# The role of the activation factory in the Windows Runtime

devblogs.microsoft.com/oldnewthing/20240822-00

August 22, 2024


Raymond Chen

The Windows Runtime has these things called activation factories, which you obtain by calling `RoGetActivationFactory`. What is an activation factory?

The primary purpose of an activation factory is given in its name: To activate (create) objects. Every activation factory implements the `IActivationFactory` interface, which has a single method: `ActivateInstance`. This method creates an object and returns it.

Now, the `IActivationFactory::ActivateInstance` method does not take any input parameters, so this can be used only if your object has a default constructor (no parameters). If your class has constructors which take parameters, then you'll need more.

Non-default constructors for a class are placed on a custom interface conventionally named `IWidgetFactory`. For example, if you had a runtime class which had a constructor that took a string parameter:

```
runtimeclass Widget
{
    Widget(String name);
}
```

then the `IWidgetFactory` interface would have a method like

```
HRESULT IWidgetFactory::CreateInstance([in] HSTRING name, [out, retval] IWidget**
result);
```

The parameters to the constructor are the parameters to the `CreateInstance` method, and the output of the `CreateInstance` method is the newly-created object.

The other thing that is provided by the activation factory is the class's static members. The static members are on an interface conventionally named `IWidgetStatics`. For example, if we had a static method `FindByName`:

```
runtimeclass Widget
{
    static Widget FindByName(String name);
}
```

Then the `IWidgetStatics` interface would have a method like

```
HRESULT IWidgetStatics::FindByName([in] HSTRING name, [out, retval] IWidget**
result);
```

In summary, the activation factory is a place to put all the things that a class can do which aren't instance members. It's the object that represents the class itself, rather than any instances of it.

**Bonus chatter**: If you think of a constructor as a "static method called `CreateInstance` that returns a newly-constructed object", then you can think of the activation factory as the place to put all the static members.