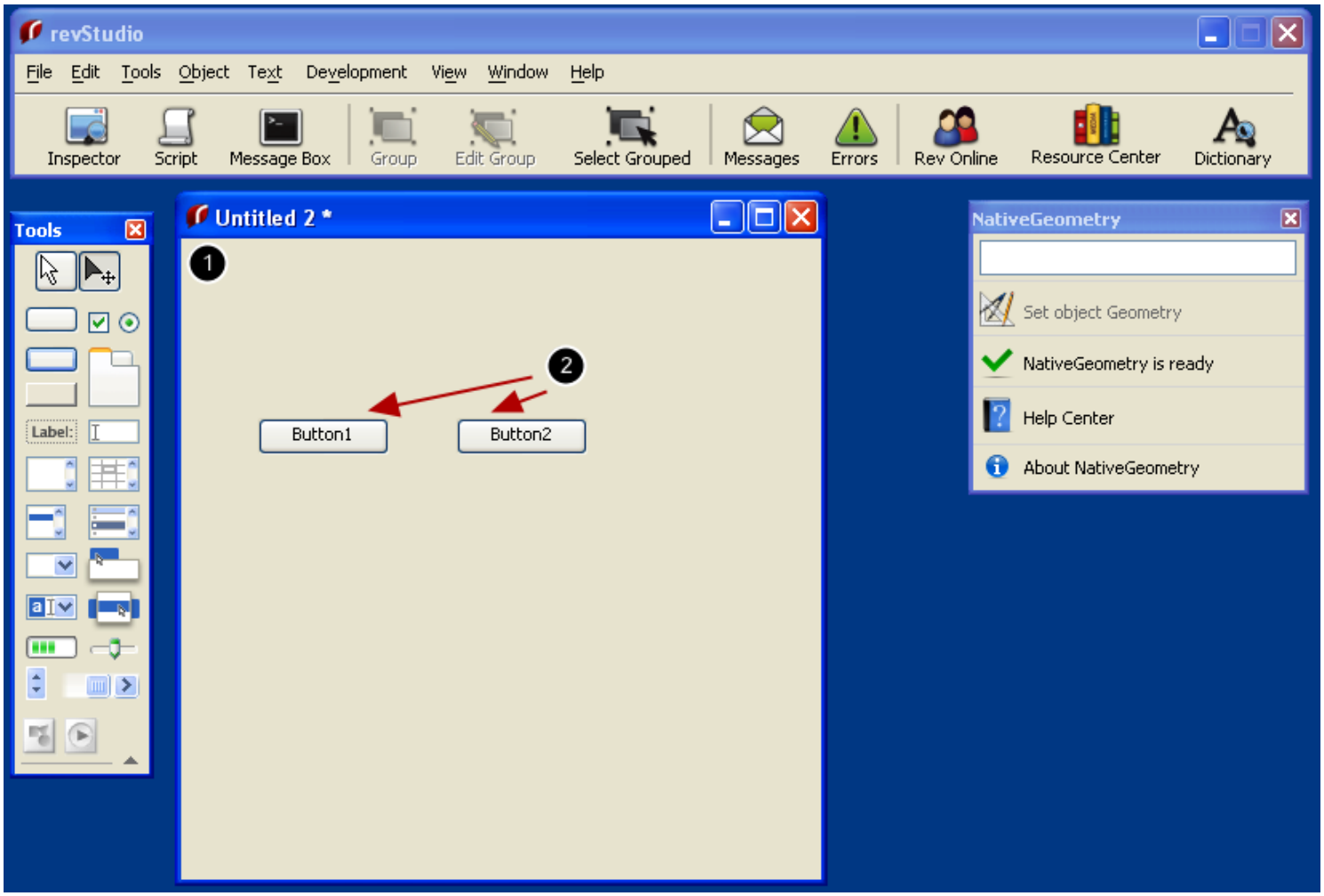
NativeSoft NativeGeometry Tutorial - 2 - The API

NativeGeometry is a complex technology, its core contains thousands of lines just to do one simple thing: The entire geometry management of your application. This tutorial will show you how to use the NativeGeometry API in order to speedily update the geometry of your application.

