VYSOKÉ“.





K multiparametrům vytvořených zde se ještě vrátíme v sekci „Podmínky“ a „Formuláře“.

2.2 iVlastnosti

Ivlastnosti jsou takový druh vlastností, které si nese každý vytvořený dokument v Inventoru. V podstatě se jedná o proměnné, kterým buď pomocí dialogového okna nebo iLogicu přiřazujeme hodnoty, se kterými můžeme dále pracovat. Rozlišujeme 3 typy iVlastností:

- Systémové
- Fyzikální
- Uživatelské

Systémové

Pro zápis do systémových iVlastností je nutné do fragmentu zapsat anglický název této vlastnosti.

```
iProperties.Value("Summary", "Title") = "Zadaná hodnota"
```

Summary (Souhrn)		Project (Projekt)		Status (Stav)	
Title	(Nadpis)	Location	(Umístění)	Revision Number	(Číslo revize)
Subject	(Předmět)	File subtype	(Podtyp souboru)	Stock Number	(Skladové číslo)
Author	(Autor)	Part Number	(Číslo součásti)	Status	(Stav)
Manager	(Vedoucí)	Description	(Popis)	Design State	(Stav návrhu)
Company	(Společnost)	Revision Number	(Číslo revize)	Checked By	(Zkontroloval)
Category	(Kategorie)	Project	(Projekt)	Checked Date	(Kontrolováno dne)
Keywords	(Klíčová slova)	Designer	(Kreslil)	Eng. Approved By	(Inž. Schválen kým)
Comments	(Poznámky)	Engineer	(Schválil)	Eng. Approved Date	(Inž. Schválen dne)
		Authority	(Zodpovídá)	Mfg. Approved By	(Výr. schválena kým)
		Cost Center	(Nákladové středisko)	Mfg. Approved Dyte	(Výr schválena dne)
		Estimated Cost	(Odhad ceny)	Checked out By	(Vydal)
		Creation Date	(Datum vytvoření)	Checked out	(Vydáno)
		Vendor	(Dodavatel)	Checkout Workgroup	(Pracovní skupina vydání)
		WEB Link	(Webový odkaz)	Checkout Workspace	(Pracovní prostředí vydání)

Seznam anglických výrazů pro jednotlivé systémové iVlastnosti

Fyzikální

```
iProperties.Material          - Materiál součásti
iProperties.PartColor         - Barva součásti
iProperties.Mass              - Hmotnost
iProperties.Volume            - Objem
iProperties.Area              - Povrch
iProperties.CenterOfGravity.X - Těžiště (v ose x) – možno nahradit za y/z
```

Uživatelské

Do uživatelských iVlastností lze zapisovat stejně jako do systémových, jen do závorky na místo popisu karty zapíšeme „custom“ a mezi další uvozovky zapíšeme název iVlastnosti – pokud uživatelská vlastnost neexistuje, automaticky se vytvoří:

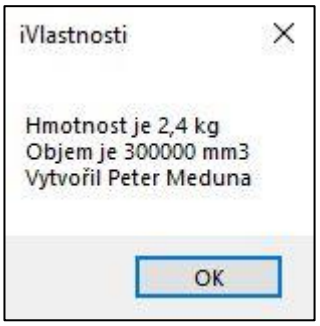
```
iProperties.Value("Custom", "Libovolný název") = "Zadaná hodnota"
```

Podle typu hodnoty proměnné se iVlastnost rozděluje na Text, Počet, Datum, Ano/ne

```
iProperties.Value("Custom", "Text") = "Textovy rezetec"
iProperties.Value("Custom", "Cislo") = 20
iProperties.Value("Custom", "Boolean") = True
iProperties.Value("Custom", "Datum") = DateValue("23.11.2020")
```

Příklad 5 – Zapisování iVlastností

Nemáme-li, nastavíme naší součásti materiál, například ocel. Otevřeme nové pravidlo a opíšeme řádky níže:




```
MessageBox.Show("Hmotnost je " & Round(iProperties.Mass,1)
& " kg" & vbLf & "Objem je " & Round(iProperties.Volume,1)
& " mm3" & vbLf & "Vytvořil " &
iProperties.Value("Summary", "Author"), "iVlastnosti")
```

Pravidlo nám vypíše definované iVlastnosti do vyskakovacího okna.

2.3 Lokální proměnné

Lokální proměnné jsou takový druh proměnné, které fungují pouze v pravidle a po dobu běhu pravidla si uchovají hodnotu, kterou jim přiřadíme (text, číslo, true/false). V pravidle jsou označeny `hnědou barvou`.

2.4 Typy proměnných

Pro zápis v programu se proměnné zpravidla deklarují, tzn. přiřazuje se k nim typ. K tomu slouží příkaz `Dim [NázevProměnné] As [TypProměnné]`. Typů proměnných existuje mnoho, pro naše příklady však postačí čtyři základní typy.

Integer

Pomocí tohoto typu deklarujeme celočíselné proměnné, přičemž velikost zapsaného čísla je maximálně 32 bitů (-2,147,483,648 až 2,147,483,647). Potřebujeme-li větší číslo, použijeme typ Long, který zapíše číslo o velikosti 64 bitů (-9,223,372,036,854,775,808 až 9,223,372,036,854,775,807).

Double

Typ Double je podobný jako Integer, ale daleko přesnější, jelikož dokáže zapsat i desetinná čísla.

Příklad 6 – Využití typu Integer a Double

Na tomto příkladu si ukážeme jednoduchou matematickou operaci. Zkopírujeme si jednotlivě následující sloupce, uložíme a spustíme. Mějme na paměti, že Integer nám ukáže pouze celá čísla.

```
Dim x As Integer
Dim y As Integer
Dim z As Integer
x = 50
y = 2
z = x * y
MessageBox.Show(z, "Příklad")
```

```
Dim x As Double
Dim y As Double
Dim z As Double
x = 2,5
y = 3,5
z = x * y
MessageBox.Show(z, "Příklad")
```

String

Proměnná String deklaruje text a textové řetězce.

Příklad 7 – Zobrazení textu

Pro každý typ se proměnné definují stejně, liší se pouze zápis.

```
Dim x As String
x = "kafe"
MessageBox.Show("K pití si dám: " & x, "Chci")
```

Praktický příklad použití řetězce string - program čte text v uvozovkách a dle něj následně přiřadí určité vlastnosti jednotlivým hladinám při exportu do DXF:

```
Dim sOut As String
sOut = "FLAT PATTERN DWG?AcadVersion=2000" _
& "&RebaseGeometry=False" _
& "&OuterProfileLayer=VNĚJŠÍ ŘEZ" _
& "&SimplifySplines=True" _
& "&SplineTolerance=0.01" _
& "&AdvancedLegacyExport=False" _
& "&MergeProfilesIntoPolyline=False" _
```

S řetězci lze pracovat i více způsoby než jen zapisováním. Lze použít například funkce:

1) pro oddělení textu

```
doc = ThisApplication.ActiveDocument.FullFileName
x = Right(doc, 7)
MessageBox.Show(x, "Title")
y = Left(doc, 3)
MessageBox.Show(y, "Title")
z = Len(doc)
MessageBox.Show(z, "Title")
```

2) pro separaci řetězce dle znaku

```
Dim Delic() As Char = {"\"}
cesta = "C:\FIRMY\ŠKOLENÍ\DATA\S01.ipt"
usek = cesta.Split(Delic)

cislo_soucasti = usek(4)

MessageBox.Show(cislo_soucasti, "Title")
```

Znak je označen jako `Char` a lze ho považovat za řetězec o jednom písmenu/znaku. Funkcí Split rozdělíme řetězec dle tohoto znaku na úseky a necháme si vypsat jen ten úsek, který, potřebujeme.

