

# Re: Architecture Advice

---

*Source:*

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework.adonet/2006-01/msg00262.html>

---

- *From:* kcamhi <[kcami@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:kcami@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)>
  - *Date:* Sat, 7 Jan 2006 19:05:02 -0800
- 

Thanks Bill – that's very helpful.

"W.G. Ryan" wrote:

> kcamhi wrote:  
>> I'm going to be using SQL Server (thanks for that advice) for a distributed  
>> application and I'd appreciate any advice on architecture.  
>>  
>> The application is similar to a membership management system at a multi-site  
>> chain -- like a fitness center chain.  
>>  
>> Tasks include:  
>> - Add new member – from any site or over the web  
>> - Edit member  
>> - Sign in/confirm member – at a location (enter ID, confirm photo)  
>> - Reporting  
>> - Etc.  
>>  
>>  
>> Simplest approach would be 2-tier client server and be done with it.  
>>  
>> However, it would be nice to be able to provide some data caching at the  
>> sites, so we can do some basic tasks if the network goes down -- like sign in  
>> a member.  
>>  
>> And I can see some benefit to having the client apps isolated from the  
>> database itself in case we ever want to move away from SQL Server, although  
>> that's not a central design goal.  
>>  
>>  
>> Any suggestions or experiences that might steer me to one solution or  
>> another? Candidates include:  
>>  
>> 1. 2-tier client/server  
>  
> ---Considering how easy it is to create true n-Tier components in .NET,  
> unless this is a small app that won't grow, I'd build it n tier.

## Re: Architecture Advice

>> 2. middle tier on the server accessed with .net remoting  
> ---You have two basic approaches, .NET Remoting/COM+ or Web Services.  
> Web Services are built on top of .NET Remoting so for performance,  
> you're going to get more out of Remoting, and TCP over Binary is going  
> to be notably faster than using WS which entails SOAP over HTTP  
>> 3. middle tier distributed at the sites that includes some data caching  
> ---This isn't mutually exclusive from #2. I'd definitely look to  
> caching. Check out the Caching Application block for some really good  
> ideas on how to implement it  
>> 4. distributed database using replication when network is online  
> ---Not sure I follow you on this, as opposed to what for instance?  
>> 5. others I'm not thinking of  
> For this, I think using Remoting and making it NTier is your best bet.  
> Remember that all you have to do to make it remotable is inherit your  
> classes from MarshalByRefObject. You can also host your components in  
> IIS for instance, and using interfaces on the client, you can have a  
> system that will allow you to drop in new DLLS and you won't have to  
> touch or recompile your client/middle tier components. You can handle  
> all such issues through config files. Also, you can split up your  
> components so that everything is remotable so you can split the load  
> over multiple machines. If you don't need/ want to up front, then you  
> can load everything on one machine (so it's client/server). Then if  
> the base grows or you need performance/security associated with  
> splitting things up, it's literally only a matter of copying files and  
> changing config file settings – nothing else. This is huge.  
>>  
>> Thanks for any war stories / advice!  
>> kc  
>  
> HTH,  
>  
> Bill  
>  
>

---

### • *Follow-Ups:*

◆ [\*Re: Architecture Advice\*](#)

◇ *From:* W.G. Ryan

### • *References:*

◆ [\*Re: Architecture Advice\*](#)

◇ *From:* W.G. Ryan

- Prev by Date: [\*Re: Architecture Advice\*](#)
- Next by Date: [\*Re: Architecture Advice\*](#)
- Previous by thread: [\*Re: Architecture Advice\*](#)
- Next by thread: [\*Re: Architecture Advice\*](#)
- Index(es):
  - ◆ [\*Date\*](#)

◆ *Thread*