Integración de Herramientas en ArcGIS

VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

Documentación seminarios

Diego Prieto Herráez

Departamento de Matemática Aplicada
Universidad de Salamanca

Enero 2017

Capítulo 1

Integración de Herramientas en ArcGIS

Salamanca, 18 de enero de 2017

1.1. Introducción

Para facilitar la ejecución de ciertas tareas, especialmente ahorrando tiempo en tareas repetitivas o facilitando su ejecución a usuarios no especializados, es posible personalizar el interfaz de las aplicaciones de ArcGIS Desktop (ArcCatalog, ArcGlobe, ArcMap y ArcScene).

Estas personalizaciones se implementan fundamentalmente mediante los siguientes lenguajes de programación:

- Avenue.
- Java.
- Python, con sus librerías ArcPy.
- Visual Basic, con sus librerías ArcObjects.
- Visual Studio .Net.

De ellos, el lenguaje Avenue se utilizó en las versiones primitivas de ArcGIS, entonces llamado ArcView; posteriormente se comenzó a utilizar la arquitectura de componentes COM de Microsoft con los lenguajes Visual Basic y .Net, siendo esta la opción más utilizada para personalizar ArcGIS hasta la fecha. Con el fin del soporte de Visual Basic por parte de Microsoft y la necesidad de utilizar tecnologías multiplataforma (la arquitectura de componentes COM de Microsoft no lo es) se abrió la puerta para que ESRI iniciase su andadura con Python.

Aunque la tendencia actual es que ESRI incorpore toda o casi todas las funcionalidades de ArcGIS en Python, de momento el acceso completo a

Figura 1: Icono de los *Add-Ins*.

todas las funcionalidades de ArcGIS solo es posible mediante la personalización con ArcObjects. ArcObjects permite personalizaciones complejas de la interfaz y proporciona el potencial de la plataforma .NET, más versátil y flexible que el entorno de Python.

De momento, anque no se pueden desarrollar complejos interfaces mediante el uso de Python, si es posible crear sencillas personalizaciones vía *Add-Ins* que en muchos casos son más que suficientes. Por otra parte, ESRI va dando mayor peso y funcionalidad a Python.

En este seminario nos centraremos en la personalización de la aplicación ArcMap mediante el uso del lenguaje interpretado Python y las librerías ArcPy para crear los *Add-Ins*.

1.2. ¿Qué es un *Add-In*?

Los *Add-Ins* son paquetes que contienen y permiten distribuir personalizaciones del interfaz de ArcGIS Desktop de una forma sencilla. Estos paquetes tienen extensión `.esriaddin` y no son más que un fichero comprimido ubicado en un directorio específico, que se instala sobre la instalación de ArcGIS Desktop existente mediante la *Esri ArcGIS Add-In Installation Utility*.

1.2.1. Tipos de *Add-Ins*

Se pueden crear distintos tipos de *Add-Ins*:

- Botones: no hacen más que ejecutar un código al hacer clic sobre los mismos.
- Herramientas: similares a los botones, pero requieren cierta interacción del usuario con el mapa para ejecutar un código. Además permiten recoger información de la interacción sobre el mapa.