3) pro zapsání řetězce velkými či malými písmeny

Vytvoříme textový parametr FIRMA a zapíšeme její hodnotu. Výsledek bude iVlastnost zapsaná velkými písmeny.

```
iProperties.Value("Custom", "FIRMA") = UCase(FIRMA)
```

Opačně lze zapsat výsledek i malými písmeny.

```
iProperties.Value("Custom", "FIRMA") = LCase(FIRMA)
```

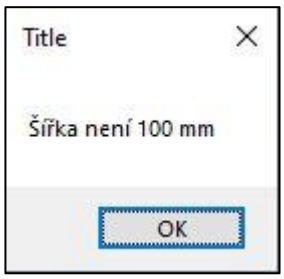
Boolean

Proměnná Boolean deklaruje jen hodnotu pravda/nepravda. Samostatně se moc nepoužívá, spíše bývá členem podmínky nebo většího celku, stejně jako v následujícím příkladu.

Příklad 8 – Použití proměnné Boolean

Vraťme se k příkladu 3. kde jsme si definovali parametry sirka a delka. Nejprve potlačíme pravidlo „obsah“, jelikož zde máme definovanou šířku na 100 mm a nedovolilo by nám součást dále upravovat. Nyní si zde vytvořme nové pravidlo s názvem „Kontrola“, kam si zapíšeme kód níže. Uložíme a spustíme. Nic se nestalo, protože šířka je stále 100 mm, což splňuje podmínku větší/menší.
Změňme v tabulce parametrů šířku třeba na 50 mm a objeví se nám hláška ve vyskakovacím okně.

```
Dim rozmer As Double
rozmer = 100 mm
Dim kontrola As Boolean

If rozmer <> sirka Then
        kontrola = True
        If kontrola = True Then
                MessageBox.Show("Šířka není 100 mm", "Title")
        End If
Else kontrola = False
        If kontrola = False Then
                MessageBox.Show("Šířka je 100 mm", "Title")
        End If
End If
```

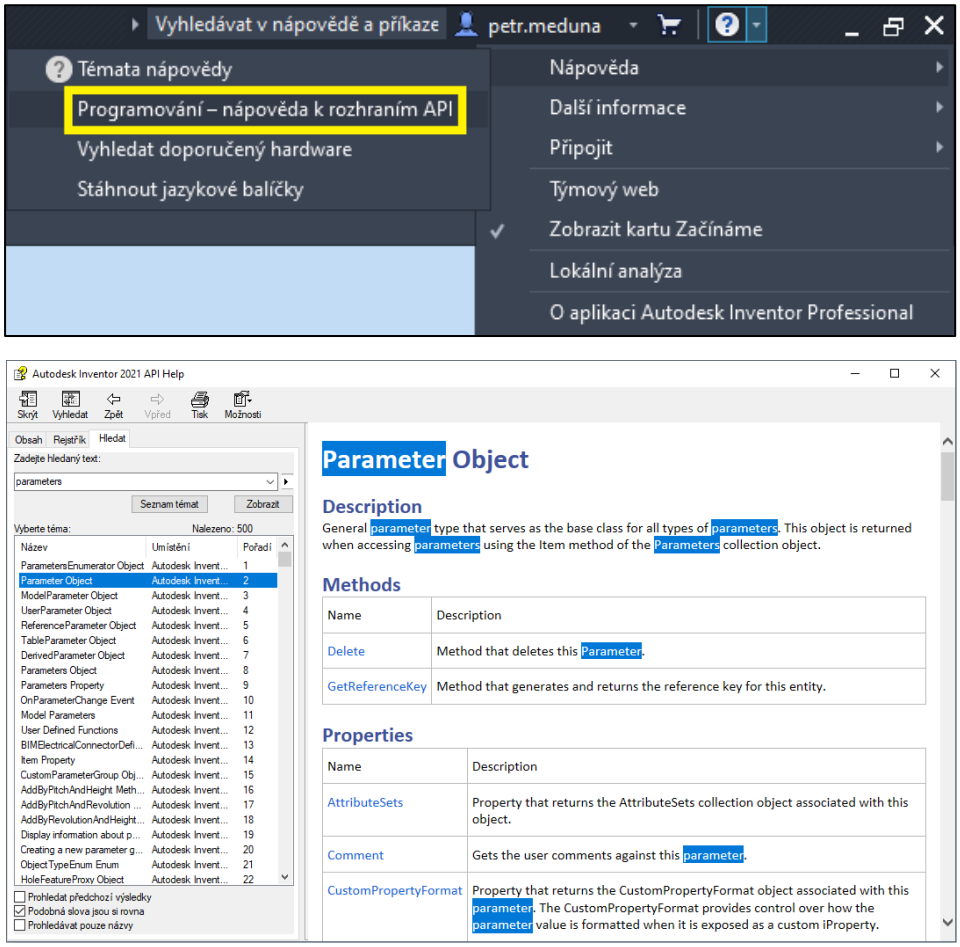


Máme dříve definovaný parametr sirka. K tomu si přidáme lokální proměnnou rozmer, podle které budeme kontrolovat a druhou proměnnou kontrola, která doplní splnění podmínky. Podmínka samotná nám pak říká, že pokud je šířka větší nebo menší a kontrola = pravda (tedy že ji chceme spustit), vyskočí nám hláška s popisem. Proměnná kontrola zde není třeba, jde pouze o ukázku implementace proměnné typu Boolean do pravidla.

Object

Pojmem object souhrnně nazýváme takový druh proměnné, která zastupuje specifickou položku či součást v Inventoru. Patří sem plochy, náčrty, prvky, dokumenty, a spousta dalších. Každý objekt má svůj specifický název a je třeba jej správně definovat pro přístup k jeho funkcím a vlastnostem.

Nápovědu pro zapisování objektů nalezneme přímo v Inventoru v nápovědě pod položkou „Programování – nápověda k rozhraním API“. Otevřeme se nápovědní okno, kde si můžeme požadovaný objekt či entitu vyhledat i s možnostmi jejich zápisu do programového prostředí iLogicu.



Okno nápovědy pro iLogic

Příklad 9 – Použití objektu a vytvoření parametru

Na příkladu níže si pomocí objektů definujících dokument zkusíme vytvořit parametr.

```
Dim oDoc As PartDocument = ThisDoc.Document

Dim oPartCompDef As PartComponentDefinition = oDoc.ComponentDefinition

Dim oParams As Parameters = oPartCompDef.Parameters

Dim oUserParams As UserParameters = oParams.UserParameters

oUserParams.AddByValue("ROZMER_X",100,UnitsTypeEnum.kMillimeterLengthUnits)
```

3 Podmínky

Podmínky neboli podmíněné příkazy jsou programové funkce, které dle zadaných kritérií vykonají určitou akci. Existuje několik druhů, pro potřeby tohoto školení však postačí základních 5 druhů.

3.1 IF-THEN-ELSE

V příkazu If-Then-Else se jedna sada akcí provede, pokud je podmínka pravdivá, a druhá sada akcí se provede, pokud podmínka není pravdivá. Po provedení příkazů pro pravdivou nebo nepravdivou podmínku pokračuje programové řízení dalším příkazem.

Příklad 10 – Ukázka použití podmínky If-Then-Else

V tomto příkladu si ukážeme změnu barvy součásti na základě velikosti parametru z vysunutí. V naší vytvořené součásti z příkladu 3 uděláme vysunutí z náčrtu 250 mm. Vytvoříme nové pravidlo s názvem „barva“ a napíšeme si následující kód:

```
If d2 = 250 Then
        iProperties.PartColor = "červená"
ElseIf  d2 > 250 Then
        iProperties.PartColor = "fialová"
ElseIf  d2 < 250 Then
        iProperties.PartColor = "žlutá"
End If
```

Zkontrolujte, zda parametr vysunutí je opravdu d2, případně opravte. Po spuštění pravidla nám pravidlo přiřadí barvu součásti v závislosti na parametru vysunutí, v tomto případě d2. Pokud je první podmínka splněna, součást se obarví červeně, pokud ne, program zkontroluje druhou podmínku pomocí příkazu Else if (jinak když).