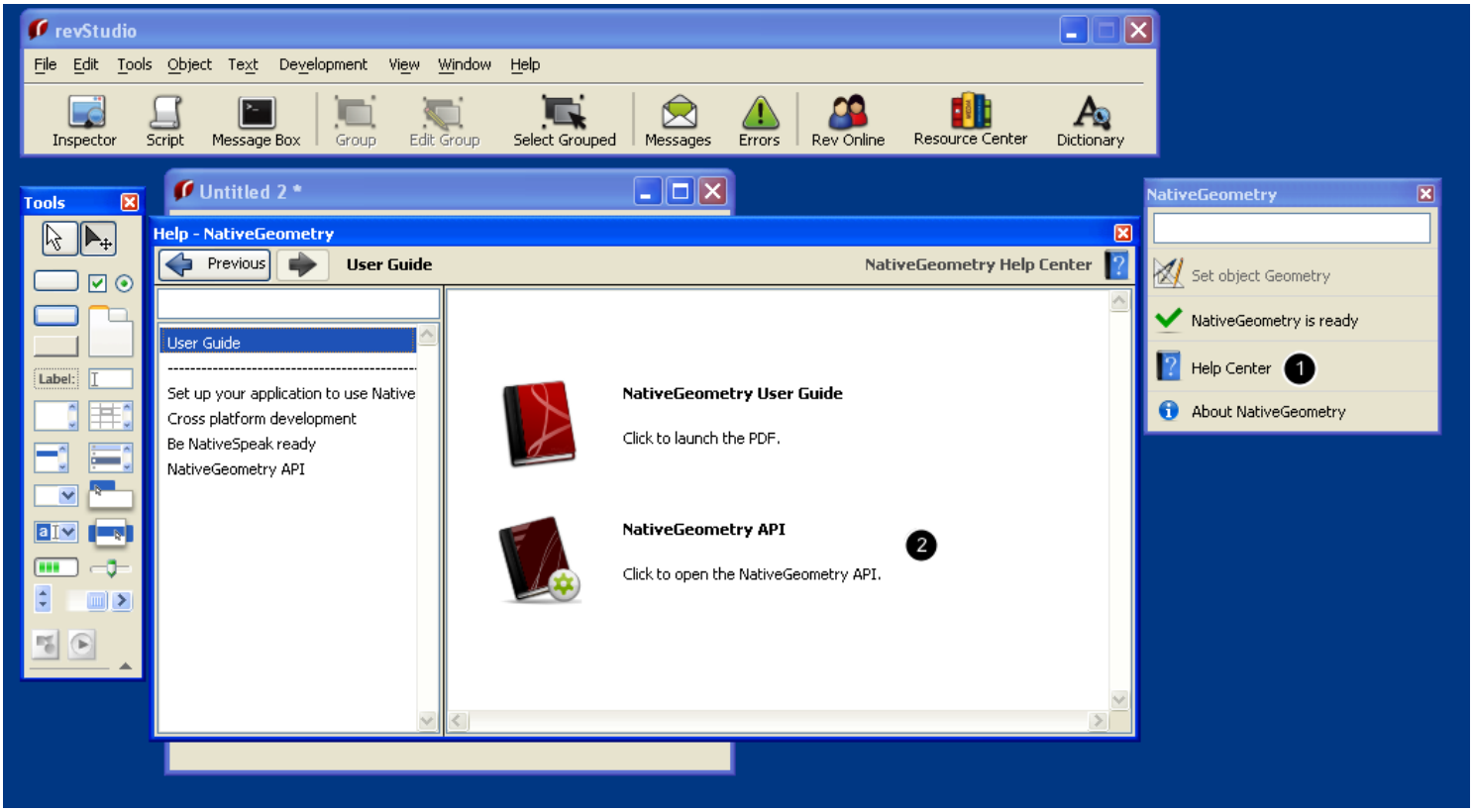
Before Anything



Before anything, your application must have NativeGeometry implemented and installed. Check out the NativeGeometry help center or the NativeGeometry first tutorial to see how to do this.

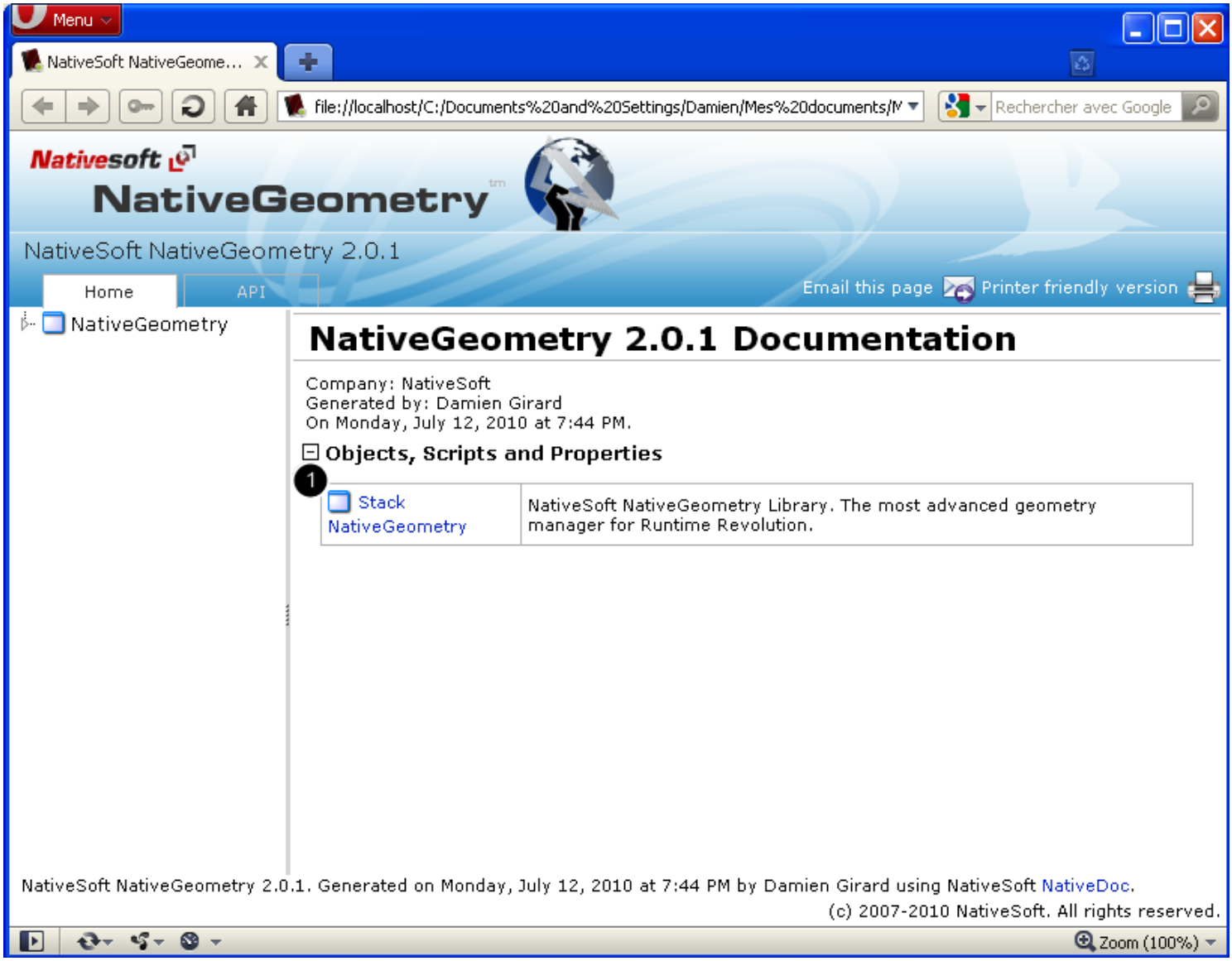
Create a new empty stack, with the required code to run NativeGeometry (1), then place two buttons on it. (2) Call them "Button1" and "Button2".

