meet the experts of enterprise infrastructure

JAVAFORUM 2009 stuttgart am 2. Juli 2009

Interoperabilität einmal anders

Thomas Haug

thomas.haug@mathema.de

www.mathema.de

MATHEMA

Agenda

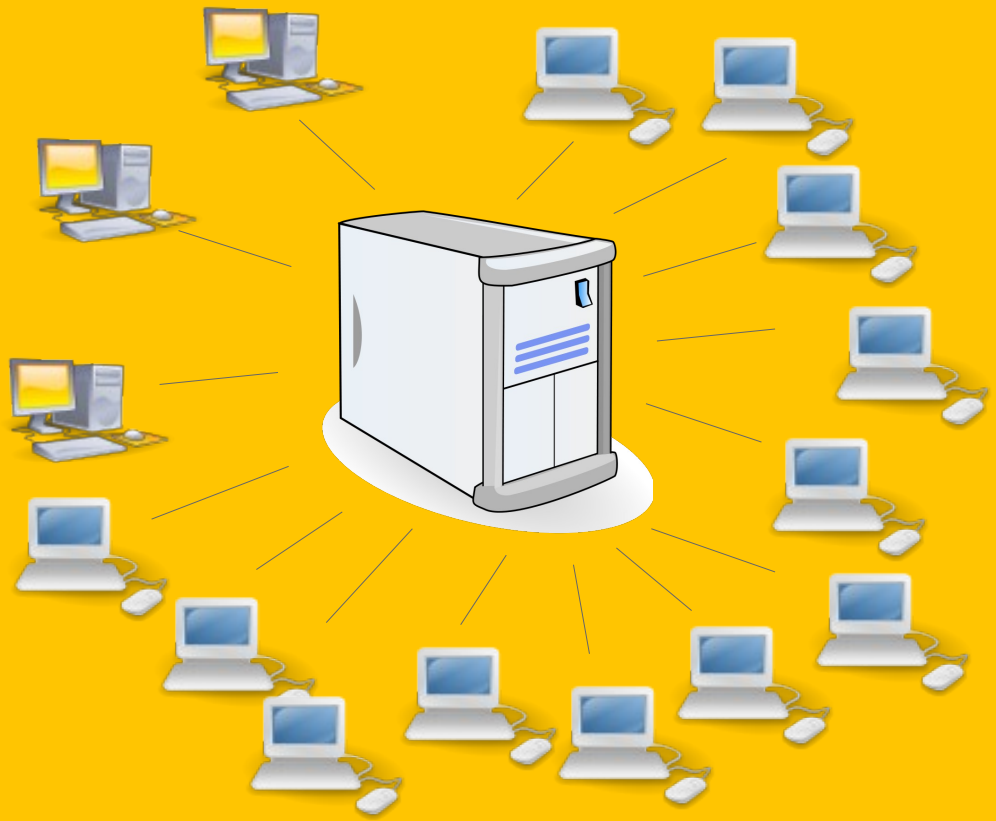
Foto: pixelio.de – knipseline

Agenda

Foto: pixelio.de – knipseline

Motivation

Heterogenität



Motivation

„The SOAP stack is generally regarded an embarrassing failure these days“

Tim Bray („Co-Erfinder“ von XML und REST Verfechter)

Warum kein SOAP und WS

Performance ?

aboutpixel.de / Formel 1 © Stefan Kofler

Warum kein SOAP und WS

Schnittstellen-Beschreibung ?