Pro zápis více podmínek najednou lze použít předpřipravené operátory. Zavřeme pravidlo a vytvoříme nové, kam si zapíšeme:

```
If d2 = 250 And sirka = 100 Then
      iProperties.PartColor = "červená"
      MessageBox.Show("Splněny jsou obě podmínky.","Title")
End if
```

V podmínce je použito And (a také), což znamená, že součást se obarví, pokud jsou splněny obě rozměrové podmínky. Existuje i operátor AndNot (A také ne), který pro splnění podmínky vyžaduje jeden parametr platný a druhý zároveň neplatný.

```
If d2 = 250 Or sirka = 100 Then
      iProperties.PartColor = "červená"
      MessageBox.Show("Splněna je alespoň jedna podmínka.","Title")
End if
```

Zde je použit operátor `Or` (nebo), který nám říká, že je podmínka splněna, když alespoň jeden z parametrů souhlasí. Lze použít i `OrElse` (nebo jinak), který programu značí, že pokud je neplatná první podmínka, podívá se na druhou a pokud je pravdivá, podmínka prošla v pořádku.

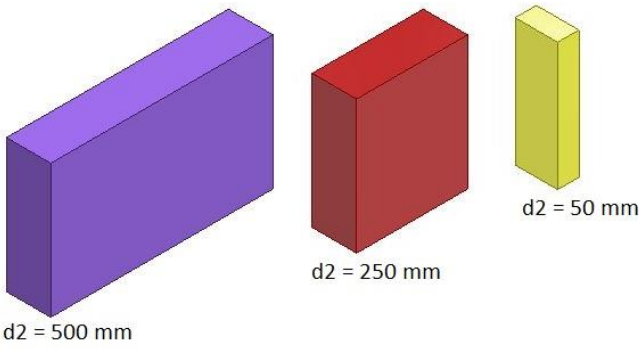
3.2 Select Case

Pokud máme k výběru více možností než 2 (větší a menší než hodnota) a nechceme využívat více podmínek `IF` za sebou, je k dispozici funkce `SELECT CASE`. Pomocí této funkce se vykoná konkrétní operace pro konkrétní variantu či rozsah hodnot parametru.

Příklad 11 – Užití Select Case

V předešlém příkladu jsme si za pomoci podmínky If zkusili změnit barvu součásti na základě velikosti vysunutí a v příkladu 4 jsme si ukázali zase vytvoření multiparametru. Nyní vše spojíme dohromady. Vytvoříme nové pravidlo s názvem „SelectCase“ a opíšeme následující kód:

```
Select Case vyska
Case "VYSOKÉ"
        d2 = 500
Case "STŘEDNÍ"
        d2 = 250
Case "NÍZKÉ"
        d2 = 50
end select
```



Již dříve jsme si definovali parametr s více hodnotami, mezi kterými můžeme volit. Select Case v tomto případě přiřadí každé hodnotě z parametru číslo, a následně po spuštění pravidla mění velikost dle zvolené hodnoty parametru vyska. V kombinaci s předchozím pravidlem nám při každém překliknutí hodnoty změní součást barvu.

Select case ale neslouží jen k přiřazení hodnoty multiparametru jako v příkladu výše. Lze s ním podmínkovat například vypínání/zapínání prvků či kontrolovat hodnoty parametrů, což lze i s obyčejnými podmínkami if-then-else, takto napsaný kód se ale stává příliš zdlouhavý.

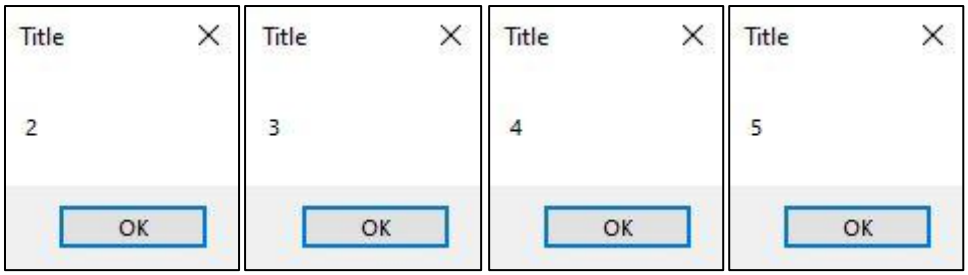
3.3 While

While je v podstatě cyklus, který do splnění podmínky vykonává neustále dokola tutéž operaci. V rámci iLogicu jde o minoritně využívanou funkci. Ve většině případů si vystačíme se dvěma předchozími.

Příklad 12 – Ukázka použití While

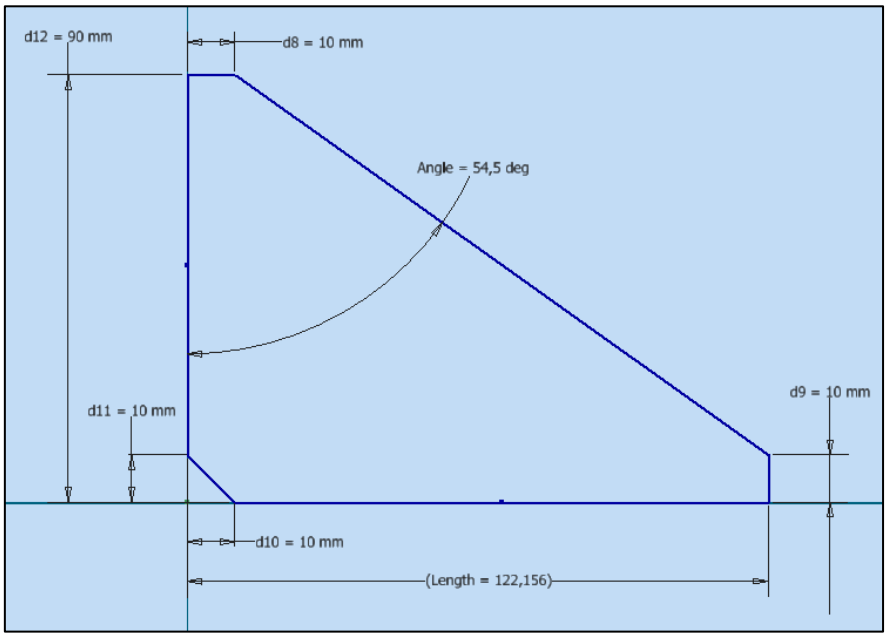
V tomto příkladu máme proměnnou i, ke které se v každém dalším cyklu přičte 1, dokud se nedostane na číslo 5, potom se program ukončí. Prakticky to znamená, že nám dle programu vyskočí vždy po každém cyklu nové okno s číselnou hodnotou i+1.

```
Dim i As Integer
i = 1

While i<5
        i = i + 1
        MessageBox.Show(i, "Title")
End While
```



Příklad 13 – Použití while na modelu

V nové součásti si vytvoříme náčrt dle obrázku níže. Dbáme na to, aby byl náčrt plně zakótovaný a aby seděly názvy parametrů. Výchozí hodnota pro úhel (Angle) je 30. Dále si vytvoříme uživatelský parametr s název Max_L, kterému zadáme hodnotu 120 mm. Referenční odvěsnu (v závorce) v parametrech přejmenujeme na Length.