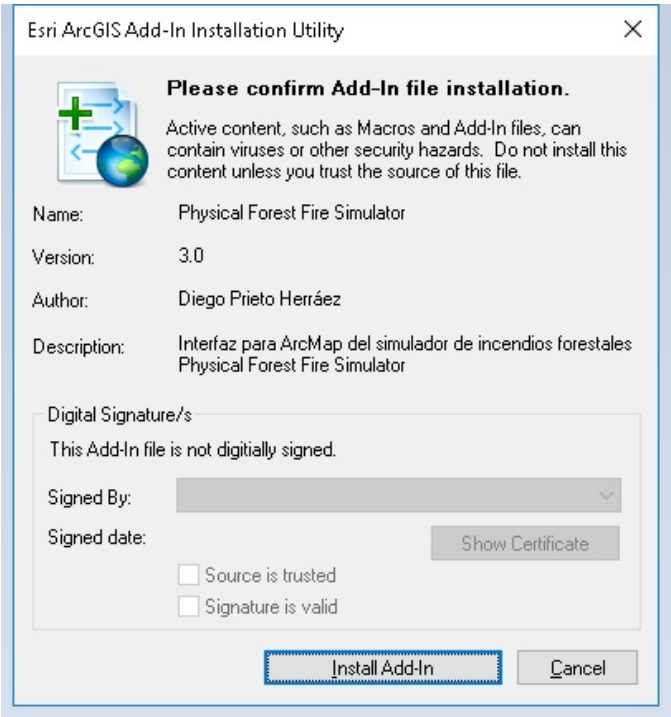

Figura 2: Instalación de un *Add-In*.

- Combos: proporcionan una lista de opciones para que el usuario seleccione una entre las mismas.

Estos *Add-Ins* se ubican sobre el interfaz de ArcGIS Desktop utilizando diferentes tipos de contenedores:

- Menús.
- Barras de herramientas.
- Paleta de herramientas.

Por último las extensiones de aplicación son el tipo de *Add-In* más complejo ya que se encargarán de coordinar la funcionalidad de diversos componentes y serán las responsables de atender a diferentes peticiones y de responder a distintos eventos tales como abrir o cerrar un documento, etc.

1.3. ArcGIS Python Add-In Wizard

Para facilitar la creacción de *Add-Ins*, ESRI ha creado el asistente *ArcGIS Python Add-In Wizard*. Esta herramienta ayuda a crear la estructura del Add-In, creando una plantilla que posteriormente se completará con la lógica deseada.

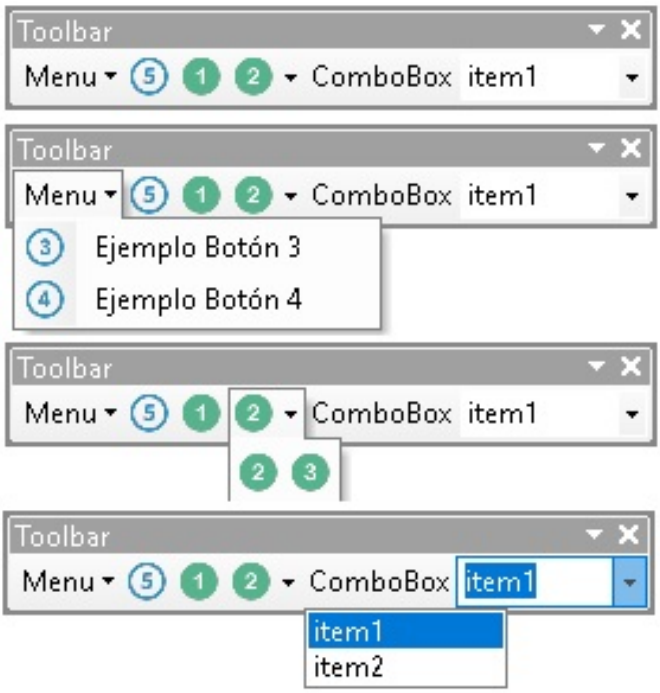

Figura 3: Tipos de *Add-Ins*.

1.3.1. Estructura de un fichero *Add-In*

Al crear un proyecto utilizando el asistente *ArcGIS Python Add-In Wizard*, se despliega una estructura de ficheros muy sencilla. En su nivel más básico, solo está compuesta por dos carpetas y un conjunto de ficheros:

- El fichero `makeaddin.py` es un script de Python que se utiliza para crear el fichero `.esriaddin`.
- El fichero `.esriaddin` es el que se utiliza para distribuir nuestra aplicación. Básicamente es un fichero comprimido (en formato `.zip`, aunque tenga otra extensión) que contiene el fichero `config.xml` y las carpetas `images` e `install`.
- El fichero `config.xml` define el interfaz de usuario así como algunas propiedades estáticas del *Add-In* tales como: nombre, autor, versión, etc.
- La carpeta `images` contiene todos los iconos o imágenes que usa el addin. Estás imágenes pueden tener diferentes formatos (`.bmp`, `.jpg`, `.png`, etc.), pero se aconseja que sean `.png` y que contengan el canal alfa para conservar los colores de la interfaz.
- La carpeta `install` contiene el script de Python (`nombre_proyecto_addin.py`)