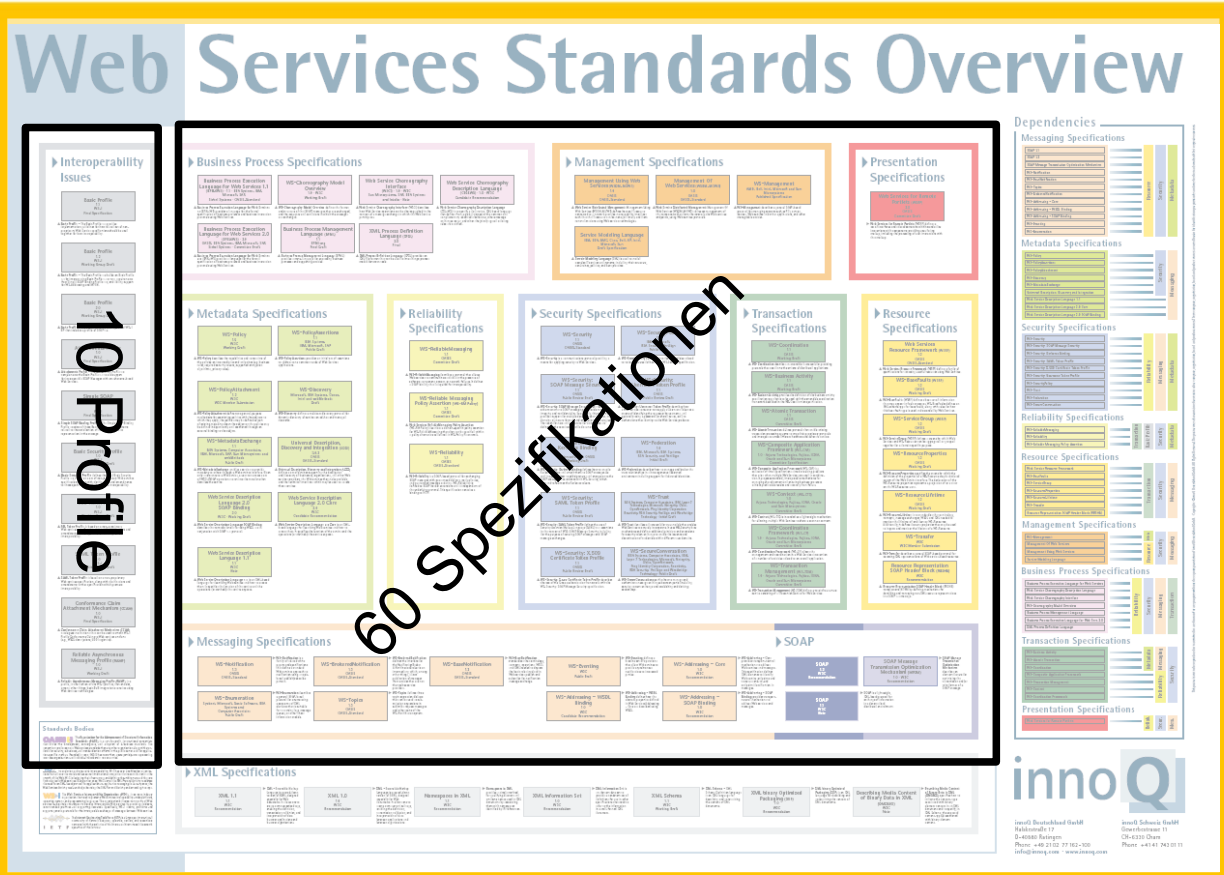
Foto: pixelio.de – Markus Hein

meet the experts of enterprise infrastructure

Warum kein SOAP und WS

JAVAFORUM 2009 stuttgart am 2. Juli 2009

Interoper-abilität ?



http://www.innoq.com/soa/ws-standards/poster/innoQ%20WS-Standards%20Poster%202007-02.pdf

MATHEMA

Warum kein SOAP und WS

SOAP
Toolkit
Sammel-
surium

Foto: pixelio.de - Olaf Rendler

Agenda

Foto: pixelio.de – knipseline

Alternativen

IIOP.NET AN ELCA CONTRIBUTION

reliable open source caucho

Hessian C#

Internet Communication Engine

- ZeroC (www.zeroc.com)
- Aktuelle Version 3.3.1 (März 2009)
- GNU General Public License
- Unterstütze Sprachen

Java

Ruby
A Programmer's Best Friend

C++

php

python™

Microsoft Visual Studio

Foto: pixelio.de - korkey

MATHEMA

Ice Eigenschaften

- Einheitliches (, einfaches) Programmier-modell für alle Plattformen und Sprachen
- Server-seitiges Threading-Modell
  - Grundsätzlich Multi-threaded
  - Thread-Pool(s) sind konfigurierbar
- Client-seitiges Threading-Modell
  - Standard 2 Threads
    - 1 Thread zum Senden
    - 1 Thread für Callbacks
    - ist erweiterbar

Foto: pixelio.de - korkey

Ice Eigenschaften

- Transportprotokolle
  - TCP
  - UDP
- Sicherheit über SSL
- Aufrufsematik
  - Synchron, Asynchron, One-Way, Batched
- *Sehr gut Dokumentiert*
- *Schnell und Effizient*