Zapíšeme následující řádky. Cílem je vytvořit pravidlo, které po změně výšky nastaví hodnotu Lenght na cílovou hodnotu Max_L v závislosti na rozsahu úhlu.

```
Parameter.UpdateAfterChange = True

Parameter("Angle") = 30
i = 0
If Parameter("Length") < Max_L Then
        While (Parameter("Length") < Max_L And Parameter("Angle") < 60)
                Parameter("Angle") = Parameter("Angle") + 1
                i = i + 1
                If i >= 50 Then
                        MessageBox.Show("Operace překročily předpokládaný
rozsah.", "Varování")
                        Exit While
                End If
        End While
End If

iLogicVb.UpdateWhenDone = True
```

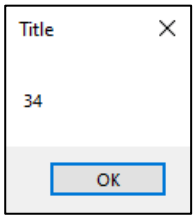
3.4 Sub

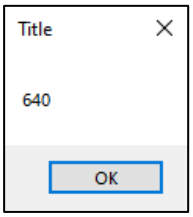
Jedná se o další cyklický podmíněný příkaz. V podstatě si kód rozdělíte na jednotlivé díly (sub). V kódu musí být na začátku vždy jeden díl hlavní (Sub main()), kde se definují proměnné a vyvolávají jednotlivé podprogramy, kterých následně může být libovolný počet. Program si v hlavním dílci příkazem „Call“ vyvolá podprogram s definovaným názvem a provede jej. Poté se opět vrátí do hlavního dílce. Takto pokračuje do doby, než vykoná všechny zadané podprogramy. Následující příklad ilustruje použití příkazu Sub.

Příklad 14 – Ukázka použití Sub

Vytvoříme si nové pravidlo s názvem „sub“ a vložíme následující kód:

```
Sub Main()
        Dim x As Integer
        Dim y As Integer
        Dim z As Integer
        Call priklad1()
        Call priklad2()
End Sub

Sub priklad1()
        x = 6
        y = 28
        z = x + y
        MessageBox.Show(z, "Title")
End Sub

Sub priklad2()
        x = 10
        y = 64
        z = x * y
        MessageBox.Show(z, "Title")
End Sub
```





Na počátku si v Sub main() definujeme proměnné a vyvoláme jednotlivé podprogramy. V samotných podprogramech se nachází hodnoty proměnných a operace, kterou s nimi chceme provést. Na konci podprogramu je vyskakovací okno, kde se nám zobrazí výsledek operace. Výhoda tohoto cyklu je ta, že definujeme jen 3 proměnné, kdežto kdybychom chtěli psát takový program pouze deklaracemi jako u příkladu 6, museli bychom si definovat spoustu proměnných (x1, x2,…).

Praktických užití takového cyklu je spoustu. Lze jej použít například u automatické tvorby uživatelských parametrů, kdy po spuštění pravidla program dle Sub main () zkontroluje, zda se nejedná o součást z obsahového centra a pokud ne, vyvolá podprogram, který zapíše parametry do modelu.

3.5 Try-Catch-End Try

Tento typ příkazu v podstatě uživateli umožní obejít výchozí mechanismus zpracování chyb a řídit posloupnost událostí následujících po chybě. To znamená, že i když po příkazu try nastane error, třeba z důvodu, že parametr neexistuje, můžeme jednoduše pokračovat dál tím, že parametr necháme vytvořit nebo použijeme vyskakovací okno, které nás upozorní na chybu.

Příklad 15 – Vytvoření iVlastnosti

Vytvoříme si nové pravidlo s názvem „TryCatch“ a zkopírujeme řádky níže:

```
Try
        iProperties.Value("Custom", "Test") = True
Catch
        MessageBox.Show("iVlastnost nebyla vytvořena", "Title")
Finally
        MessageBox.Show("Akce provedená nezávisle na Try a Catch", "Title")
End Try
```

V bloku Try se nachází nastavení uživatelské iVlastnosti. Program se pokusí iVlastnost vytvořit a pokud by se tak nestalo z důvodu nějakého erroru, vyskočí hláška z bloku Catch. Příkaz Finally pak provede požadovanou akci, ať už se požadavek splnil či nikoliv. Zde nutný není, avšak v tomto podmíněném příkazu musí být vždy blok buď s Catch nebo Finally.

Příklad 16 – Zapisování parametru

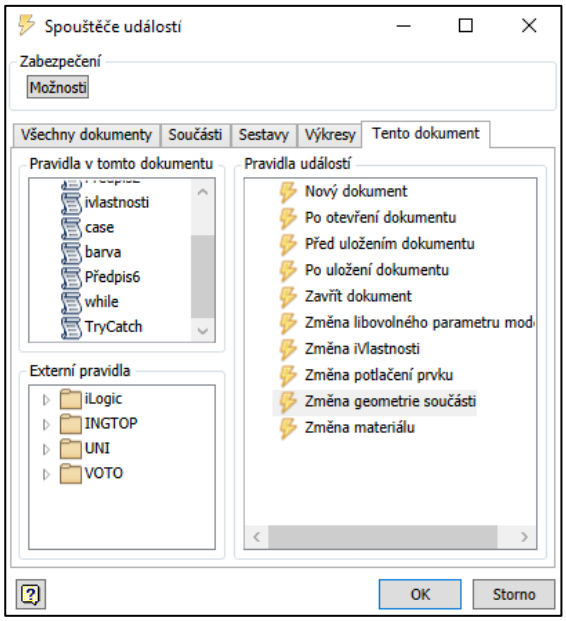
Zde budeme postupovat stejně jako u příkladu 9. Ten však obsahuje chybu, která zapříčiní, že po každém spuštění pravidla se vytvoří nový parametr a pokud takový parametr již existuje, přidá se k němu rostoucí index (ROZMER_X_1, ROZMER_X_1_2, ROZMER_X_1_2_3, …). To lze opravit přidáním příkazu Try, který zkontroluje, že proměnné dokáže přiřadit existující parametr a pakliže tomu tak je, příkaz pod Catch se neprovede.

```
Dim oDoc As PartDocument = ThisDoc.Document

Dim oPartCompDef As PartComponentDefinition = oDoc.ComponentDefinition

Dim oParams As Parameters = oPartCompDef.Parameters

Dim oUserParams As UserParameters = oParams.UserParameters

Try
x = Parameter("ROZMER_X")
Catch
oUserParams.AddByValue("ROZMER_X",100,UnitsTypeEnum.kMillimeterLengthUnits)
End try
```

4 Spouštěče událostí

4.1 Spouštěče událostí

Prozatím jsme si ukázali jak spustit pravidlo skrze dialogové okno iLogicu, případně ručně poklikem. Tento způsob je velmi nepraktický, zvlášť máme-li mnoho pravidel. Pro usnadnění funkce spouštění, kdy potřebujeme třeba rozběhnout několik pravidel naráz, slouží funkce „Spouštěče událostí“, které nalezneme hned vedle nástroje pro přidání pravidla (Správa > iLogic > Spouštěče událostí).



Po kliknutí se nám otevře editor spouštění, kde na levé straně máme rozdělené interní a externí pravidla a na straně druhé karty pro jednotlivé typy dokumentů s přiřazenými spouštěči, které nám pravidla rozběhnou. Nutno dodat, že námi vytvořená interní pravidla, která jsme si vytvářeli v rámci tohoto školení, slouží jen záložka „Tento dokument“, jelikož pravidla se váží jen k součásti a nelze je tedy aplikovat na výkres či sestavu. K tomu slouží pravidla externí, o kterých si ale řekneme později.