First Contact with the NativeGeometry API

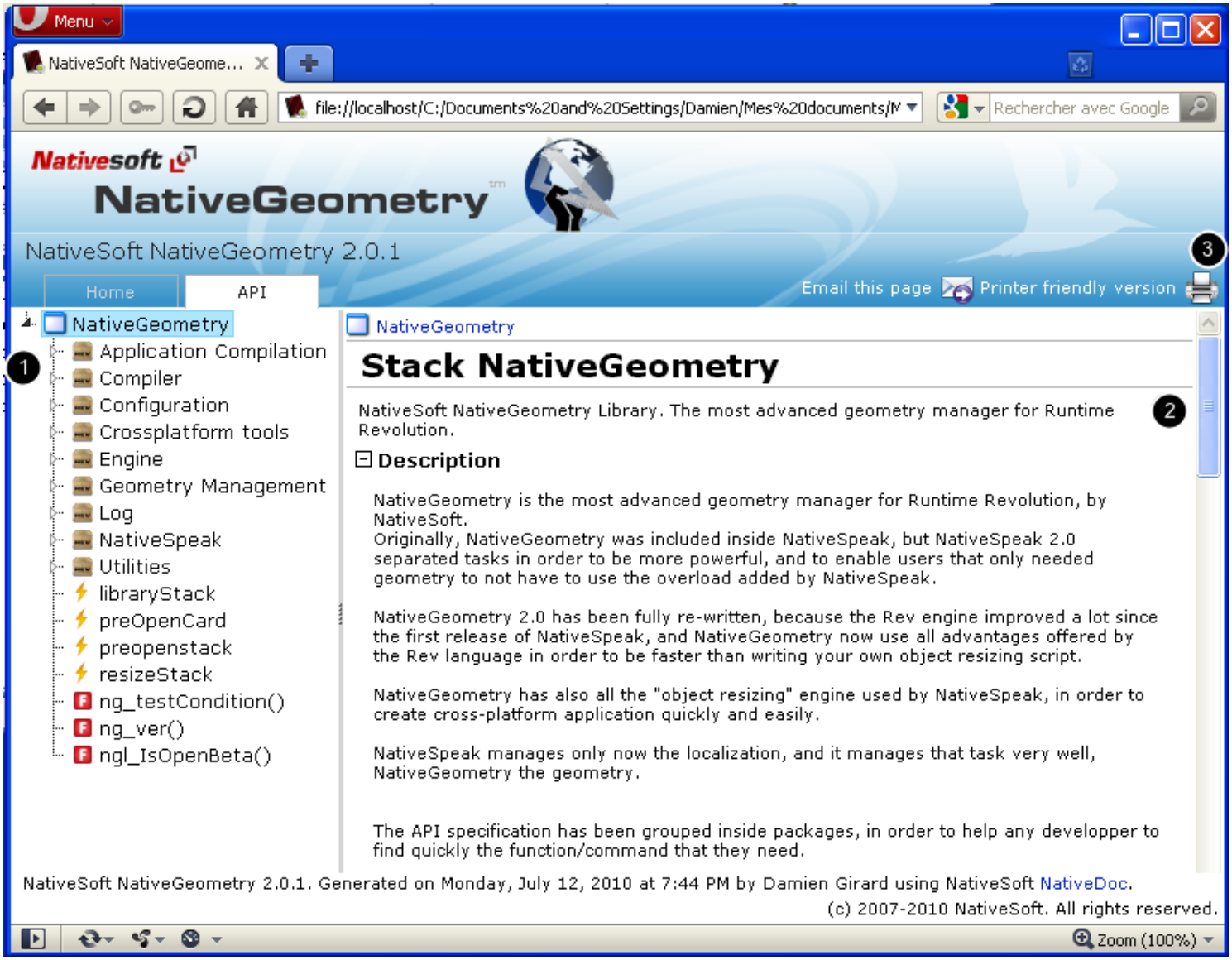


NativeSoft has developped another product, NativeDoc, this enables you to generate beautiful API documentation from Revolution source code. It uses the standard DoxyGen/JavaDoc notation. NativeGeometry, of course, has been developed with this tool.

So, to open the NativeGeometry API documentation, click "Help Center" (1) then "NativeGeometry API" (2).



Your favorite web browser will have opened the NativeGeometry API documentation. Click "Stack NativeGeometry" (1) to go directly to the NativeGeometry documentation.

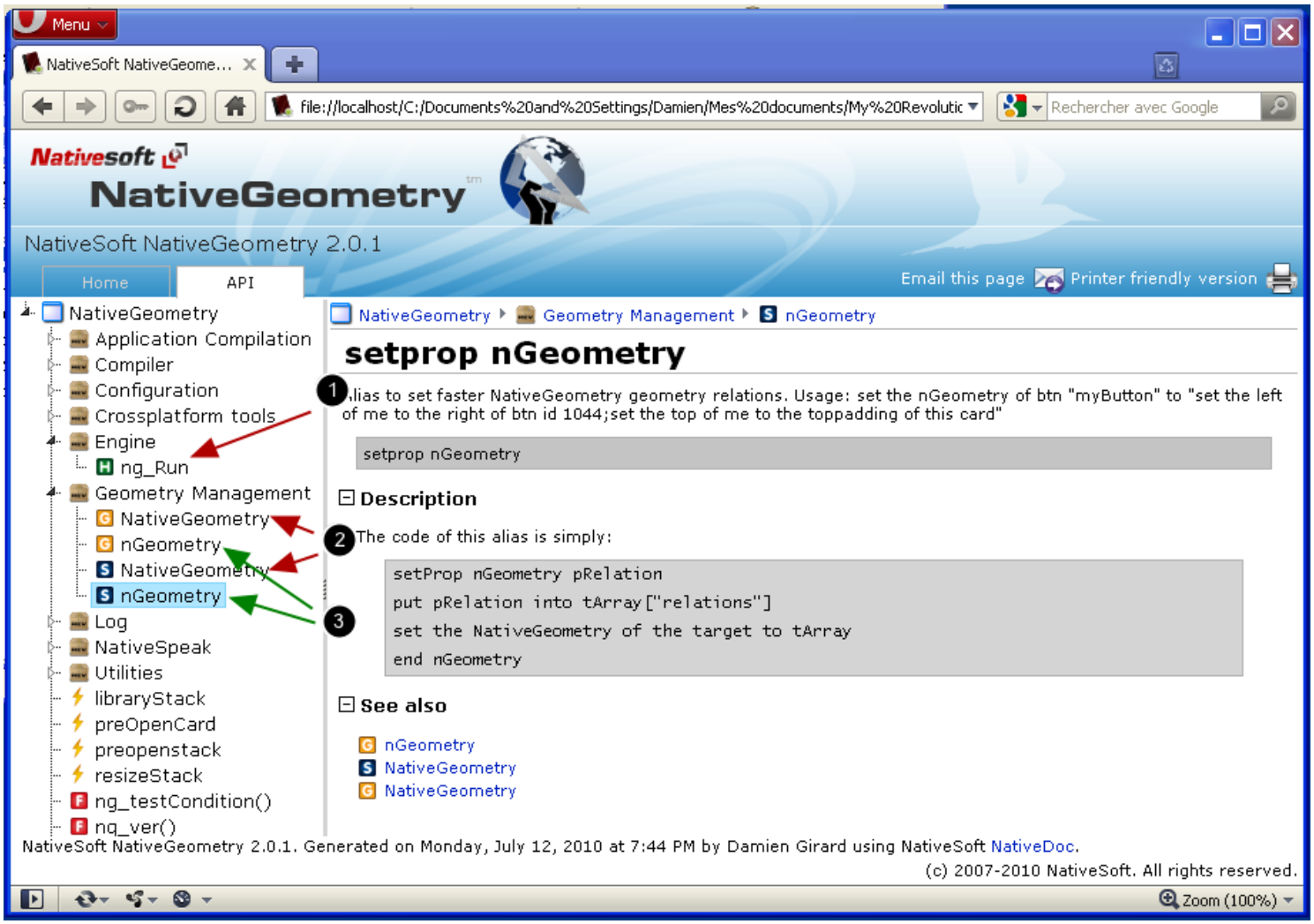


The NativeGeometry API documentation uses the standard NativeDoc website template, it is organised like this:

- The "tree" (1) enables you to quickly go to a function, as you see NativeGeometry commands/functions are organized in "Packages"
- The "content" (2) contains the content of the documentation.
- The "Printer friendly version" (3) enables you to easily print a page.