Foto: pixelio.de - korkey

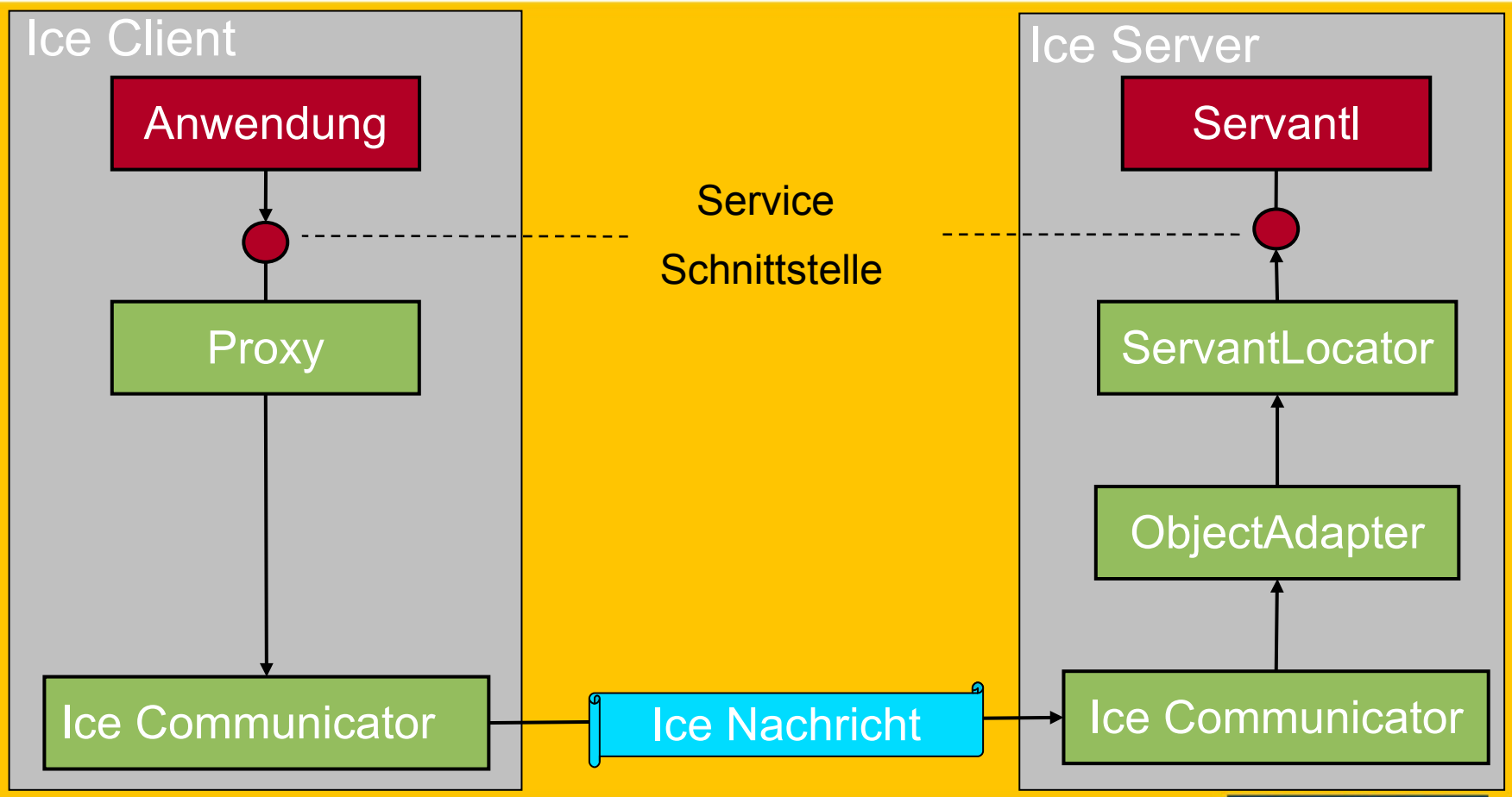
Eiskalte Helfer

- IceBox
- IceStorm
- Freeze
- FreezeScript
- Glacier2
- IcePatch2
- IceGrid

Foto: pixelio.de - korkey

MATHEMA

Ice Architektur Überblick



Specification Language for Ice (SLICE)

```
 1:   module ohneSeife {
 2:      exception Fehler {};
 3:      interface DemoService {
 4:         void melde(string wert) throws Fehler;
 5:         string getWert();
 6:         void getWert2(out string wert2);
 7:         idempotent void sayHello(string text);
 8:         void WerteTuple get();
 9:      };
10:   };
```