Příklad 17 – Funkce spouštěče

Vytvořme si pravidlo s názvem „hmotnost“ a zkopírujme následující řádek:

```
MessageBox.Show(iProperties.Mass & " kg", "Hmotnost součásti")
```

Uložíme a spustíme. Vyskočí nám okno s napsanou aktuální hmotností součásti. Otevřeme si Spouštěče událostí a aktivujeme kartu Tento dokument. Zde vidíme všechny interní pravidla. Vezmeme pravidlo „hmotnost“ a přetáhneme na spouštěč „Změna geometrie součásti“. Potvrdíme OK a nyní se při každé změně geometrie aktualizuje hmotnost a vyskočí nám okno s aktuální hmotností.

4.2 iTrigger

iTrigger je další nástroj na spouštění pravidel. Nalezneme jej hned pod funkcí Spouštěče událostí. Funguje podstatně jinak, než způsob popsaný výše. Při kliknutí na tlačítko iTrigger se do součásti vytvoří parametr iTrigger0, který následně musíme ještě definovat v každém pravidle, které chceme přes iTrigger spustit. Vždy připíšeme následující:

```
trigger = iTrigger0
```

Nyní vždy, když klikneme na tlačítko iTrigger přičte se k hodnotě tohoto parametru +1, což je změna, díky níž se pravidlo obsahující tento řetězec spustí.

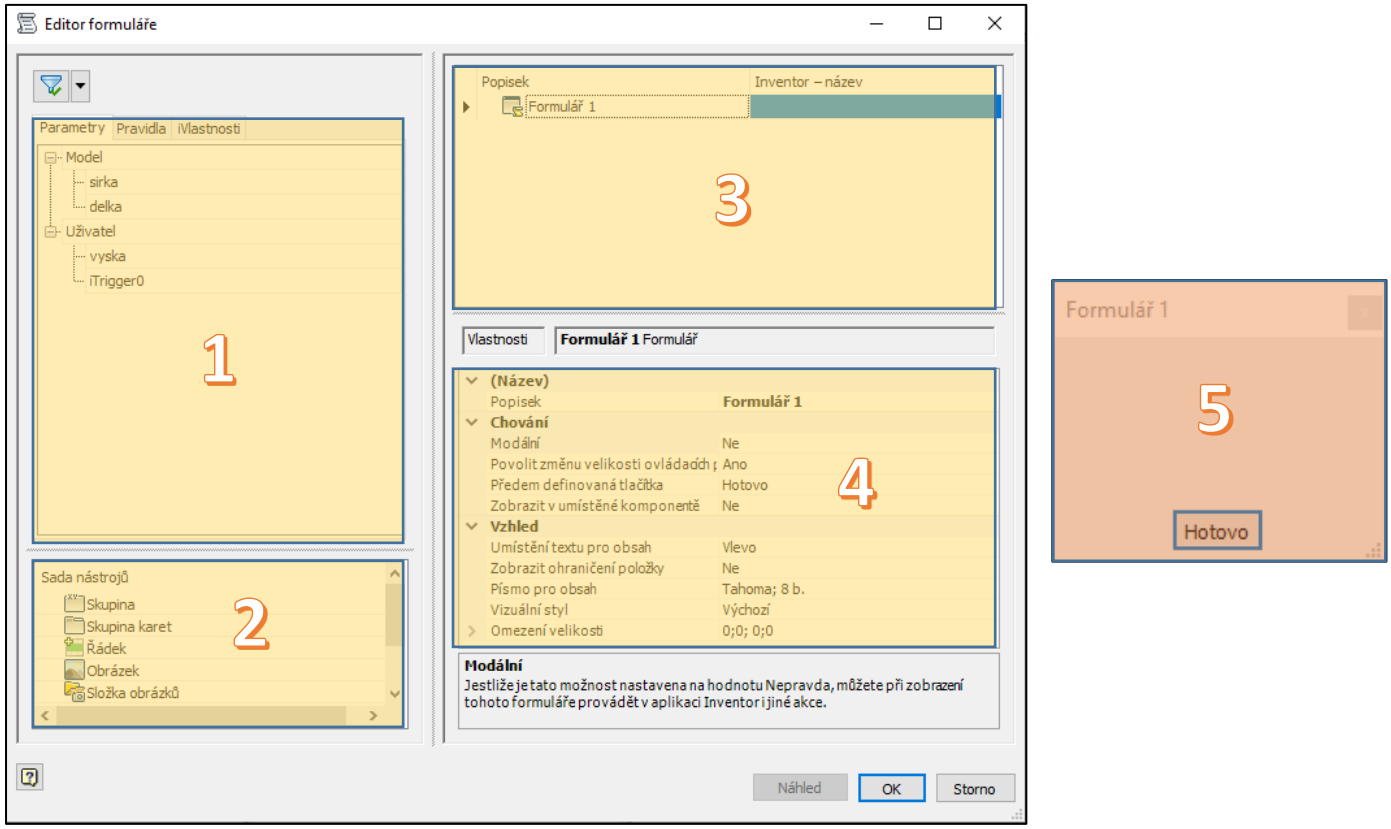
Příklad 18 – Spouštění pravidla přes iTrigger

Můžeme si to hned ukázat na dalším příkladu. Klikneme na iTrigger a řetězec z textu si zkopírujeme do pravidla „hmotnost“ z příkladu 14 a třeba do pravidla „iVlastnosti“ z příkladu 5. Takto změněné pravidlo vždy uložíme a spustíme. Klikneme znovu na iTrigger a pravidla se nám teď spouští všechny najednou, respektive tak, jak jsou za sebou seřazená ve stromu prohlížeče iLogic

5 Formuláře

Nedílnou součástí iLogicu jsou i formuláře, které jsou nástrojem pro spojování pravidel, editaci obsahu souboru (změnu parametrů, iVlastností,…), či změny funkce modelu.

Vytvořit takový formulář je jednoduché. V prohlížeči iLogic si přepneme na kartu formuláře a pravým kliknutím do prostoru v nabídce vybereme „Přidat formulář“. Otevře se nám uživatelské okno, které si následně zeditujeme.

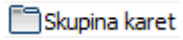
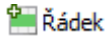
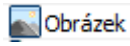
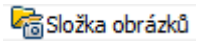
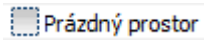


1. **Karty s proměnnými a pravidly**

Zde volíme parametry, ivlastnosti či pravidla a přetahujeme je do sekce 3, kde se nám vytvoří pole pro zadání hodnoty či v případě pravidla tlačítko pro spuštění pravidla.

2. **Grafické prvky**

Ve druhém oddíle se nachází nástroje na formátování dialogového okna a rozložení funkcí a polí v něm. Díly lze kombinovat mezi sebou, například můžeme vytvořit kartu, ve které bude skupina, a položky budou vloženy v řádku. Vysvětlení jednotlivých prvků je níže v tabulce.

	Vytvoří samostatný blok blok položek, který lze v okně sbalit/rozbalit
	Vytvoří kartu, kam přenášíme položky. Mezi kartami lze překlikávat.
	Vytvoří řádek. Veškeré položky přenesené do řádku se nyní vkládají horizontálně.
	Nástroj pro vložení obrázku, například firemního loga.
	Přidá složku s obrázky.
	Přidá do rozhraní prázdný prostor. Slouží především k odsazení položek a obrázků mezi sebou, jelikož můžeme přesně definovat jeho velikost.
	Přidá popisek k položkám, například jednotky – nemusíme je tedy definovat nějakou iVlastností, nýbrž je stačí pouze ve formuláři popsat.
	Přidá do okna posuvný dělič.

3. Strom funkcí a proměnných

V podstatě se jedná o pracovní prostor formuláře. Do grafických prvků, které mezi sebou kombinujeme, zároveň přidáváme i proměnné a pravidla.