Before starting to explain the main commands/functions/properties of NativeGeometry, begin by reading the the introduction of the NativeGeometry API. (2)

The Main Properties/Command of NativeGeometry

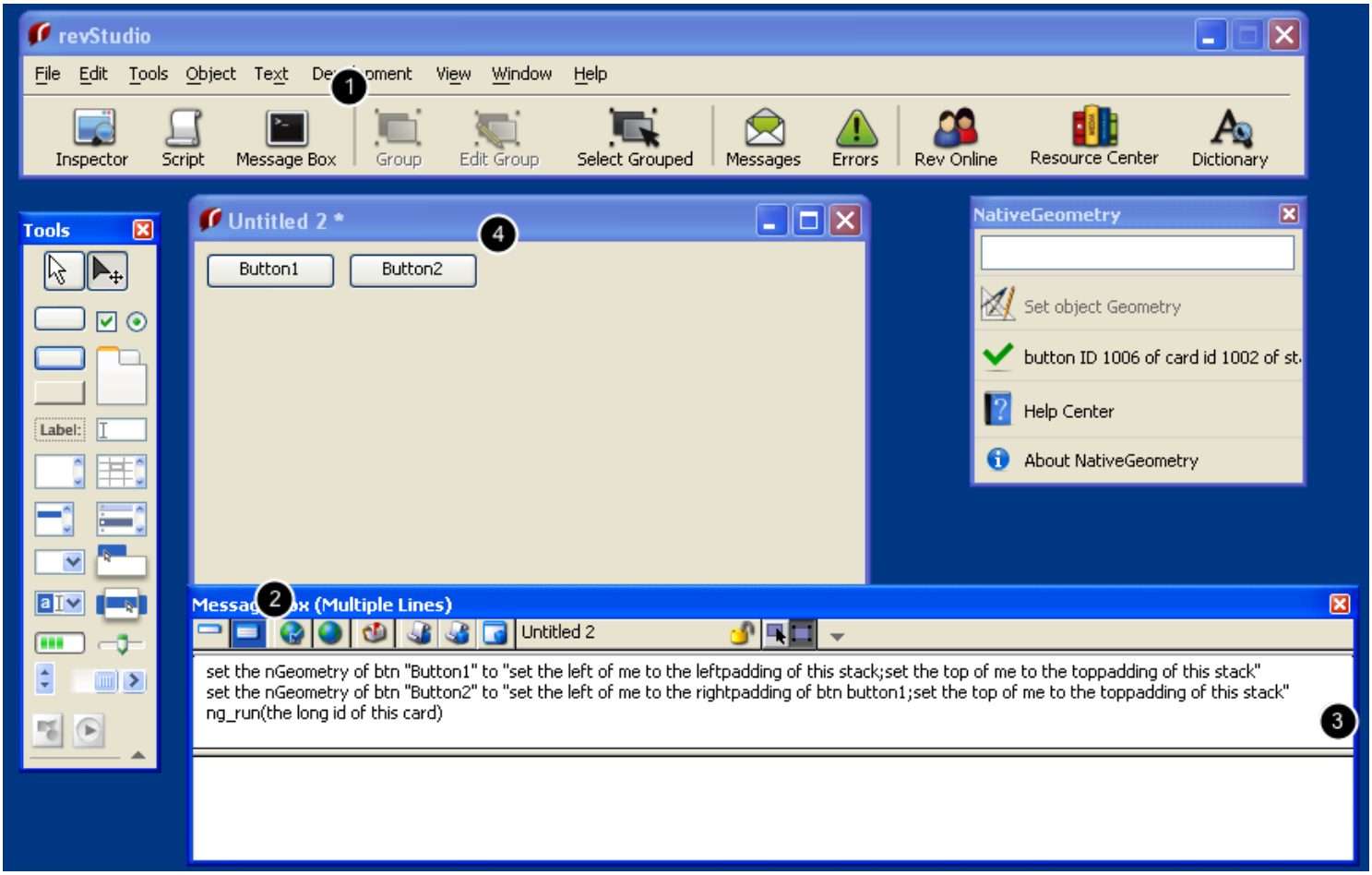


The NativeGeometry API is as simple as possible, it requires only 2 properties and 1 command to handle all the geometry operations of your application. We added 2 alias's in NativeGeometry 2.0.1 in order to make the use of the API easier.

So, we have:
- The command "ng_Run". -> This is the command that "launches" the NativeGeometry engine, and will resize everything.
- The get/set the NativeGeometry of ... (2) -> This controls the main properties of NativeGeometry, it enables you to set all NativeGeometry properties of any object!
- The get/set the nGeometry of ... (3) -> This is an alias of the NativeGeometry properties, it enables you to set faster geometry relations.

We recommend you read the documentation of these properties/command next. You don't need to fully understand everything in the API documentation, but you will understand this tutorial better if you read the documentation first.

