|  |  |
| --- | --- |
|  |  |
| Question 5: All endpoints use the same baseadress. | Mehrere Endpoints können die gleiche Adresse haben , sofern das gleiche Binding-Objekt verwendet wird. Das ist der Fall wenn mehrere Endpoints zu einem Service-Objekt gehören.  D.h. alle Endpoints müssen das gleiche Binding haben, z.B. basicHttpBinding. |
| Question 3: Hosting in IIS mit svc file | <%@ ServiceHost Language="C#" Debug="true" Service="Server.Server" %>  Als Ersatz für “new ServiceHost()” |
| Question 4: Hosting in IIS mit svc file und Factory | <%@ ServiceHost Factory="System.ServiceModel.ServiceHostFactory" Language="C#" Debug="true" Service="Server.Server" %>  <%@ ServiceHost Factory="Server.OwnServericeHostFactory" Language="C#" Debug="true" Service="Server.Server" %>  Dadurch können auch abgeleitete ServiceHost verwendet werden. |
| Question 6: Welche Binding könne in IIS verwendet werden ? | basicHttpBinding  wsHttpBinding  ws2007HttpBinding  wsDualHttpBinding  webHttpBinding |
| Asp.NET-Compatibility | [AspNetCompatibilityRequirements( RequirementsMode = AspNetCompatibilityRequirementsMode.Required) Class ServiceX {  Required 🡪 true NotAllowed 🡪 false Allowed 🡪 true oder false  <system.serviceModel> <serviceHostingEnvironment aspNetCompatibilityEnabled="true" /> |
| Question 10: ServiceHost.ApplyConfiguration | ServiceHostBase.ApplyConfiguration  Loads the service description information from the configuration file and applies it to the runtime being constructed. |
| Question 12: ServiceAuthorizationManager |  |
| Question 14: <claimTypeRequirements> | System.ServiceModel. Security.Tokens.ClaimTypeRequirement  Optional ist defaultmässig auf false.  So kann festgelegt werden, welche Claims da sein müssen.  <bindings>  <wsFederationHttpBinding>  <binding name="TTT">  <security mode="Message">  <message>  <claimTypeRequirements>  <add claimType="" isOptional="false"/>  </claimTypeRequirements>  </message>  </security>  </binding> |
| ClaimTypes | System.IdentityModel.Claims . ClaimTypes  Enthält je eine string-Konstante für jeden ClaimType  z.B.  http://schemas.xmlsoap.org/ws/2005/05/ identity/claims/emailaddress  http://schemas.xmlsoap.org/ws/2005/05/ identity/claims/gender |
| Claim | string ClaimType  Object Resource  string Right |
| ClaimSet  DefaultClaimSet X509CertificateClaimSet WindowsClaimSet | ClaimSet.Count  ClaimSet.Item[index]  ClaimSet.Issuer  Static ClaimSet.System **(Wenn mit Usr und Pwd auth . wird)**  Static ClaimSet.Windows **(Wenn Windows-Authentication verwendet wird)** |
| Zugriff auf ClaimSet | OperationContext.Current. ServiceSecurityContext. AuthorizationContext.ClaimSets  oder  css = ServiceSecurityContext.Current.AuthorizationContext.ClaimSets |
| Question 15: | SecurityTokenService (STS) (MSDN-Beispiel)  Client zum Server: wsFederationHttpBinding  Client zum SecurityTokenService: wsHttpBinding  Server zum SecurityTokenService: nie  (Server prüft mit Hilfe des STS-Zertifikates, ob die Tokens gut sind.)  wsFederationHttpBinding wird verwendet, damit das Custom Token übertragen werden kann. |
| Message | MS\_Samples\WCF\Basic\Contract\Message\Untyped  public static [Message](bfa595cd-d85d-bcef-3dec-267b8cb534b2.htm) CreateMessage(  [MessageVersion](6733a448-16cc-1cab-cf37-98779c1da509.htm) version,  string action,  Object body  )  public T GetBody<T>()  T ist der ursprüngliche Body-Typ sein. |
| Message-Class  Wie kann der Inhalt der Message ausgegeben werden ? | XmlWriter xw = XmlWriter.Create("test.xml");  msg.WriteMessage(xw);  xw.Close();  <?xml version="1.0" encoding="utf-8"?>  <s:Envelope xmlns:s="…">  <s:Header>  <Action s:mustUnderstand="1" xmlns="…">uhu</Action>  </s:Header>  <s:Body>  <int xmlns="…">12</int>  </s:Body>  </s:Envelope> |
|  | XmlDictionaryReader xdw = msg.GetReaderAtBodyContents();  string x = xdw.ReadOuterXml();  <int xmlns="http://schemas.microsoft.com/ 2003/10/Serialization/">12</int> |
|  |  |
| Welche bindings unterstützen streaming ? | Streaming ist besonders gut für sehr grosse Datenmengen. (z.B. 50MB)  basicHttpBinding  netTcpBinding  netNamedPipeBinding  <http://haveyougotwoods.com/archive/2008/04/14/wcf-message-streaming.aspx>  <basicHttpBinding> <binding name="XY" **transferMode**="StreamedRequest" /> </basicHttpBinding>  **Streamed** - Both in and out messages are streamed  **StreamedRequest** - Messages sent from client to server are streamed **StreamedRespone** - Only messages returned from the server to the client are streamed  **Buffered** - This is the default of buffering all data and sending it in one burst  when using streams the only allowed data types are **Message**, **Stream**, or an **IXMLSerializable** implementation for **ALL** methods in your service |