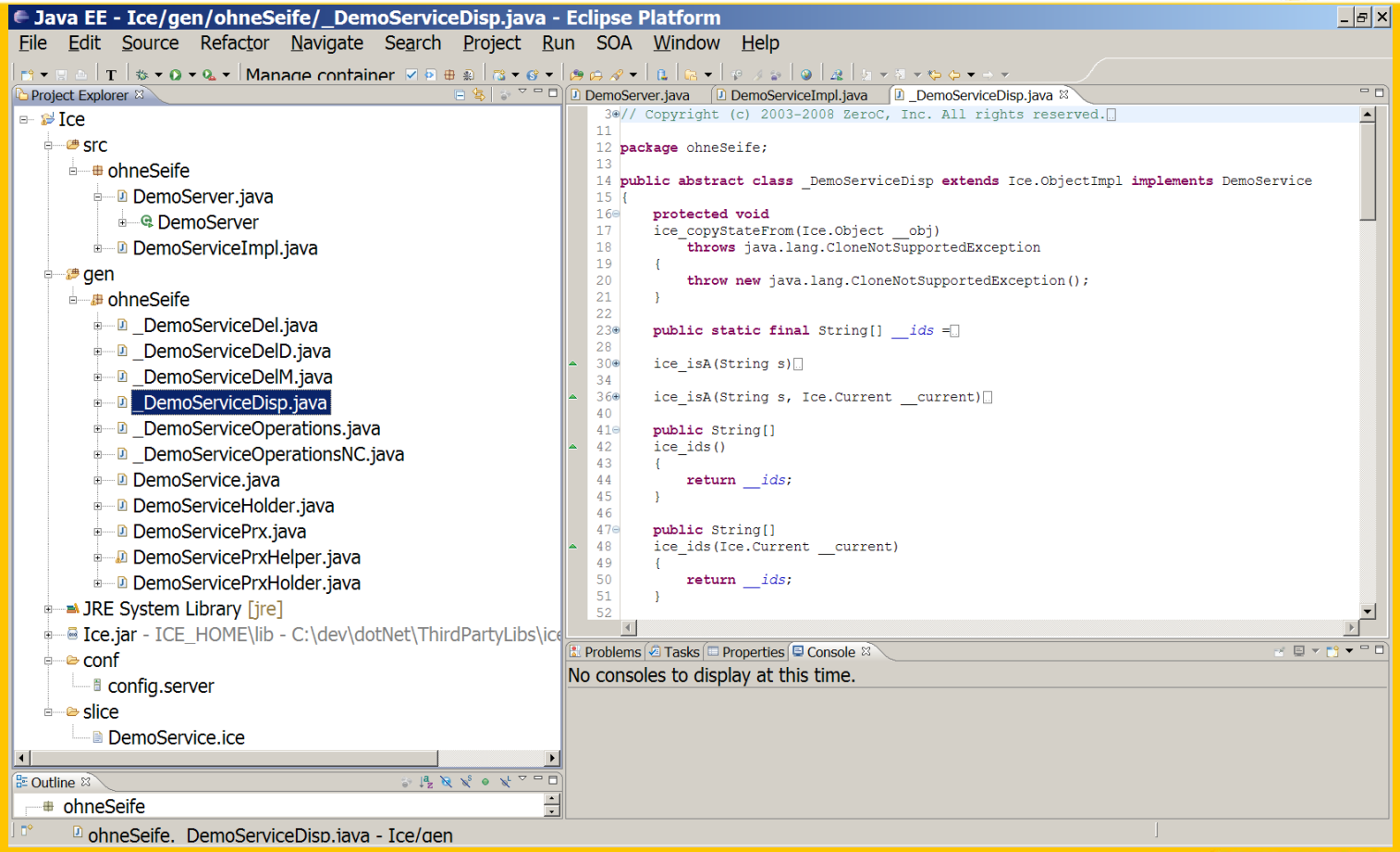
```
 1:   struct WerteTuple {
 2:      string name;
 3:      short  wert;
 4:   };
```

Alternative 1

```
 1:   class WerteTuple {
 2:      string name;
 3:      short  wert;
 4:      string toString();
 5:   };
```

Alternative 2

Ice generierte Artefakte



Java Ice Servant


```
1: public class DemoServiceImpl extends _DemoServiceDisp {

2:     public void sayHello(String txt, Ice.Current ctx) {
           System.out.println("Client gesendet "+txt);
           displayCurrent(ctx);
       }

3:     private void displayCurrent(Ice.Current current) {
           System.out.println("Current ID name     = "
                              +current.id.name);
           System.out.println("Current Mode       = "
                              +current.mode.toString());
           System.out.println("Current Adapter    ="
                              +current.adapter.getName());
       }
   }
```