First Operation with the NativeGeometry API - the nGeometry property

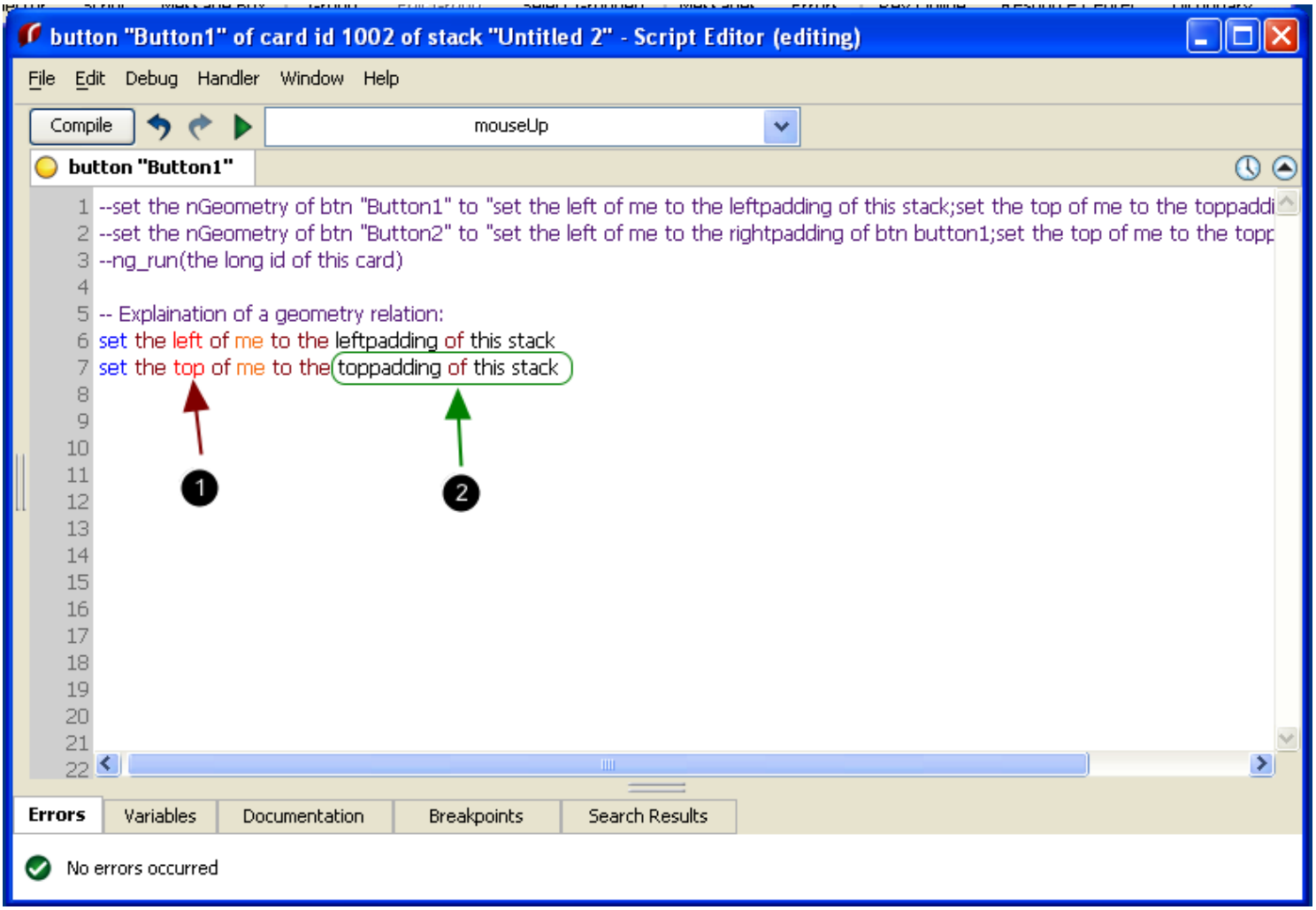


So, now we will set up your first operation with the NativeGeometry API. We will do the same things that we have done in the first tutorial, but simply without the GUI here.

So, we want to have the "Button1" to be placed on the top left, and the "Button2" to be at the right of the button 1. To do that:
- Open the message box (1)
- Click "multiline message box" (2)
- Place the following code inside it: (3)

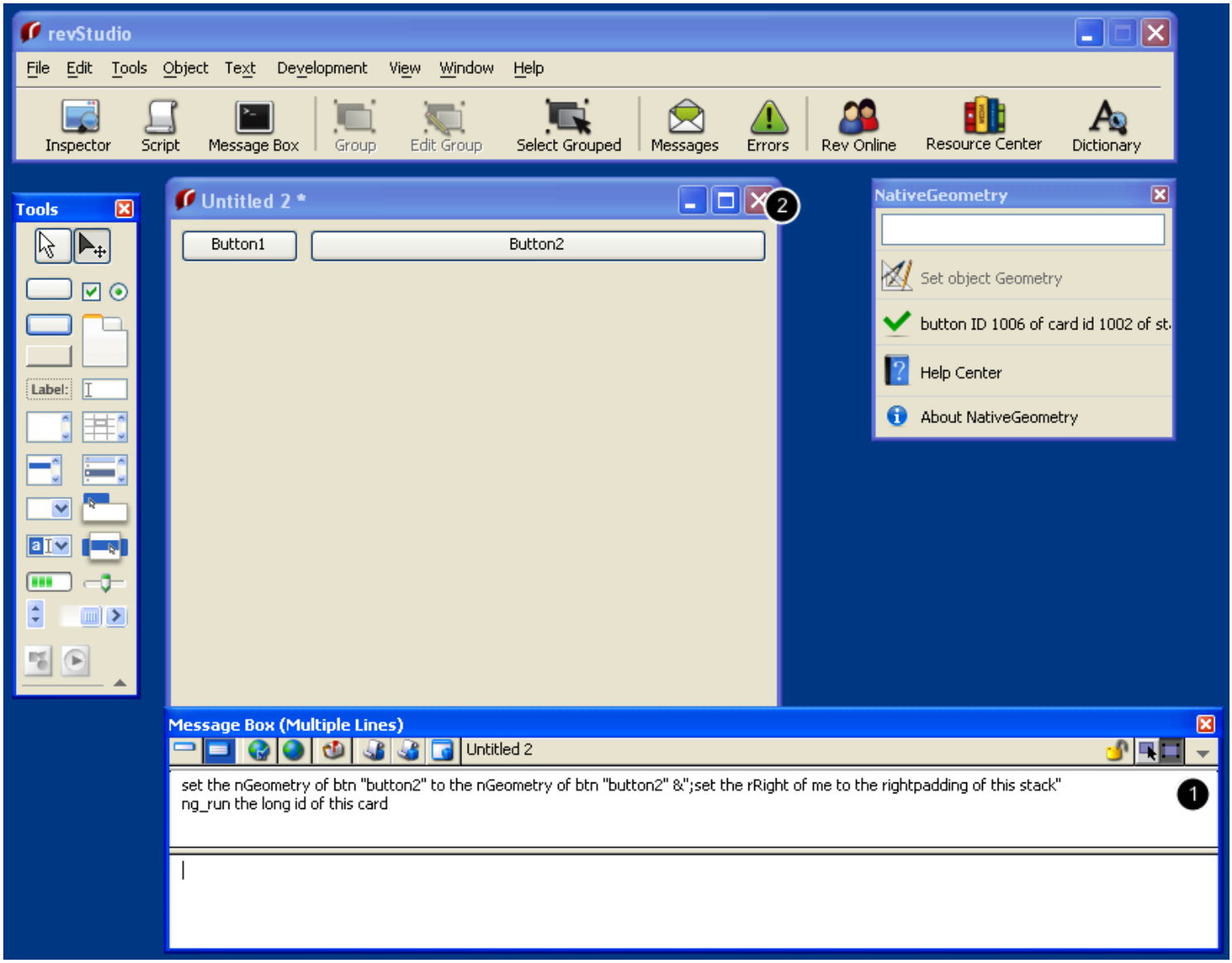
set the nGeometry of btn "Button1" to "set the left of me to the leftpadding of this stack;set the top of me to the toppadding of this stack"
set the nGeometry of btn "Button2" to "set the left of me to the rightpadding of btn button1;set the top of me to the toppadding of this stack"
ng_run(the long id of this card)
- Press enter, and see NativeGeometry working :) (4)

The Geometry Relation Format



The geometry relation format has been designed to be as natural and painless as possible. It looks like a classic Rev script.

