

Architecture–switch to Web Services; newbie needs advice

Source:

<http://www.tech–archive.net/Archive/DotNet/microsoft.public.dotnet.framework.webservices/2005–03/0144.html>

From: Ney André de Mello Zunino (*zunino_at_inf.ufsc.br*)

Date: 03/04/05

Date: Fri, 04 Mar 2005 13:10:06 -0300

Hello.

Firstly, a bit of background information: I am involved in the development of a relatively small distributed application which, in short, deals with the remote editing and maintenance of some conceptual graphs (the very domain being irrelevant). For many reasons, including personal preference and motivation, I had set two main constraints for the choice of technologies to be employed: they ought to be both **free** and **portable**. This led me to the gathering of the following tool set:

- * C++ as the programming language for all tiers
- * Ice [1] as the communication framework
- * MySQL [2] as the database backend
- * MySQL++ [3] as the API to the database backend
- * Gtkmm [4] as the UI toolkit for the clients

After some struggle, I managed to get the basic architecture in place so that I was able to have my clients and middle–tier talk and exchange information. From that point on, I thought I was done. Unfortunately, I have faced countless difficulties, many of which aggravated by the lack of proper documentation and other resources, inherent of open source initiatives. I don't mean to sound ungrateful. I fully understand the fact that those people are out there putting in some serious efforts and mostly getting nothing in return (financially, at least). The thing is that sometimes that model does not go well with real–life business models and deadlines.

So, here I am, under a drive to go for an alternative solution. I spent some time looking around yesterday and doing some thinking and here is a basic summary of the new architecture I envisaged:

- * Build the middle–tier as a Web Service
- * Build the client application with C# and Windows Forms on Visual Studio .NET 2003

I am aware that I am giving up some virtues with these new choices. The goal is to be able to rely on better–supported tools in order to complete the development timely. On the other hand, I have several questions I would like to have clarified:

1) At the moment, there is a real constraint with regards to the platform from where the Web Service will run; it has to be on a Linux system (there are only Linux servers available).

1.a) What does one need in order to turn a regular HTTP server into a Web Services–capable provider? In other words, what makes a server a Web Services provider? As far as my understanding goes (I am a complete Web Services newbie, having never implemented anything related), a server need only be able to parse SOAP messages to process the encoded request and to encode responses as SOAP messages to respond to clients. So, supposing I have a regular Linux server running Apache HTTPd, what do I need to install in order to be able to host Web Services?

1.b) In spite of the initial constraint that the Web Service must be hosted on a Linux server, I would like to know whether it is common and whether it would be easy to have the same service hosted on a different platform, e.g. a Windows server. Because one of the most advertised virtues of Web Services is their interoperability capabilities, I would assume that porting them should not be very hard as well. However, we know for a fact that practice doesn't always reflect theory.

2) How would the C# client be affected if the Web Service were moved from the initial Linux server to a different platform? I expect that the client would be (mostly) insulated from that kind of change.

3) How difficult is it and how long would it take (roughly) for a Web Services newbie to grasp the basic concepts necessary to realize a basic architecture like this?

4) Could you indicate some must–read elementary material and also some hands–on tutorial on how to build a C# application which consumes Web Services that were not created with .NET (if that matters at all)?

I apologize if I couldn't be clearer in exposing my concerns. I would be very glad if this thread could serve to clear up my almost inexistent understanding of Web Services and the ever–growing list of hype terminology which frightens me. All in all, I prefer to believe the thing must not be as scary as it seems. I just need to find a proper introduction.

[1] <http://zeroc.com/>

[2] <http://dev.mysql.com/>

[3] <http://tangentsoft.net/mysql++/>

[4] <http://gtkmm.org/>

Thank you,

microsoft.public.dotnet.framework.webservices: Architecture–switch to Web Services; newbie needs advice

--

Ney André de Mello Zunino