Design question: How can I access an IPC mechanism transparently?

Asked 16 years, 2 months ago Modified 13 years, 1 month ago

Viewed 597 times

0

I want to do this (no particular language):

```
print(foo.objects.bookdb.books[12].title);
```

or this:

```
book = foo.objects.bookdb.book.new();
book.title = 'RPC for Dummies';
book.save();
```

Where foo actually is a service connected to my program via some IPC, and to access its methods and objects, some layer actually sends and receives messages over the network.

Now, I'm not really looking for an IPC mechanism, as there are plenty to choose from. It's likely not to be XML based, but rather s. th. like Google's protocol buffers, dbus or CORBA. What I'm unsure about is how to structure the application so I can access the IPC just like I would any object.

In other words, how can I have OOP that maps transparently over process boundaries?

Not that this is a design question and I'm still working at a pretty high level of the overall architecture. So I'm pretty agnostic yet about which language this is going to be in. C#, Java and Python are all likely to get used, though.

language-agnostic design-patterns ipc soa rpc

Share

Improve this question

Follow

edited Oct 13, 2008 at 11:52

asked Oct 13, 2008 at 11:38

Hanno Fietz

31.4k ● 49 ● 155 ● 238

3 Answers

Sorted by: Highest score (default)

2

I think the way to do what you are requesting is to have all object communication regarded as message passing. This is how object methods are handled in ruby and smalltalk, among others.

With message passing (rather than method calling) as your object communication mechanism, then operations such as calling a method that didn't exist when you wrote the code becomes sensible as the object can do

something sensible with the message anyway (check for a remote procedure, return a value for a field with the same name from a database, etc, or throw a 'method not found' exception, or anything else you could think of).

It's important to note that for languages that don't use this as a default mechanism, you can do message passing anyway (every object has a 'handleMessage' method) but you won't get the syntax niceties, and you won't be able to get IDE help without some extra effort on your part to get the IDE to parse your handleMessage method to check for valid inputs.

Share Improve this answer

Follow

answered Oct 13, 2008 at 11:44

workmad3

25.6k ● 4 ● 37 ● 56

0

Read up on Java's RMI -- the introductory material shows how you can have a local definition of a remote object.

The trick is to have two classes with identical method signatures. The local version of the class is a facade over some network protocol. The remote version receives requests over the network and does the actual work of the object.

You can define a pair of classes so a client can have

```
foo= NonLocalFoo( "http://host:port" )
foo.this= "that"
foo.save()
```

And the server receives set_this() and save() method requests from a client connection. The server side is (generally) non-trivial because you have a bunch of discovery and instance management issues.

Share Improve this answer

Follow

answered Oct 17, 2008 at 2:15

S.Lott

391k ● 82 ● 517 ● 788

-1

You shouldn't do it! It is very important for programmers to see and feel the difference between an IPC/RPC and a local method call in the code. If you make it so, that they don't have to think about it, they won't think about it, and that will lead to very poorly performing code.

Think of:

```
foreach o, o.isGreen in someList {
   o.makeBlue;
}
```

The programmer assumes that the loops takes a few nanoseconds to complete, instead it takes close to a second if someList happens to be remote.

Share Improve this answer

Follow

edited Nov 1, 2011 at 15:27

Jason Plank

2,336 ● 5 ● 32 ● 40

answered Oct 13, 2008 at 12:22

edgar.holleis
4,991 ● 2 ● 25 ● 27

1 Yeah, that's a good point. I was already thinking of that, too. However, this is easy to fix by naming conventions. If it's always like someIpcList or ipcFoo.bar(), you know what's happening. I'm not designing for an arbitrary language / framework but for a specific application and a specific team.
– Hanno Fietz Oct 13, 2008 at 13:59

Imho, the programmer should program against an interface, and *not* be aware of what lies behind it. It's the one that glues the program components together should think about this.
– xtofl Jan 24, 2012 at 11:31