A NativeGeometry relations will always have this construction:
- set the <property> (1) of me to <value> (2)



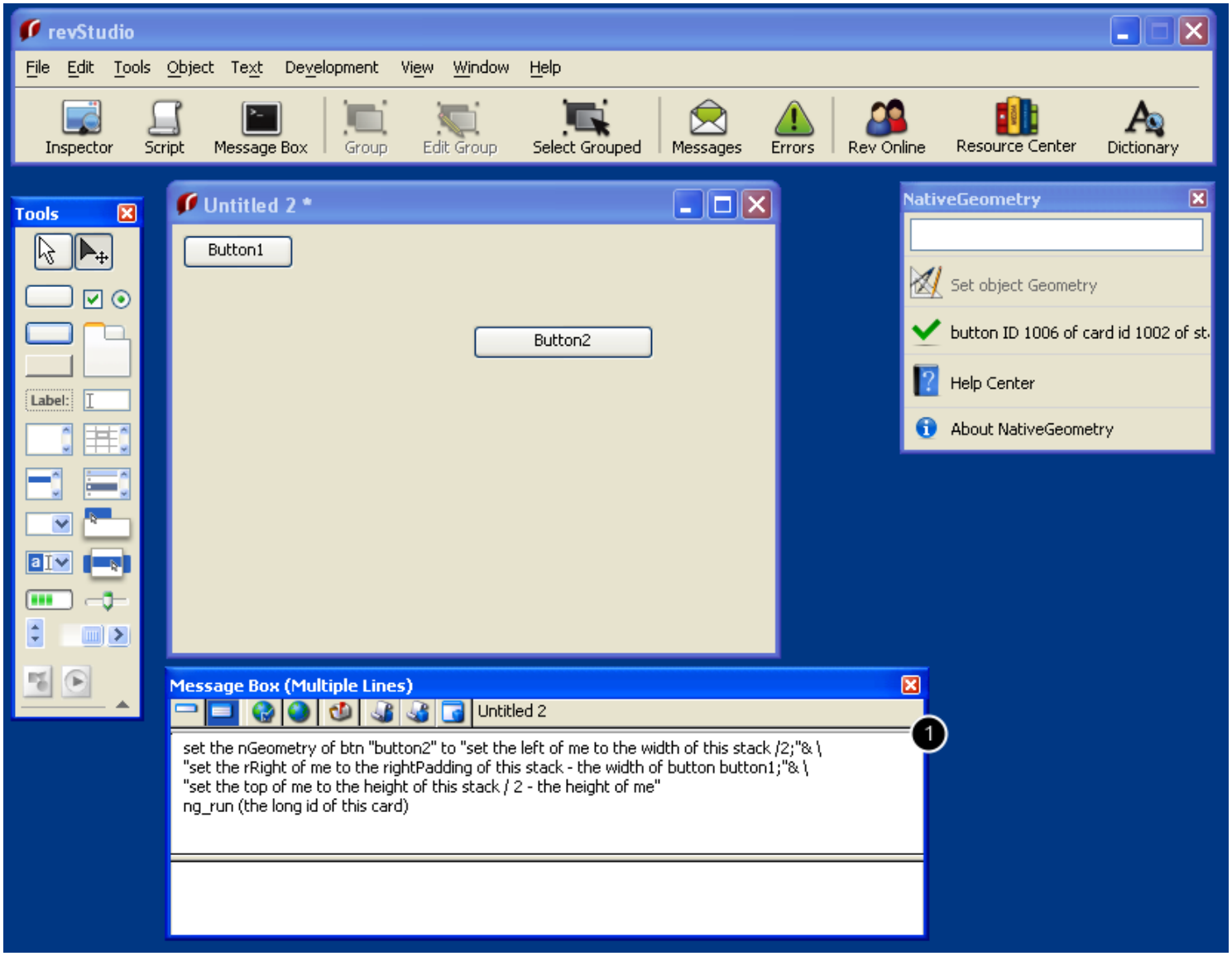
<Property> can be a classical property:
- left
- top
- right
- bottom

Or an extended one:
- rLeft
- rTop
- rRight
- rBottom

The "r" before the "left|top|right|bottom" means that it will be a "resize" operation instead of a "move". If you set the rRight of an object, its left will not be modified but its right will be placed where you want to have it placed. The width of the object is updated.

To try, we will simply add to the button "button2" the relation "set the rRight of me to the rightPadding of this stack". Place the following code inside the message box (1):
set the nGeometry of btn "button2" to the nGeometry of btn "button2" &";set the rRight of me to the rightpadding of this stack"
ng_run the long id of this card

Now you have the button "Button2" that has its right to the rightpadding of the stack.



<Value> can be a fixed value (eg. "15") or a computed one (eg. "the width of this stack").

In the value, you can use the keyword "me", "this stack". A serious advantage compared to writing scripts inside the resizeStack handler!

To test, simply paste the following code in the message box:
*set the nGeometry of btn "button2" to "set the left of me to the width of this stack /2;"& *

*"set the rRight of me to the rightPadding of this stack - the width of button button1;"& *
"set the top of me to the height of this stack / 2 - the height of me"
ng_run (the long id of this card)

Then launch it and you should see that the object has the desired position.

Note: If you read the documentation of the "NativeGeometry" property, you will have seen that NativeGeometry handles conditions too (eg. "if the hilite of btn..."). We will look at this feature in another tutorial.

The Automatic Geometry Dependency Solver

NativeGeometry is so simple to use because it has an "Automatic geometry dependency solver".