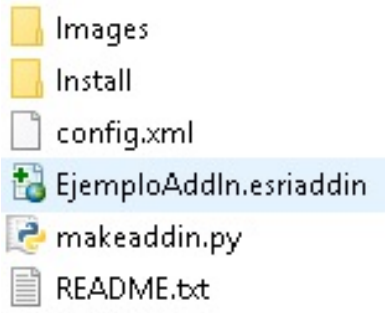

Figura 4: Estructura de ficheros de un Add-In

que ejecuta la lógica del proceso que almacena el *Add-In*. Será en este fichero en el que escribiremos el código que ejecute la aplicación, en los diferentes botones, herramientas, combos, etc.

1.3.2. Creacción de un *Add-In*

El primer paso para crear un *Add-In* es la creación de un proyecto mediante el asistente *ArcGIS Python Add-In Wizard*. Para ello se definen los siguientes parámetros y se guardan, sin salir del asistente.

- Directorio de trabajo.
- Aplicación de ArcGIS a la cual va dirigido el *Add-In*.
- Nombre del *Add-In*.
- Versión.
- Compañia.
- Descripción.
- Autor.
- Imagen.

Una vez definidos los parámetros se crean los *Add-Ins*, comenzando con sus contenedores (menús, barras de herramientas y paleta de herramientas) para despues crear los propios *Add-Ins* (botones, herramientas y combos). En cada paso hay que completar los parámetros informativos asociados a cada elemento:

- Nombre.
- Descripción.
- Clase Python asociada.

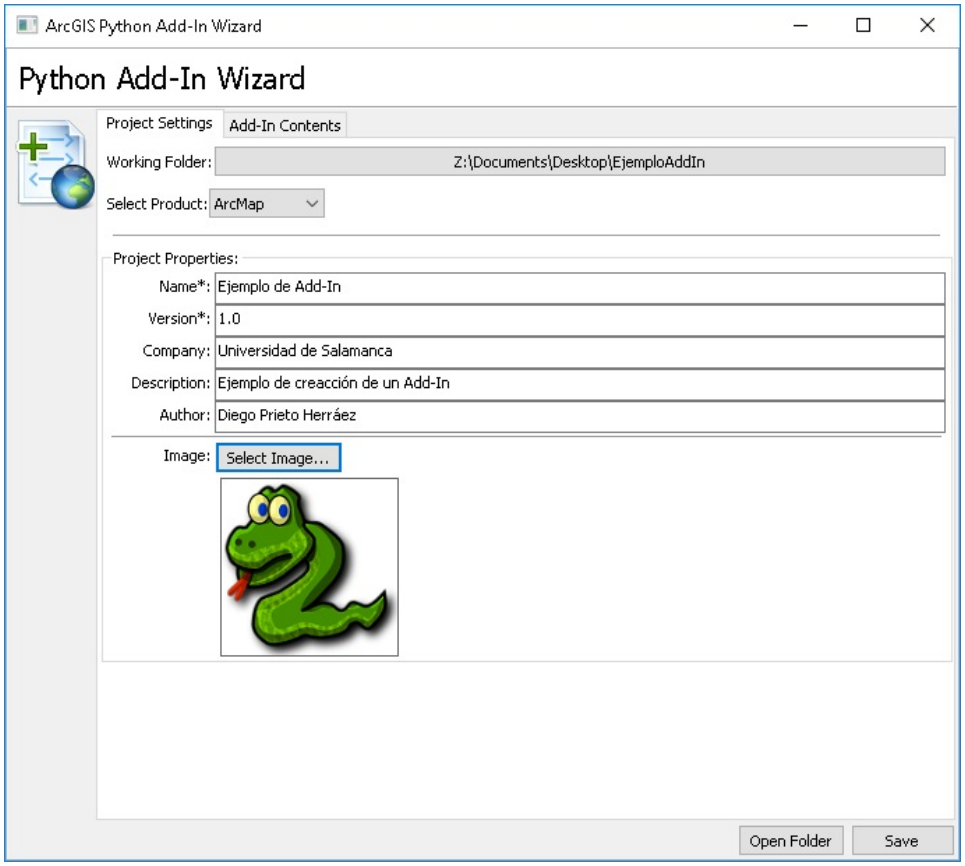

Figura 5: *ArcGIS Python Add-In Wizard*

- Titulo de ayuda.
- Texto de la ayuda.
- Imagen.