Java Ice Server


```
1: package ohneSeife;
   import Ice.Application;

2: public class DemoServer extends Application {

3:    public static void main(String[] args) {
         DemoServer server = new DemoServer();
         int status = server.main("DemoServer", args,
                                  "conf\\config.server");
         if (status != 0) {
            System.exit(status);
         }
      }
4:    public run () { // nächste Folie }

5: }
```

Java Ice Server


```
1: public int run(String[] args) {

2:     Ice.ObjectAdapter adapter =
           communicator().createObjectAdapter("DemoService");

3:     adapter.add(new DemoServiceImpl(),
           communicator().stringToIdentity("DemoServant"));

4:     adapter.activate();

5:     communicator().waitForShutdown();

6:     return 0;

7: }
```

Java Ice Server Konfiguration


```
1: DemoService.Endpoints=tcp -p 10000:udp -p 10000

2: # Verbindungs Probleme protokollieren
   Ice.Warn.Connections=1

3: # Server-seitiges ACM in Sekunden
   Ice.ACM.Server=10

4: # Netzwerk Protokollierung
   # 0 -nix, 1 öffnen schliessen, 3 Datentransfer
   Ice.Trace.Network=1

5: # Protokoll Tracing
   # 0 - kein, 1 Tracing
   Ice.Trace.Protocol=0
```

Ice Client implementieren (C#)


```
1: namespace ohneSeife {

2:     class IceClient : Ice.Application {

3:         public static void Main(string[] args) {

4:             IceClient client = new IceClient();

5:             int status = client.main(args,
                                        "conf\\config.client");
6:             if (status != 0)
                   System.Environment.Exit(status);
7:         }

8:         public override int run(string[] args) { ... }
       }
   }
```

Ice Client implementieren (C#)


```
1:  public override int run(string[] args) {

2:      Ice.ObjectPrx iceObject =
          communicator().propertyToProxy("DemoService.Proxy")
          .ice_twoway().ice_timeout(-1).ice_secure(false);

3:      DemoServicePrx proxy =
          DemoServicePrxHelper.checkedCast(iceObject);

4:      if (proxy != null) {
           proxy.sayHello("Hallo ein C# Ice Client");
           return 0;
        }
        return 1;

5:  }
```

Ice Client Konfiguration


```
1: DemoService.Proxy=DemoServant:tcp -p 10000

2: # Verbindungs Probleme protokollieren
   Ice.Warn.Connections=1

3: # Netzwerk Protokollierung
   # 0 -nix, 1 öffnen schliessen, 3 Datentransfer
   Ice.Trace.Network=1

4: # Protokoll Tracing
   # 0 - kein, 1 Tracing
   Ice.Trace.Protocol=0
```

Ice Vor- und Nachteile

- ✔ Einheitliches Programmiermodell für alle Plattformen
- ✔ Performance
- ✔ Sehr gute Dokumentation
- ▪ Reliable Messaging (ordered messages) wird unterstützt
- ▪ Kompexität
- ✗ Keine Transaktionen
- ✗ Vendor Lock-In

Foto: pixelio.de - korkey

MATHEMA

ActiveMQ

- Apache Projekt
- Akutelle Version 5.2.0
- JMS 1.1 Provider
  - Nachrichten-basierte Kommunikation
- Nachrichten
  - transient
  - persistent
- Transaktionsanbindung (auch XA)

Foto: aboutpixel.de © Claudia Keimel

ActiveMQ

- Unterstützte Sprachen (Clients)
  - über OpenWire
  - über Stomp
- .Net Messaging API (NMS)