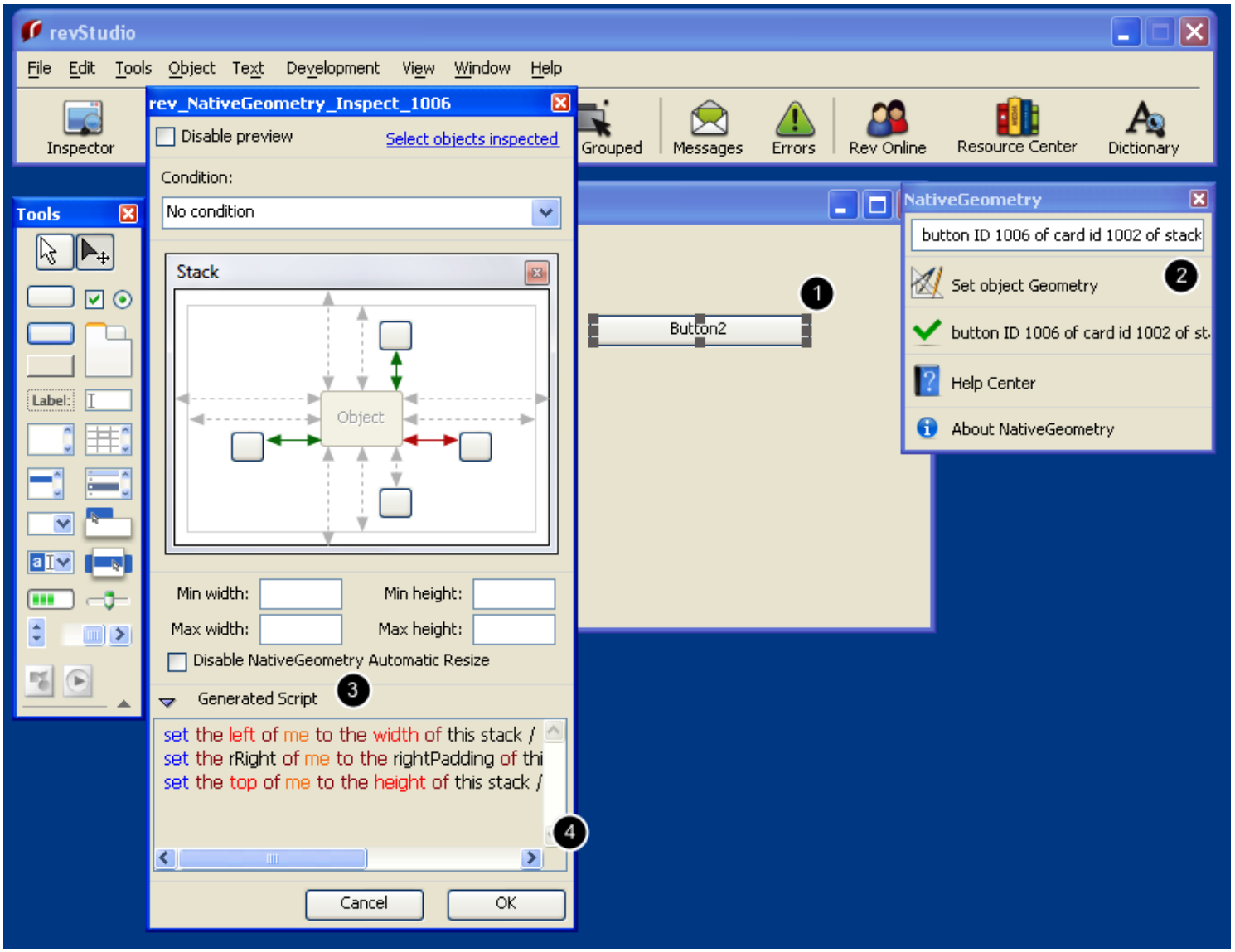
Some explanation of this feature:
- The NativeGeometry geometry relation compiler parses geometry relations
- A dependency tree then is made by the "automatic geometry dependency solver"
- Objects are resized in the order defined by the tree

Sounds easy, but this major feature means you don't have to worry about the order in which objects have to be resized. In our example it is easy, the button "button1" has to be resized first, then the button "button2". But when you have hundred of objects this becomes very much more difficult!

The most important element of this feature is that it supports conditional geometry relations. You will see all the power of the "automatic geometry dependency solver" in the "conditional geometry" tutorial.

So in summary - don't worry about the order you resize your objects in. Let NativeGeometry handle that.

The NativeGeometry inspector and the API



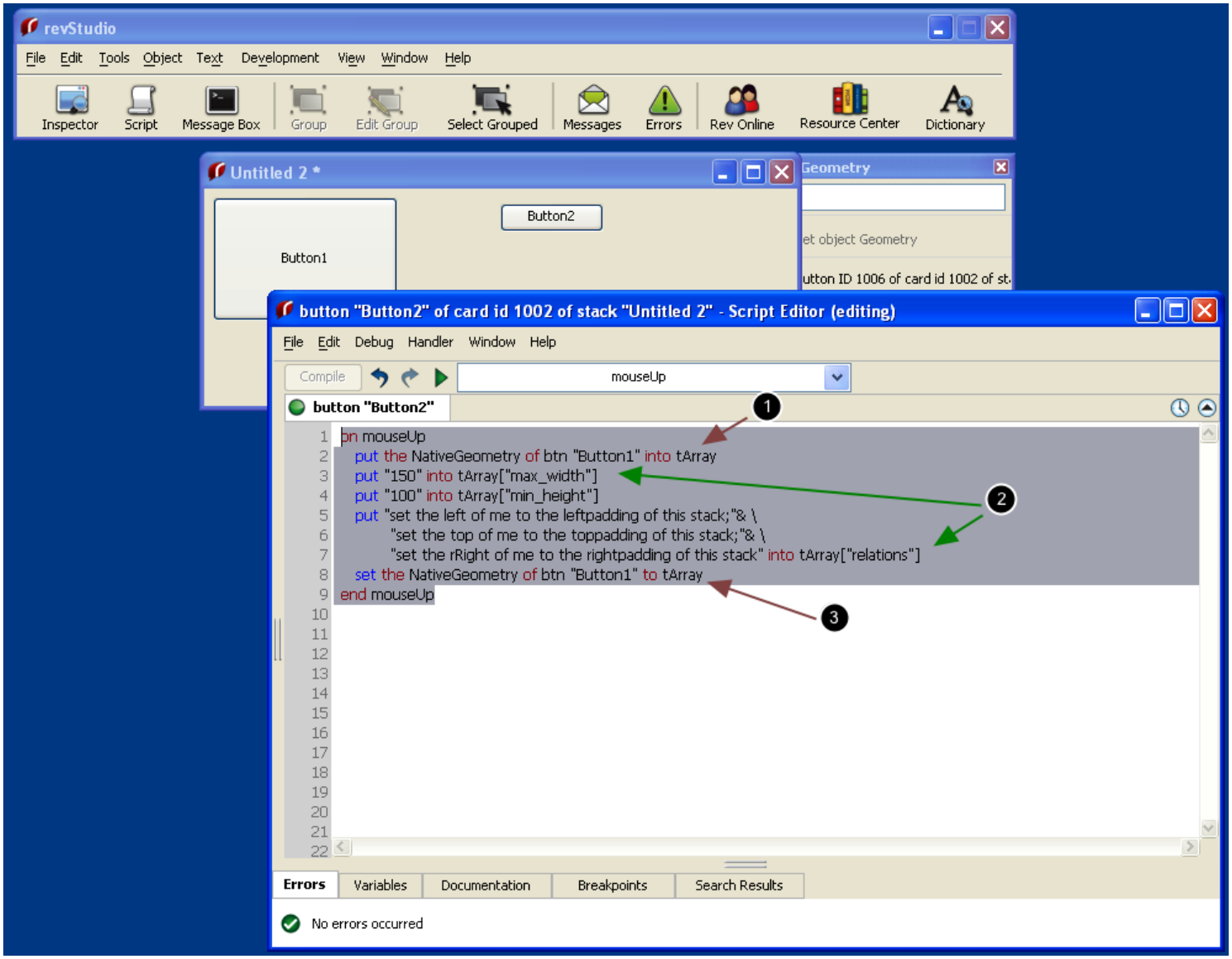
The NativeGeometry inspector is simply a tool that generates and parses geometry relations, it does not do anything else.