4. Vlastnosti

Téměř každý prvek má i své vlastnosti, které můžeme definovat a dotvářet tak pracovní prostředí formuláře. Počet vlastností se liší v závislosti na zvoleném prvku, dá se však říci, že v základu jsou 4 druhy vlastností: Název, Chování, Různé a Vzhled.

Chování

Modální	Hodnota NE – lze provádět při editaci formuláře i jiné akce v Inventoru
Povolit změnu velikosti ovládacích prvků	Kliknutím pravým tlačítkem na formulář se zobrazí tato možnost
Předem definovaná tlačítka	Definuje koncová tlačítka, kterými formulář potvrdíme
Zobrazit v umístěné komponentě	Při umístění komponenty do sestavy se zobrazí formulář
Povolení názvu parametru	Při nastavení daného parametru na hodnotu Pravda se tento prvek (např. pravidlo) aktivuje a lze jej spustit
Název parametru obrázku	Změnou hodnoty textového parametru lze měnit i obrázek
Upravit typ ovládacího prvku	Změní typ zadávání hodnot parametru

Název

Název parametru/pravidla	Zobrazí název daného parametru/pravidla
Popisek	Popisek parametru/pravidla, lze libovolně měnit

Různé

Pouze ke čtení	Zvolíme možnost ano, pokud chceme mít iVlastnost nebo parametr bez možnosti zápisu

Vzhled

Umístění textu pro obsah	Umístění textu, které se má použít pro ovládací prvky obsažené ve formuláři
Zobrazit ohraničení položky	Čarou ohraničí veškeré ovládací prvky formuláře
Písmo pro obsah	Definuje styl textu ve formáři
Vizuální styl	Výběr z předpřipravených šablon formulářů
Omezení velikosti	Omezí velikost prvků nebo formuláře – okno nelze myší roztáhnout
Písmo	Definuje styl textu pro daný prvek
Popisek nástroje	Přidá k prvku bublinku s popisem
Obrázek	Naimportuje obrázek ze složky do formuláře
Zobrazit popisek	Zobrazí popisek obrázku

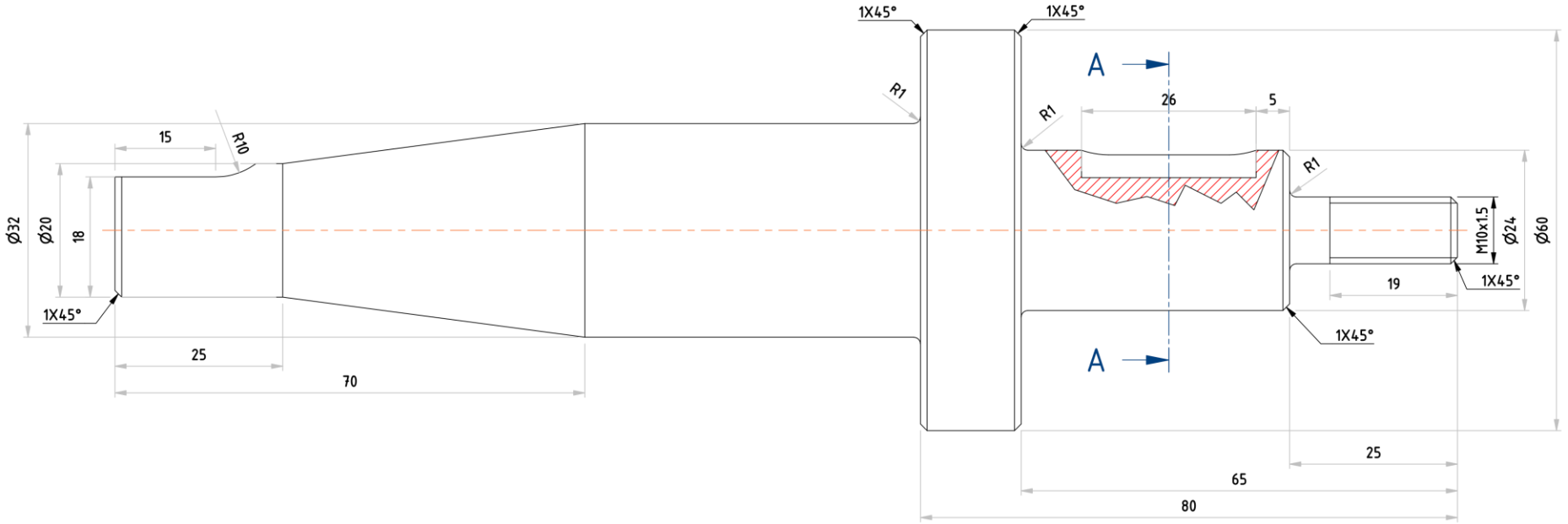
Příklad 19 – procvičení pravidel a vytvoření formuláře

V tomto příkladu si zopakujeme vše doposud probrané. Nakreslíme si náčrt dle přiloženého výkresu. Dbáme na to, aby náčrt byl zakótovaný, jako je na výkrese.

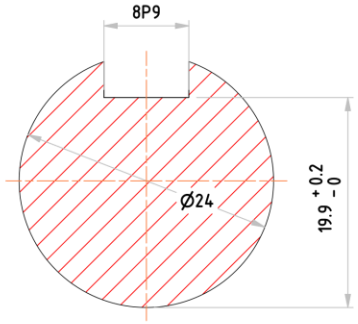
Základní parametry budou proměnlivé (průměr, délka) a přes multiparametr definujeme materiál (ocel, zinek, hliník, olovo). Pravidlem vytvoříme iVlastnosti Projekt, Číslo výkresu, Povrchová úprava, přičemž povrchová úprava (cementování, nitridování, moření) bude vycházet z multiparametru. Obráběné prvky (díra, zkosení, závit) budou označeny červenou barvou a skrz formulář je bude možné přebarvit zeleně a žlutě.

Nejprve si vytvoříme multiparamter PU s povrchovými úpravami v zadání a následně multiparametr MATERIAL s materiály v zadání. Uložíme si nové pravidlo s názvem „iVlastnosti“, kde si jednotlivé vlastnosti definujeme.

```
Try
        iProperties.Value("Custom", "Projekt")= iProperties.Value("Custom",
        "Projekt")
Catch
End Try
Try
        iProperties.Value("Custom", "Číslo výkresu") =
        iProperties.Value("Project", "Part Number")
Catch
End Try
Try
        iProperties.Value("Custom", "Povrchová úprava") = PU
Catch
End Try
```



A-A (2 : 1)



title:		dimension:		note:	
surface finish:		total surface area [mm2]:		material:	weight [g]:
scale:	format:	date:	designer (3D model):	designer (drawing):	
1,5 : 1	A3	30.11.2020	Peter Meduna	ADEON CZ	
title: s02					
gener. tolerances: ČSN ISO 2768-mK / ČSN EN ISO 13920-BF / ČSN EN ISO 5817-B		projection:	project Nr.:	documentation:	drawing Nr.: / index:
					List:1 List0:1

Další pravidlo s názvem „materiál“ bude obsahovat pouze přiřazení parametru k materiálovým knihovnám v Inventoru.

```
iProperties.Material = MATERIAL
iLogicVb.UpdateWhenDone = True
```

Poslední řádek znamená, že se součást aktualizuje hned po skončení pravidla, tudíž se zobrazí provedené změny. Pro správnou funkčnost budeme potřebovat ještě jeden parametr Prebarveni, který bude mít hodnotu pouze Ano/Ne.