Cada *Add-In* tiene un script asociado desarrollado en el lenguaje interpretado Python, que se almacena en un fichero llamado por defecto `nombre_proyecto_addin.py`. Dentro de este fichero se deberá identificar la clase asociada a cada *Add-In* y modificarla para dar la funcionalidad deseada a cada evento del *Add-In*.

1.3.3. Instalación de un *Add-In*

Para finalizar, una vez añadida la lógica deseada al fichero Python, se ejecutará el fichero `makeaddin.py`, lo que creará el paquete comprimido `.esriaddin` que podremos distribuirlo e instalarlo en ArcGIS Desktop utilizando la *Esri ArcGIS Add-In Installation Utility*.

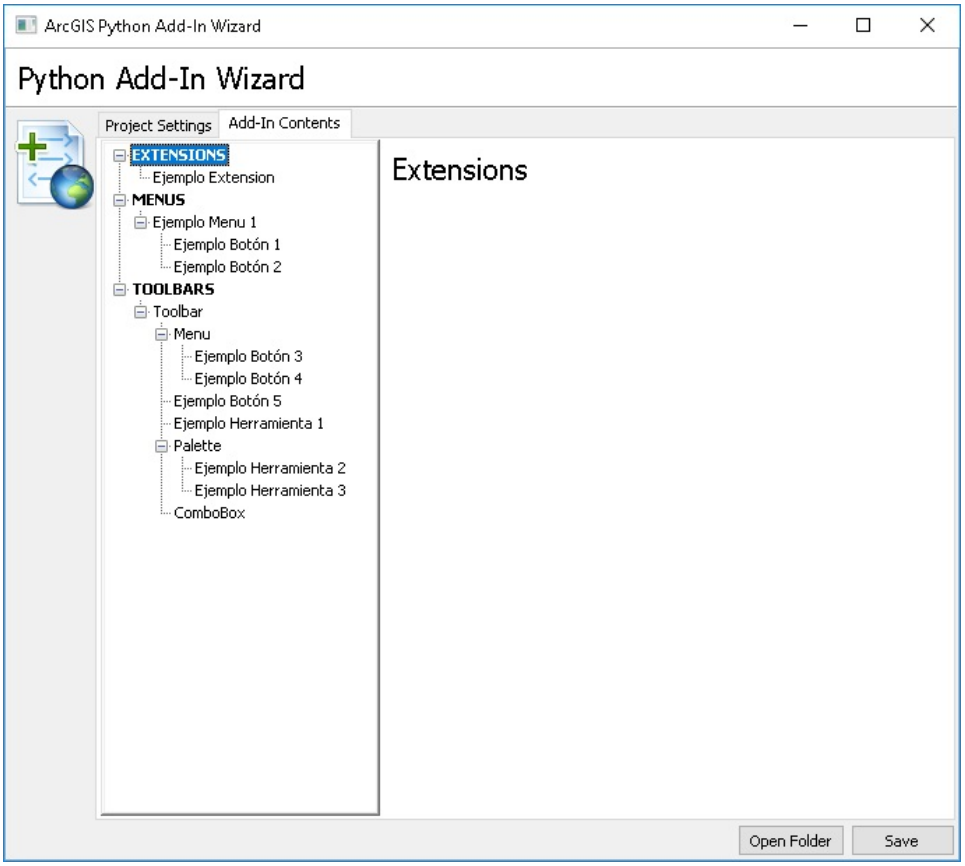

Figura 6: *ArcGIS Python Add-In Wizard*

A partir de aquí ya es posible abrir ArcGIS Desktop y probar nuestro *Add-In*. La barra de herramientas o el menu deben estar visibles y listas para probarlas. Si no lo están, pueden hacerse visibles en el menu *Customize* $\longrightarrow$ *Customize Mode...* $\longrightarrow$ *Commands* $\longrightarrow$ *Menu* o pinchando con el botón derecho del ratón sobre las barras de herramientas, y seleccionando los elementos correspondientes.