Java

Ruby
A Programmer's Best Friend

Perl

C++ Smalltalk

php

python

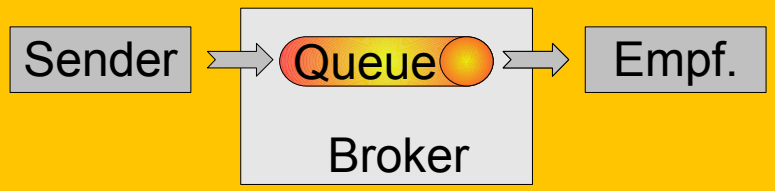
Microsoft Visual Studio

Foto: aboutpixel.de © Claudia Keimel

MATHEMA

ActiveMQ Kommunikationsmodell

- Point-to-Point



- Publisher-Subscriber

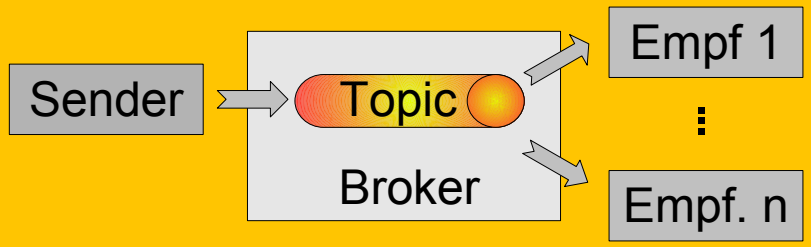


Foto: aboutpixel.de © Claudia Keimel

MATHEMA

JMS Publisher in Java


```
1:   ConnectionFactory factory = new
        ActiveMQConnectionFactory("tcp://localhost:61616");
     Connection connection = factory.createConnection();

2:   Session jmsSession = connection.createSession(false,
                                        Session.AUTO_ACKNOWLEDGE);
     Topic topic = jmsSession.createTopic("DemoTopic");

3:   MessageProducer producer =
                               jmsSession.createProducer(topic);
     producer.setDeliveryMode(DeliveryMode.NON_PERSISTENT);

4:   jmsSession.createTextMessage("Hallo eine Nachricht");
     producer.send(message);
```

NMS Subscriber in C#


```
1:   using Apache.NMS.ActiveMQ;
     using Apache.NMS;
     namespace NMSSubscriber {
        class Subscriber {
          static void Main(string[] args) {

2:           ConnectionFactory factory = new
                   ConnectionFactory("tcp://localhost:61616");
             IConnection con = factory.CreateConnection();

3:           ISession jmsSession =
           con.CreateSession(AcknowledgementMode.AutoAcknowledge);
             ITopic topic = jmsSession.GetTopic("DemoTopic");

4:           IMessageConsumer consumer =
                              jmsSession.CreateConsumer(topic);
```

NMS Subscriber in C#


```
5:              consumer.Listener += new MessageListener(
                                          new Client().OnMessage);
                connection.Start();
            }

6:          public void OnMessage(IMessage message) {
              try {
                 if (message is ItextMessage) {
                   ITextMessage msg = message as ITextMessage;
                   Console.WriteLine("Nachricht : " + msg.Text);
                 }
              }
              catch (NMSException e) {
                 Console.WriteLine(e);
              }
            }
```

Transaktionale Nachrichten mit NMS


```
1: ConnectionFactory factory = new ConnectionFactory(URL);
   IConnection con = factory.CreateConnection();

2: ISession jmsSession =
      con.CreateSession(AcknowledgementMode.Transactional);
   ITopic topic = jmsSession.GetTopic("DemoTopic");
```

```
1: public void OnMessage(IMessage message) {
       ITextMessage textMessage = message as ITextMessage;
       // ...
       if (...) // OK
           this.session.Commit();
       Else // Nicht OK
           this.session.Rollback();
   }
```

Giftige Nachrichten mit NMS


```
1:  MessageConsumer activeMqConsumer = consumer as MessageConsumer;

2:  if (activeMqConsumer != null) {
       activeMqConsumer.MaximumRedeliveryCount = 3;
    }
```

```
1:   public void OnMessage(NMS.ITextMessage message) {
      if (message.NMSRedelivered) {
          Console.WriteLine("Nachtrichten Wiederholung");
      }
     // ...
  }
```

Vor- und Nachteile

- ✔ Integrationsaufwand
  - Ähnliches Programmiermodell in Java und .Net
- ✔ JMS Transaktionen werden unterstützt
- ✔ Reliable Messaging wird unterstützt
- ✗ Dokumentation NMS
- ✗ Professional Services ?

Foto: aboutpixel.de © Claudia Keimel

Agenda

Foto: pixelio.de – knipseline

Fazit

Es gibt sowohl für RPC-basierte als auch Nachrichten-basierte Systeme Alternativen zu SOAP und Webservices

Die Entscheidung, ob und welche Alternative einsetzbar ist, ist grundsätzlich vom Projekt und dessen Anforderungen abhängig

Fazit

Eigenschaft	WCF / Java	ActiveMQ	Ice
Lizenzmodell		Apache	(GPL)
Codierung	XML	binär	binär
Transport	HTTP	TCP	TCP UDP
Firewall-tauglich	leicht	schwierig	möglich
Programmier-modell	Modell des eingesetzten Produkts	JMS (-like)	Gleiches Modell für .Net & Java
Schnittstellen	WSDL	*API*	SLICE
Dokumentation	++	+	++
Prof. Service	++	+	++

Fazit

Eigenschaft	WCF / Java	ActiveMQ	Ice
Sicherheit	WS Security HTTS		OpenSSL
Transaktionen	WS Transaction	JMS Transaction	nein
Zuverlässigkeit	WS Reliability	JMS durable	*Bedingt* *(keine persistenten Nachrichten)*
Performance	-	+	++

Fragen?