Nyní si můžeme napsat pravidlo pro barvu obrábění. Vytvoříme nové a zapíšeme následující kód:

```
If Prebarveni = True Then
        Feature.Color("Vysunutí1") = "Červená"
        Feature.Color("Vysunutí2") = "Červená"
        Feature.Color("Zkosení1") = "Červená"
        Feature.Color("Zaoblení1") = "Červená"
        Feature.Color("Závit1") = "Červená"
Else
        Feature.Color("Vysunutí1") = "As Material"
        Feature.Color("Vysunutí2") = "As Material"
        Feature.Color("Zkosení1") = "As Material"
        Feature.Color("Zaoblení1") = "As Material"
        Feature.Color("Závit1") = "As Material"
End If
iLogicVb.UpdateWhenDone = True
```

Pokud ve formuláři zaškrtneme políčko (viz dále), dle tohoto pravidla se nám obrábění přebarví, pokud odškrteneme, součást si zachová vzhled materiálu.

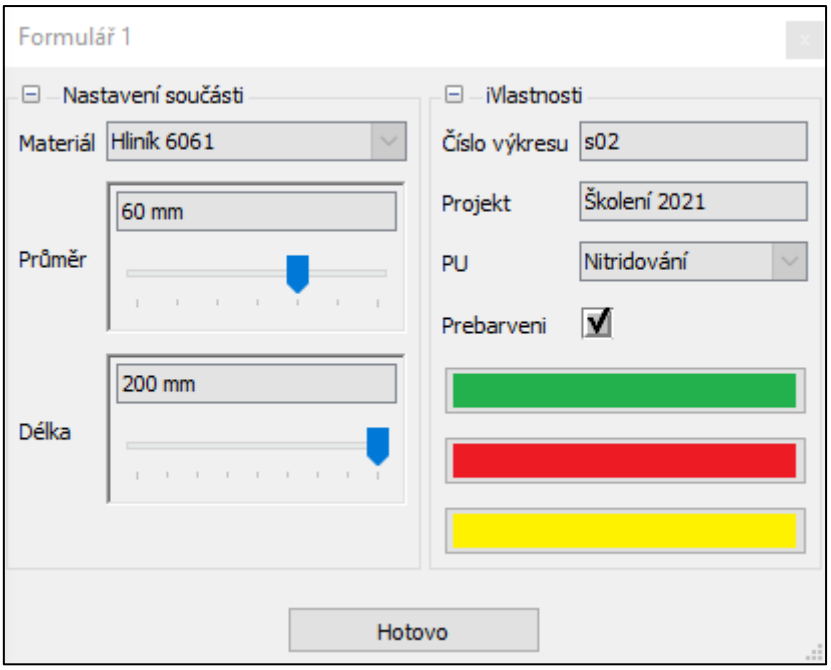
Abychom mohli měnit barvu, vytvoříme si ještě tři pravidla, každé pro jednu. Nazveme je podle barev. Ty vycházejí ze stylu vzhledů, tudíž musíme při zapisování dbát na to, zda se barvy v pravidlech s nimi shodují.

```
Feature.Color("Vysunutí1") = "Zelená"
Feature.Color("Vysunutí2") = "Zelená"
Feature.Color("Zkosení1") = "Zelená"
Feature.Color("Zaoblení1") = "Zelená"
Feature.Color("Závit1") = "Zelená"
If Prebarveni = False Then
        Feature.Color("Vysunutí1") = "As Material"
        Feature.Color("Vysunutí2") = "As Material"
        Feature.Color("Zkosení1") = "As Material"
        Feature.Color("Zaoblení1") = "As Material"
        Feature.Color("Závit1") = "As Material"
End If
iLogicVb.UpdateWhenDone = True
```

Do ostatních dvou si kód zkopírujeme a jen změníme nápis „zelená“ na „žlutá“ a „červená“. K tomu nám může pomoci funkce „Najít a nahradit“ v editoru pravidla, o které jsme si říkali na začátku. Do políčka „Najít“ napíšeme barvu v pravidle např. zelená a do políčka „Nahradit čím:“ žlutá. Potvrdíme a barvy v pravidle se nám nahradí.

Nyní když máme pravidla hotová, můžeme se přesunout na tvorbu formuláře, kam přesuneme iVlastnosti, parametry a pravidla. V novém formuláři si do řádku vložíme dvě skupiny, do první dáme parametr s materiálem, průměrem a délkou. Parametr průměr nastavíme na posuvník, kde minimální hodnota bude 40 mm, maximální 70 mm a velikost kroku bude po 5 mm. Parametr délka bude stejný, jen minimální hodnota bude 160 a maximální 200 mm.

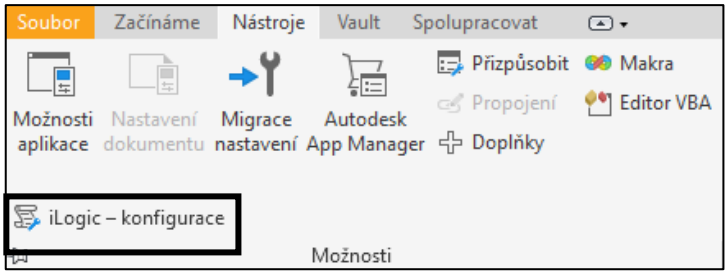
Do druhé skupiny vložíme iVlastnosti Číslo výkresu (nastavíme jako ke čtení), Projekt, PU a Prebarveni, který navolíme jako zaškrtávací políčko. Pod ně přidáme ještě tři pravidla s barvami. V políčkách zrušíme zobrazený text a vložíme obrázky barev ze složky. Výsledný formulář by měl vypadat nějak takto:



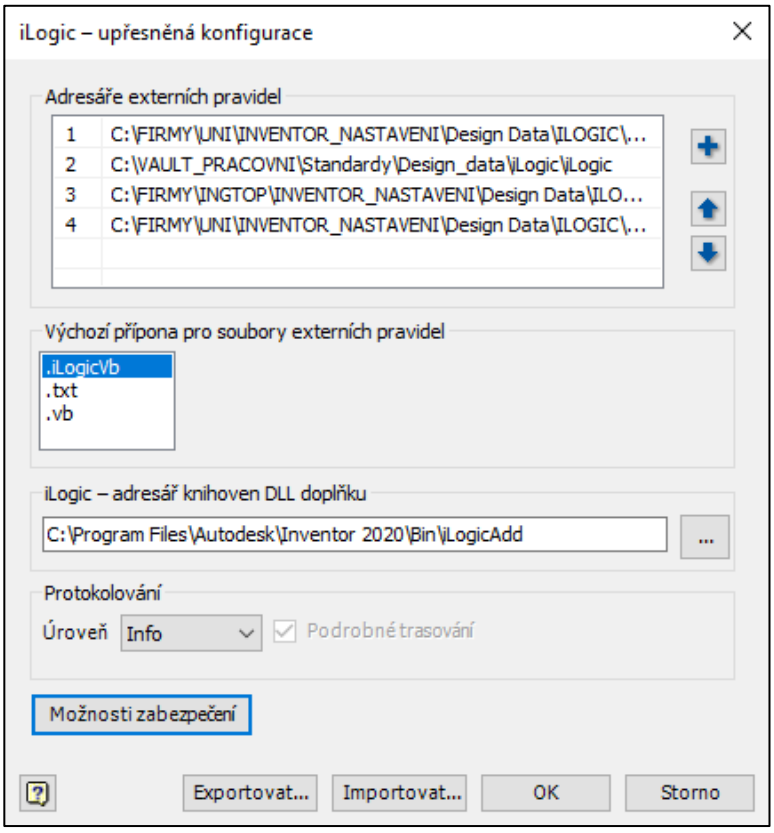
6 Externí pravidla

Všechny pravidla, která jsme doposud vytvářeli (interní), mají jednu nepříjemnou vlastnost: fungují pouze v součásti, ve které jsou napsaná. Jinými slovy nemohou řídit více součástí najednou. Toto lze obejít tím, že pravidlo vložíme do šablony, nicméně v případě potřeby editace pravidla bychom museli zeditovat i všechny součásti, které vycházejí z oné šablony. To je pochopitelně nežádoucí.