1.4. Lógica del *Add-In*

1.4.1. Python

Python es un lenguaje de programación interpretado cuyas principales carácterísticas son:

- Orientación a objetos.
- Multiplataforma.

```
import arcpy
import pythonaddins

class ButtonClass1(object):
    """Implementation for EjemploAddIn_addin.button1 (Button)"""
    def __init__(self):
        self.enabled = True
        self.checked = False
    def onClick(self):
        pass

class ToolClass1(object):
    """Implementation for EjemploAddIn_addin.tool1 (Tool)"""
    def __init__(self):
        self.enabled = True
        self.shape = "NONE" # Can set to "Line", "Circle" or "Rectangle" for
    def onMouseDown(self, x, y, button, shift):
        pythonaddins.MessageBox("Has cliqueado en el punto ("+ str(x)+","+ st
```

Figura 7: `nombreproyectoaddin.py`

- Código abierto.
- Gratuito.
- Fácil de entender, aprender y manejar.

Utilizando este lenguaje se elaboraran los scripts (ficheros con extensión `.py`) que nos permitirán el uso de variables y funciones, e incorporar bucles y condiciones, así como facilidad en la reutilización e intercambio de código.

1.4.2. Arcpy

ArcGIS integra una serie de librerías de análisis espacial que operan sobre Python denominadas ArcPy. Este paquete de librerías proporciona una manera rápida y sencilla de realizar:

- Análisis de datos geográficos.
- Manejo y conversión a diferentes formatos.
- Administración de información geográfica.
- Generación automatizada de mapas y series cartográficas.
- Acceso de lectura y escritura a datos alfanuméricos.

Dicho de otro modo, ArcPy permite decir a ArcGIS que debe hacer utilizando Python en lugar de interactuar con las toolboxes a traves de la interfaz gráfica de usuario.

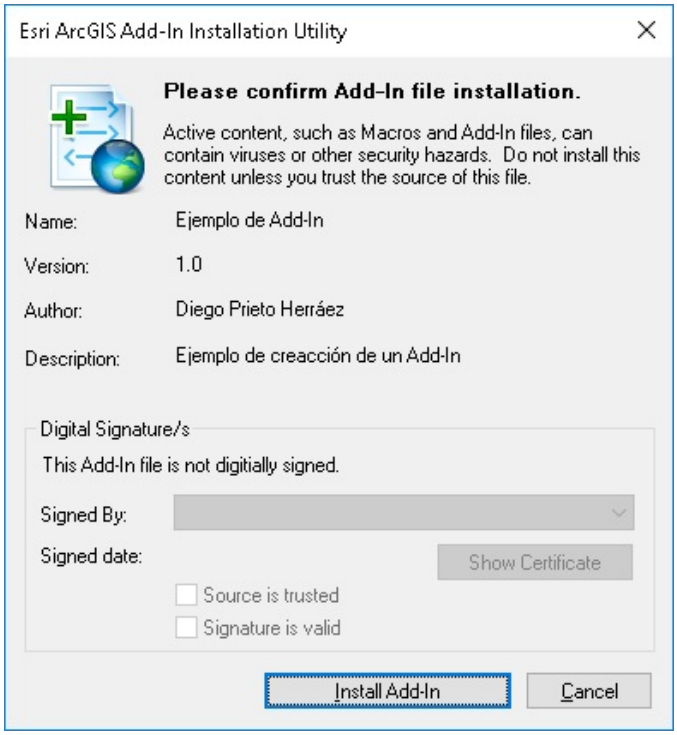

Figura 8: Instalación de un *Add-In*.