| Transactions on operations | [TransactionFlow(TransactionFlowOption.Mandatory)]  [OperationBehavior( TransactionAutoComplete = true, TransactionScopeRequired = true )] |
| Transactions on behavior | [ServiceBehavior( ConcurrencyMode=ConcurrencyMode.Single, InstanceContextMode=InstanceContextMode.PerSession, ReleaseServiceInstanceOnTransactionComplete=false, TransactionIsolationLevel=System.Transactions.IsolationLevel.ReadCommitted, TransactionTimeout = "00:03:00" )]  “00:03:00”: drei Minuten |
| Transaction (defaults) | TransactionAutoCompleteOnSessionClose=false ReleaseServiceInstanceOnTransactionComplete=trueTransactionIsolationLevel= Serializable |
| Welche Bindings unterstützen Transactions ? | Alle ausser “BasicHttpBinding” |
| Braucht es eine Session für Transaction ? | <http://social.msdn.microsoft.com/Forums/en-US/wcf/thread/39e98e70-fe8e-4108-b7a1-7fe6e66fa012>  Nein, Transactions brauchen keine Sessions.  TransactionAutoCompleteOnSessionClose = true (requires a session since without a session, there cannot be a session close)  TransactionAutoComplete = false (requires the use of a session since without a session, there cannot be a later method that would mark the transaction complete) |
| ServiceKnownType (1. Möglichkeit) | [ServiceKnownType(typeof(Widget))] [ServiceKnownType(typeof(Machine))] [ServiceContract()]  public interface ICatalog2 { |
| ServiceKnownType (1. Möglichkeit) | [ServiceKnownType("GetKnownTypes", typeof(Helper))]  static class **Helper** { public static IEnumerable<Type> **GetKnownTypes**(ICustomAttributeProvider provider) { System.Collections.Generic.List<System.Type> knownTypes = new System.Collections.Generic.List<System.Type>(); // Add any types to include here. knownTypes.Add(typeof(Widget)); knownTypes.Add(typeof(Machine)); return knownTypes; } } |
| WebGet  WebInvoke | <http://hyperthink.net/blog/http-pox-programming-basics/>  <http://mtaulty.com/CommunityServer/blogs/mike_taultys_blog/archive/2007/05/01/9279.aspx>  POX=Plain Old XML  JSON=JavaScript Object Notation  non-SOAP scenarios  WebGet 🡪 für GET-Requests  WebInvoke 🡪 für POST (kann aber auch auf PUT und auch für andere eingestellt werden.) |
| POX konfigurieren | Binding: webHttpBinding  public class WebHttpBehavior : [IEndpointBehavior](c77ee01c-f80c-5a6f-4859-1df42c2df0ce.htm)  <endpointBehaviors>  <behavior name="W">  <webHttp/>  </behavior>  </endpointBehaviors> |
| JSON konfigurieren | MS\_Samples\WCF\Extensibility\Ajax\JSONP |
| Message-Inspector (Client) | IClientMessageInspector  void AfterReceiveReply(ref System.ServiceModel.Channels.Message reply, object correlationState)  object BeforeSendRequest(ref System.ServiceModel.Channels.Message request, IClientChannel channel)  Message kann verändert werden. So können z.B. auch neue Headers hinzugefügt werden.  Es kann auch eine komplett neue Message hinzugefügt werden. |
| Message-Inspector (Server) | IDispatchMessageInspector  Object AfterReceiveRequest(  ref Message request,  IClientChannel channel,  InstanceContext instanceContext  )  void BeforeSendReply(  ref Message reply,  Object correlationState  ) |
| Parameter-Inspector (client und Server) | public interface IParameterInspector  void AfterCall(  string operationName,  Object[] outputs,  Object returnValue,  Object correlationState  )  Object BeforeCall(  string operationName,  Object[] inputs  ) |
| COM+ über WCF freigeben. | ComSvcConfig.exe  <http://msdn.microsoft.com/en-us/library/ms734723.aspx> |
| Interoperability with Web Services Enhancements 3.0 | WCF clients and services are configured to use the **August 2004** version of the WS-Addressing specification. |
| MSQ private queue | <add key="queueName" value=".\private$\ServiceModelSamples" />  if (!MessageQueue.Exists(queueName)) MessageQueue.Create(queueName, true);  net.msmq://localhost/private/ServiceModelSamples  Public queue MachineName\QueueName  Private queue MachineName\Private$\QueueName |
| SecurityTokenAuthenticator | protected abstract ReadOnlyCollection<IAuthorizationPolicy> ValidateTokenCore(  SecurityToken token  )  When the ValidateTokenCore method is overridden, follow these guidelines:  When the security token passed into the token parameter cannot be validated, throw the SecurityTokenValidationException exception.  When there are no authorization policies in effect for this application, return an empty ReadOnlyCollection<(Of <(T>)>) of type IAuthorizationPolicy.  When nullNothingnullptra null reference (Nothing in Visual Basic) is returned by this method, Windows Communication Foundation (WCF) throws the SecurityTokenValidationException exception. |
| SecurityPermission | [PrincipalPermission |
|  |  |
|  |  |

|  |  |
| --- | --- |
|  |  |
|  |  |
| WMI |  |
| Ein WebService wo nur admins zugreifen können |  |
| Welche bindings unterstützen reliability ? |  |
|  |  |
|  |  |