Externí pravidla řeší všechny omezení interních pravidel. Jde v podstatě o odkazy na textové soubory s pravidly, které jsou nezávislé na typu otevřeného souboru v Inventoru. Výhodou je, že úpravou pravidla se změní i všechny součásti, které pravidlo využívají.



Konfiguraci cesty k externím pravidlům najdeme na kartě *Nástroje -> Možnosti -> iLogic –konfigurace.* Zde si v dialogovém okně po kliknutí na symbol + můžeme přidat cestu ke složce s externími pravidly, která se většinou ukládají do složky iLogic v Design datech.



V zásadě jsou dva způsoby vytvoření takového pravidla. V okně prohlížeče iLogic v záložce *Externí pravidla* klikneme pravým tlačítkem a zvolíme položku *Vytvořit externí pravidlo*. Zadáme název pravidla a uložíme. Pravidlo se přidá do stromu pod skupinu Ručně přidané pod složku, jakou jsme zvolili.

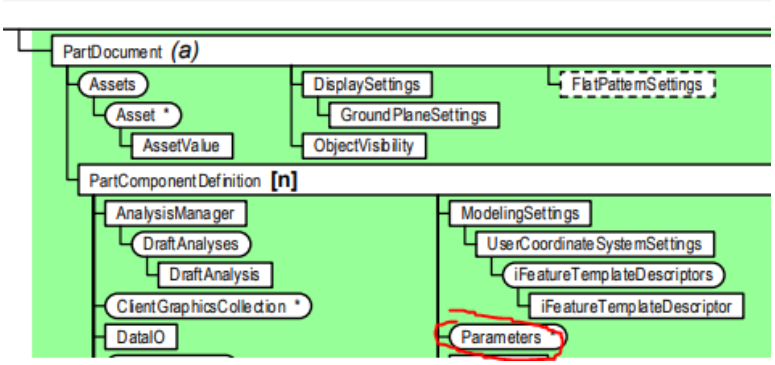
Druhým způsobem je vytvoření pravidla přímo do textového dokumentu, který uložíme s danou příponou (.iLogicVb/.txt) do složky v Design Datech a následně si přidáme cestu k souboru v konfiguraci.

Editovat takové pravidlo je velmi snadné, jelikož jej můžeme otevřít a upravit přímo ze stromu iLogic prohlížeče, případně můžeme editovat rovnou textový dokument bez použití Inventoru (nejčastěji přes *Poznámkový blok*). Zde je však menší problém v tom, že Poznámkový blok nedisponuje žádnou kontrolou správnosti kódu jako editor pravidel iLogic, tudíž o funkčnosti/nefunkčnosti se přesvědčíme až po spuštění pravidla.

Ke psaní externích pravidel však musíme přistupovat jinak než k zápisu interního pravidla. Rozdíl je v tom, že externí pravidlo nepracuje s žádným konkrétním dokumentem a také v tom, že některé syntaxe zde nelze použít, například nelze psát přímo název parametru, pravidlo jej bere jako lokální proměnnou.

Má-li pravidlo pracovat s nějakým konkrétním prvkem v sestavě/součásti/výkresu, je třeba k němu získat nejprve přístup vhodnou definicí proměnných zachovávající sestupnou strukturu. To poměrně dobře ilustruje Inventor API Object Model, který je přílohou tohoto školení.

Inventor API Object Model je dokument obsahující veškeré prvky a nástroje, se kterými lze pracovat a využívat je pomocí iLogicu. Dobře znázorňuje vztahy mezi prvky pomocí stromových struktur, takže lze jednoduše dohledat, jak získat přístup k určitému prvku.



Příklad 20 – dohledání parametru

Vezměme znovu pravidlo z příkladu 9, kde jsme pracovali s objekty. Pokud se podíváme na samotný způsob zápisu pravidla a porovnáme jej se strukturou API dokumentu, zjistíte, že se shodují.

```
'Nejprve definujeme dokument, ve kterém pravidlo spouštíme
Dim oDoc As PartDocument = ThisDoc.Document

'Poté definujeme samotnou součást, zda jde o normální či plech
Dim oPartCompDef As PartComponentDefinition = oDoc.ComponentDefinition

'Následně určíme, že pro tuto definici chceme přístup do jejích parametrů
Dim oParams As Parameters = oPartCompDef.Parameters

'A zapíšeme přístup k uživatelským parametrům
Dim oUserParams As UserParameters = oParams.UserParameters

'Dostaneme-li se k uživatelským parametrům, můžeme za tečkou
'vyvolat funkce nebo vlastnosti, kterými daný prvek disponuje
'- v tomto případě jde o funkci AddByValue, tedy "přidej dle zadané
hodnoty"
oUserParams.AddByValue("ROZMER_X",100,UnitsTypeEnum.kMillimeterLengthUnits)
```

V podstatě by šel zápis zapsat do dvou řádků, jelikož každá další proměnná vychází z té předchozí:

```
oUserParams =
ThisDoc.Document.ComponentDefinition.Parameters.UserParameters
oUserParams.AddByValue("ROZMER_Y",100,UnitsTypeEnum.kMillimeterLengthUnits)
```

7 Globální formuláře

U globálních formulářů platí stejná logika jako u externích pravidel. Fungují globálně, tedy pro všechny díly, sestavy a výkresy. Editace formuláře je totožná s formulářem jen pro dokument, jediný rozdíl je v tom, že se zde využívá globálních pravidel namísto interních.

Vytvořit takový formulář můžeme tak, že vytvoříme standardní interní formulář, který ale využívá externích pravidel. Poté ve stromu *iLogic* klikneme na formulář pravým tlačítkem a dáme *Kopírovat*. Nyní se překlikneme do karty *Globální formuláře* a zde po kliknutí pravým tlačítkem dáme *Vložit formulář*. Vytvoří se zde identický formulář, který se však uloží zároveň jako soubor do složky UI v Design Datech a zapíše se zde do kořenového xml souboru **iLogicBrowserUiFormSpecification.xml**, kam se globální formuláře definují v rámci karty Globální formuláře.

Příklad 21 – Vytvoření globálního formuláře

Vezměme si interní formulář z příkladu 19, a překopírujme jej do karty globální formuláře. Následně si otevřeme složku obsahu UI. Vytvořili se dva soubory konkrétně Název_formuláře.xml a Název_formuláře.state.xml. První zmíněný obsahuje soubor všech všech prvků ve formuláři, jejich vlastnosti a velikost, druhý poté zachycuje rozmístění jednotlivých grafických prvků a jejich uspořádání. Tyto dva soubory jsou velmi důležité a je nutné je vkládat do design dat společně, jednotlivě totiž nefungují.

Vraťme se ještě ke kořenovému xml souboru. Ten slouží v podstatě stejně jako dva výše zmíněné xml soubory s tím rozdílem, že upravuje pouze grafický vzhled samotné karty Globální formuláře. Ve standardním formuláři si přidáváme parametry a tlačítka (pravidla), do tohoto si prozměnu přidáváme pouze externí pravidla a formuláře (máme-li jich více).

Slouží tedy jako pohodlnější cesta ke spouštění externích pravidel, jelikož si můžeme jednotlivé skupiny pravidel (patřící jen k součásti například) přidat do skupin a tím tak velké množství pravidel filtrovat.

Globální formuláře se standardně ukládají do C:\Users\Public\Documents\Autodesk\Inventor xxxx\Design Data\iLogic\UI\ , kde xxxx znamená aktuálně používanou verzi Inventoru.