Vielen Dank!

Foto: pixelio.de – Marc Tollas

thomas.haug@mathema.de

MATHEMA

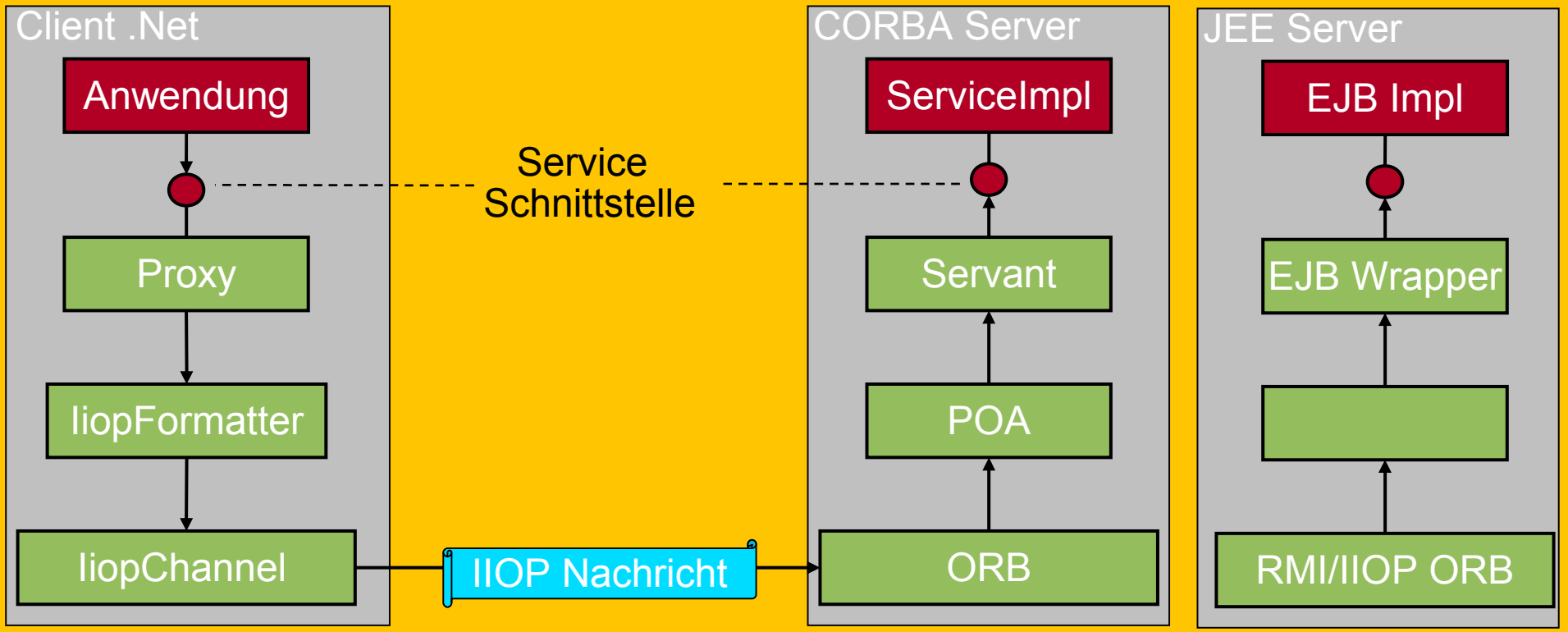
Zusätzliche Alternativen

IIOP.Net 1/4

- Open Source Framework für .Net
  - http://iiop-net.sourceforge.net/
  - LGPL Lizenz Model
  - In C# implementiert
- Aktuelle Version 1.9 SP1 (Oktober 2008)
- Basiert auf CORBA 2.3
  - Bidirectional IIOP
  - Value Types
  - Portable Interceptors
- IIOP/SSL über ein Plugin

IIOP.Net 2/4

- IIOP Integration über das .Net Remoting Framework



IIOP.Net 3/4


```
1:   module ohneSeife {
2:      interface DemoService {
3:         long meinIntTest(in long wert);
           string meinStringTest(in string zeichenkette);
           Komponent echo(in Komponent komponent);
        };
4:   };
```