Para llevar a cabo estas tareas, ArcPy proporciona acceso a:

- Herramientas de Geoprocesado de ArcToolBox.
- Funciones.
- Clases: SpatialReference, Extent, Point, Polygon, etc.
- Módulos: arcpy.mapping, arcpy.sa, arcpy.da, y arcpy.time. Cada módulo posee tareas especializadas

introduciendo diferentes comandos en la ventana de Python.

Para hacer uso de esta librería, en las primeras líneas del programa deberemos importarla mediante la línea:

```
1 import arcpy
```

1.4.3. Pythonaddins

El módulo pythonaddins incluye funciones para dar soporte a los Add-Ins de Python. Entre estas funciones podemos encontrar:

- `OpenDialog()`: que permite abrir información.
- `SaveDialog()`: que permite guardar información.
- `GPToolDialog()`: que permite llamar a una toolbox.
- `GetSelectedTOCLayerOrDataFrame()`: que permite identificar la capa seleccionada en la tabla de contenidos.
- `GetSelectedCatalogWindowPath()`: que permite identificar la capa seleccionada en el catálogo.
- `ProgressDialog()`: que carga una ventana de progresión.

Para hacer uso de esta librería, en las primeras líneas del programa deberemos importarla mediante las líneas:

```
1 import arcpy
2 import pythonaddins
```

Códigos de Ejemplo

Archivo .XML

```
1  <ESRI.Configuration xmlns="http://schemas.esri.com/Desktop/AddIns" xmlns:xsi="http://www.w3.
2  <Name>Ejemplo de Add-In</Name>
3  <AddInID>{4c1d039c-66d2-4288-a8ef-40352ef2a6a0}</AddInID>
4  <Description>Ejemplo de creacci&#243;n de un Add-In</Description>
5  <Version>1.0</Version>
6  <Image>Images\Python_Icon_by_kaelan.png</Image>
7  <Author>Diego Prieto Herr&#225;ez</Author>
8  <Company>Universidad de Salamanca</Company>
9  <Date>01/17/2017</Date>
10 <Targets>
11   <Target name="Desktop" version="10.1" />
12 </Targets>
13 <AddIn language="PYTHON" library="EjemploAddIn_addin.py" namespace="EjemploAddIn_addin">
14   <ArcMap>
15     <Commands>
16       <Button caption="Ejemplo Bot&#243;n 3" category="Ejemplo de Add-In" class="ButtonClass
17       <Button caption="Ejemplo Bot&#243;n 4" category="Ejemplo de Add-In" class="ButtonClass
18       <Button caption="Ejemplo Bot&#243;n 5" category="Ejemplo de Add-In" class="ButtonClass
19       <Tool caption="Ejemplo Herramienta 1" category="Ejemplo de Add-In" class="ToolClass1"
20       <Tool caption="Ejemplo Herramienta 2" category="Ejemplo de Add-In" class="ToolClass2"
21       <Tool caption="Ejemplo Herramienta 3" category="Ejemplo de Add-In" class="ToolClass3"
```

```
22          <ToolPalette canTearOff="false" category="Ejemplo de Add-In" columns="2" id="EjemploAddIn_addin.
23            <Items>
24              <Tool refID="EjemploAddIn_addin.tool2" />
25              <Tool refID="EjemploAddIn_addin.tool3" />
26            </Items>
27          </ToolPalette>
28          <ComboBox caption="ComboBox" category="Ejemplo de Add-In" class="ComboBoxClass" id="EjemploAddIn
29          <Button caption="Ejemplo Bot&#243;n 1" category="Ejemplo de Add-In" class="ButtonClass1" id="Eje
30          <Button caption="Ejemplo Bot&#243;n 2" category="Ejemplo de Add-In" class="ButtonClass2" id="Eje
31        </Commands>
32        <Extensions>
33        </Extensions>
34        <Toolbars>
35          <Toolbar caption="Toolbar" category="Ejemplo de Add-In" id="EjemploAddIn_addin.toolbar" showInit
36            <Items>
37              <Menu refID="EjemploAddIn_addin.menu" />
38              <Button refID="EjemploAddIn_addin.button5" />
39              <Tool refID="EjemploAddIn_addin.tool1" />
40              <ToolPalette refID="EjemploAddIn_addin.toolpalette" />
41              <ComboBox refID="EjemploAddIn_addin.combobox" />
42            </Items>
43          </Toolbar>
44        </Toolbars>
45        <Menus>
46          <Menu caption="Menu" category="Ejemplo de Add-In" id="EjemploAddIn_addin.menu" isRootMenu="false
47            <Items>
48              <Button refID="EjemploAddIn_addin.button3" />
49              <Button refID="EjemploAddIn_addin.button4" />
50            </Items>
51          </Menu>
52          <Menu caption="Ejemplo Menu 1" category="Ejemplo de Add-In" id="EjemploAddIn_addin.menu1" isRoot
53            <Items>
54              <Button refID="EjemploAddIn_addin.button1" />
55              <Button refID="EjemploAddIn_addin.button2" />
56            </Items>
57          </Menu>
58        </Menus>
59      </ArcMap>
60    </AddIn>
61    </ESRI.Configuration>
```

