

Exercise WCF Best Practices

Brian Noyes
Chief Architect, IDesign
www.idesign.net
brian.noyes@idesign.net

About Brian

Chief Architect
IDesign Inc. (www.idesign.net)

Microsoft Regional Director
MVP

Microsoft MVP
Connected Systems

E-mail: brian.noyes@idesign.net

Blog: <http://briannoyes.net>

Publishing

Developing Applications with Windows Workflow Foundation, LiveLessons training DVD, June 2007

Smart Client Deployment with ClickOnce, Addison Wesley, January 2007

Data Binding in Windows Forms 2.0, Addison Wesley, January 2006

MSDN Magazine, MSDN Online, CoDe Magazine, The Server Side .NET, asp.netPRO, Visual Studio Magazine

Speaking

Microsoft TechEd US, Europe, Malaysia, Visual Studio Connections, DevTeach, INETA Speakers Bureau, MSDN Webcasts

Agenda



Service Best Practices

Client Best Practices

Data Contracts

SOAP vs REST

Service Definition

- Separate contract from implementation
 - Contract (interface) first
- Define services in a class library, not directly in a host project
- Layering
 - Separate Service Layer?
- Instance model
 - Change to Per Call as default
 - Session / Singleton when?

Service Exception Handling

- For operation specific exceptions
 - Try/catch, throw `FaultException<T>`
- Favor using `FaultException<T>`
 - `FaultException` can be ambiguous to the client because unhandled exceptions arrive as a `FaultException`
- Include `FaultContract` in service contract definition if you throw `FaultExceptions`
 - Part of the API you are exposing
- For global exception handling from services
 - Use an error handler
- Include exception details in debug builds only

Service Security

- Intranet services
 - Default Windows Auth may be all you need
 - Possibly Hybrid – Windows Creds / Custom Application Roles
- Extranet / Internet / Custom security needs
 - Use ASP.NET Membership / Role providers
- ASP.NET providers
 - Really a standard .NET framework security infrastructure
 - Built in providers for Windows or SQL Server – based credentials / roles
 - Easy to implement custom providers
 - Establish principal on the thread
 - Re-usable across ASP.NET, WCF, WPF & Windows Forms (via Client Application Services)

Service Hosting

- Favor WAS Hosting when Server 2008 is an option
 - Multiple protocol support
 - IIS Hosting model and tools
- Favor IIS for external HTTP only services
 - Better on-box scalability / availability through worker process model
 - Better management tools
- Favor self-hosting for stateful services, callbacks, .NET Service Bus, debugging
- Have a console-based debug self-host for development time
 - Can be a Windows Service project that is used for production self-hosting with a mode switch for debugging
- Consider Dublin hosting in the future

Self Host Code

- Do not put ServiceHost in a using statement in production code
 - Dispose can throw an exception that masks the real exception thrown from Open call
 - Explicitly call Close in try/catch, log/ deal with exception in catch

Agenda

Service Best Practices



Client Best Practices

Data Contracts

SOAP vs REST

Client Proxy Classes

- Favor static proxy class over ChannelFactory
 - Connection caching in the base class in 3.5
 - Place for encapsulation of common patterns
- Hand-code or micro-code generate proxy classes for internal services
 - Less bloated code
 - Share service contract and data contracts through libraries
 - Explicit control over config file

Client Proxy Classes

- Add Service Reference for external services or when you want an async API on the client
 - Clean up config after it destroys it
 - Make sure to add references to data contract libraries before adding the service reference to avoid duplicate definitions
 - Live with the duplicate service contract definition instead of needing to repeatedly clean up the proxy code

Client Proxy Management

- Cache client proxies if frequent calls to avoid session establishment cost
 - If secure / reliable session enabled
 - Have to deal more cautiously with faulted proxies
 - Check proxy state before using
 - Get rid of proxy after exception
- Don't put proxies in a using statement
 - Dispose call might throw exception and mask real exception
 - Explicitly close in a try/catch block, and if Close throws an exception, Abort the proxy to ensure resource clean up

Client Exception Management

- All exceptions thrown from a service call derive from `CommunicationException`
- `FaultException` could be wrapped unhandled exception on the client, or explicit error returned from the service
- `FaultException<T>` always an explicit error returned from the service
- Simple approach:
 - Any exception from a proxy call, safe close the proxy
- Advanced approach:
 - `FaultException<T>` - proxy is reusable

Agenda

Service Best Practices

Client Best Practices

➤ Data Contracts

SOAP vs REST

Data Contracts

- Favor data contracts over serializable types
 - More explicit model, better control over what the client sees
- Implement `IExtensibleDataObject`
 - Avoids dropping data that the service / client does not understand
- Avoid passing complex .NET specific types for interoperable services
 - `DataSets` and `Exception` types

Data Contracts

- Avoid `XmlSerializer` and `MessageContracts` except for interoperable scenarios and REST services

Agenda

Service Best Practices



Client Best Practices

Data Contracts



SOAP vs REST

SOAP vs REST

-  Favor SOAP services when you are writing a service that only your code will consume
-  Favor REST services for publicly exposed, data oriented services

Resources

- IDesign WCF Master Class
- IDesign WCF Coding Standard – <http://www.idesign.net>
- Programming WCF, Juval Lowy, O'Reilly & Associates, 2007
- Learning WCF, Michele Leroux Bustamante, O'Reilly & Associates, 2007
- Connect Apps with WCF, Brian Noyes, Visual Studio Magazine, Feb 2008, http://visualstudiomagazine.com/articles/2008/02/01/connect-apps-with-wcf.aspx?sc_lang=en
- Wenlong Dong's Blog: <http://blogs.msdn.com/wenlong/default.aspx>

E-mail: brian.noyes@idesign.net

Blog: <http://briannoyes.net>