Select the button "Button2" (1), then click "Set object Geometry" (2), click then on "Generated Script" (3) and you will see the code that the inspector has read and will apply (4).

So, if you want to place in your sourcecode a geometry relation and you want to generate it fast, simply use the NativeGeometry geometry inspector.

Note: The NativeGeometry geometry inspector handles all features of the NativeGeometry API, it will parses even the most strange geometry relations (and it support conditions too !).

Set/Get the NativeGeometry



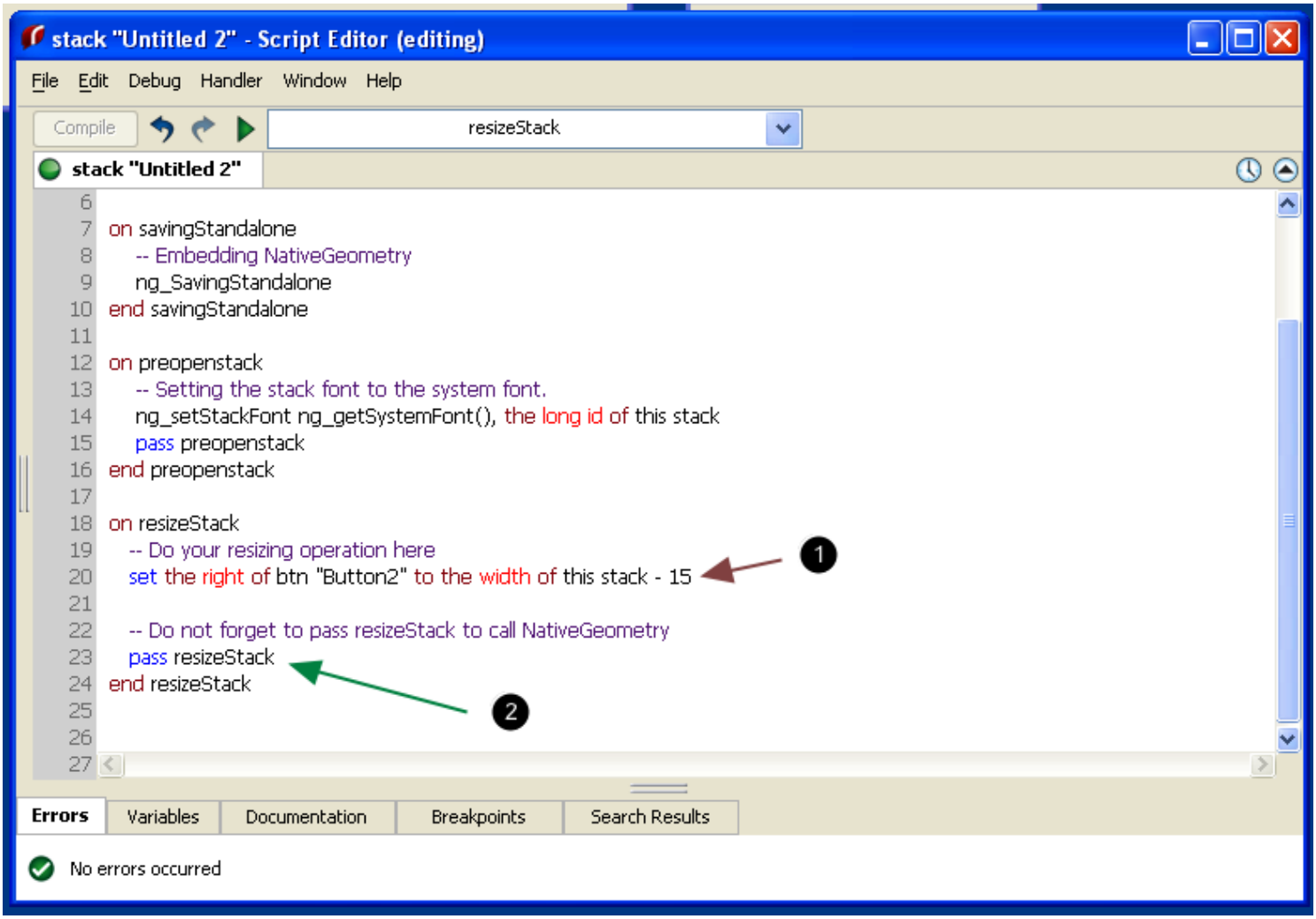
This commands enables you to fully manage the "NativeGeometry" of an object, this includes "min|max" "width|height", and the auto-resize property.

The usage of this property is a bit harder than "set the nGeometry", which is why in NativeGeometry 2.0.1 the property "nGeometry" was added.

Basically, to use the NativeGeometry property:
- You read the NativeGeometry array with "put the NativeGeometry" (1)
- Then you modify the content of the array. (2)
- Finally you set the NativeGeometry of the object with the array. (3)

The content of the array is explained in the "NativeGeometry" properties documentation.

Setting Your Geometry Manually Before Calling NativeGeometry



You can still continue to use your resizestacks handlers, the only thing you need to do if you want to use NativeGeometry is to "pass" it.

So, do not forget to "pass resizeStack" in order to let NativeGeometry work.

This is the end of the second NativeGeometry tutorial, it showed you how to use the NativeGeometry API in order to make excellent GUI's.

The next tutorial will be focused on the conditionnal geometry relations.

Kind Regards,

Damien Girard
http://www.nativesoft.fr