Archivo .py

```
import arcpy
import pythonaddins
import os
import time

class ButtonClass1(object):
  """Implementation for EjemploAddIn_addin.button1 (Button)"""
  def __init__(self):
    self.enabled = True
    self.checked = False
  def onClick(self):
    arcpy.env.addOutputsToMap = False
    arcpy.env.overwriteOutput = True
    mxd = arcpy.mapping.MapDocument('CURRENT')
    mxd.author = "Diego Prieto Herraez"
    mxd.credits = "2017. SINUMCC. Universidad de Salamanca"
    mxd.title = "Prueba"
    dataframe = arcpy.mapping.ListDataFrames(mxd)[0]
    dataframe.name = "Prueba"
    dataframe.spatialReference = 25830
    for lyr in arcpy.mapping.ListLayers(mxd,"",dataframe):
    arcpy.mapping.RemoveLayer(dataframe,lyr)
    dataframe.extent = [-23768,3897187,1131858,4867188]
    ButtonClass2.enabled = False
    arcpy.RefreshActiveView()
    arcpy.RefreshTOC()
    del mxd, dataframe

class ButtonClass2(object):
  """Implementation for EjemploAddIn_addin.button2 (Button)"""
  def __init__(self):
    self.enabled = True
    self.checked = False
  def onClick(self):
    pythonaddins.SaveDialog("Guardar...")

class ButtonClass3(object):
  """Implementation for EjemploAddIn_addin.button3 (Button)"""
  def __init__(self):
    self.enabled = True
    self.checked = False
  def onClick(self):
```

```
43        pythonaddins.OpenDialog("Abrir...")
44
45    class ButtonClass4(object):
46      """Implementation for EjemploAddIn_addin.button4 (Button)"""
47      def __init__(self):
48        self.enabled = True
49        self.checked = False
50      def onClick(self):
51        pythonaddins.MessageBox("Texto del cuadro", "Cuatro uno!",0)
52        pythonaddins.MessageBox("Texto del cuadro", "Cuatro dos!",1)
53        pythonaddins.MessageBox("Texto del cuadro", "Cuatro tres!",2)
54        pythonaddins.MessageBox("Texto del cuadro", "Cuatro cuatro!",3)
55        pythonaddins.MessageBox("Texto del cuadro", "Cuatro cinco!",4)
56        pythonaddins.MessageBox("Texto del cuadro", "Cuatro seis!",5)
57        pythonaddins.MessageBox("Texto del cuadro", "Cuatro siete!",6)
58
59    class ButtonClass5(object):
60      """Implementation for EjemploAddIn_addin.button5 (Button)"""
61      def __init__(self):
62        self.enabled = True
63        self.checked = False
64      def onClick(self):
65        with pythonaddins.ProgressDialog as dialog:
66          dialog.title = "Progress Dialog"
67          dialog.description = "Copying a large feature class."
68          dialog.animation = "File"
69          for i in xrange(100):
70            dialog.progress = i
71            time.sleep(0.125)
72            if dialog.cancelled:
73              raise Exception("Ooops")
74
75    class ComboBoxClass(object):
76      """Implementation for EjemploAddIn_addin.combobox (ComboBox)"""
77      def __init__(self):
78        self.items = ["item1", "item2"]
79        self.editable = True
80        self.enabled = True
81        self.dropdownWidth = 'WWWWWW'
82        self.width = 'WWWWWW'
83      def onSelChange(self, selection):
84        pass
85      def onEditChange(self, text):
86        pass
```

```
87        def onFocus(self, focused):
88          pass
89        def onEnter(self):
90          pass
91        def refresh(self):
92          pass
93
94      class ToolClass1(object):
95        """Implementation for EjemploAddIn_addin.tool1 (Tool)"""
96        def __init__(self):
97          self.enabled = True
98          self.shape = "NONE" # Can set to "Line", "Circle" or "Rectangle" for interactive shape d
99        def onMouseDown(self, x, y, button, shift):
100         pythonaddins.MessageBox("Has cliqueado en el punto ("+ str(x)+","+ str(y)+").", "Ejemplo
101       def onMouseDownMap(self, x, y, button, shift):
102         pass
103       def onMouseUp(self, x, y, button, shift):
104         pass
105       def onMouseUpMap(self, x, y, button, shift):
106         pass
107       def onMouseMove(self, x, y, button, shift):
108         pass
109       def onMouseMoveMap(self, x, y, button, shift):
110         pass
111       def onDblClick(self):
112         pass
113       def onKeyDown(self, keycode, shift):
114         pass
115       def onKeyUp(self, keycode, shift):
116         pass
117       def deactivate(self):
118         pass
119       def onCircle(self, circle_geometry):
120         pass
121       def onLine(self, line_geometry):
122         pass
123       def onRectangle(self, rectangle_geometry):
124         pass
125
126     class ToolClass2(object):
127       """Implementation for EjemploAddIn_addin.tool2 (Tool)"""
128       def __init__(self):
129         self.enabled = True
130         self.shape = "NONE" # Can set to "Line", "Circle" or "Rectangle" for interactive shape d
```

```
131         def onMouseDown(self, x, y, button, shift):
132           pass
133         def onMouseDownMap(self, x, y, button, shift):
134           pass
135         def onMouseUp(self, x, y, button, shift):
136           pass
137         def onMouseUpMap(self, x, y, button, shift):
138           pass
139         def onMouseMove(self, x, y, button, shift):
140           pass
141         def onMouseMoveMap(self, x, y, button, shift):
142           pass
143         def onDblClick(self):
144           pass
145         def onKeyDown(self, keycode, shift):
146           pass
147         def onKeyUp(self, keycode, shift):
148           pass
149         def deactivate(self):
150           pass
151         def onCircle(self, circle_geometry):
152           pass
153         def onLine(self, line_geometry):
154           pass
155         def onRectangle(self, rectangle_geometry):
156           pass
157 
158       class ToolClass3(object):
159         """Implementation for EjemploAddIn_addin.tool3 (Tool)"""
160         def __init__(self):
161           self.enabled = True
162           self.shape = "NONE" # Can set to "Line", "Circle" or "Rectangle" for interactive shape drawing and
163         def onMouseDown(self, x, y, button, shift):
164           pass
165         def onMouseDownMap(self, x, y, button, shift):
166           pass
167         def onMouseUp(self, x, y, button, shift):
168           pass
169         def onMouseUpMap(self, x, y, button, shift):
170           pass
171         def onMouseMove(self, x, y, button, shift):
172           pass
173         def onMouseMoveMap(self, x, y, button, shift):
174           pass
```

```
175         def onDblClick(self):
176           pass
177         def onKeyDown(self, keycode, shift):
178           pass
179         def onKeyUp(self, keycode, shift):
180           pass
181         def deactivate(self):
182           pass
183         def onCircle(self, circle_geometry):
184           pass
185         def onLine(self, line_geometry):
186           pass
187         def onRectangle(self, rectangle_geometry):
188           pass
```