Mittels IDLToCLS .Net Assembly erzeugen

```
1:   string ior = "IOR:000000....14300";
2:   IiopClientChannel channel = new IiopClientChannel();
3:   ChannelServices.RegisterChannel(channel,false);

4:   DemoService demoService = (DemoService)
            RemotingServices.Connect(typeof(DemoService), ior);

5:   demoService.meinStringTest("Hallo, ein .Net Client");
```

IIOP.Net 4/4 Vor- und Nachteile

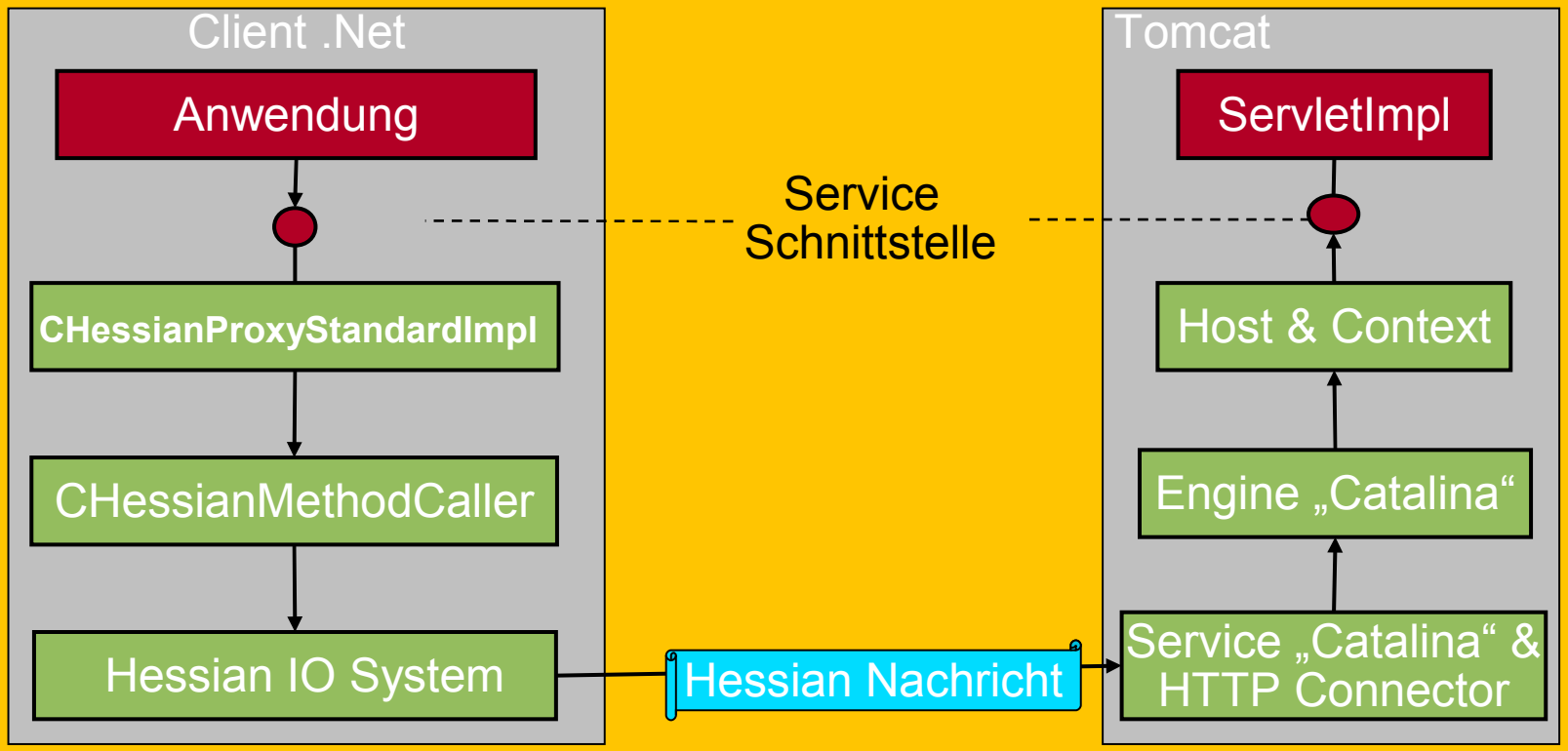
- ✔ Integrationsaufwand
  - Leichte Integration in existierende CORBA Landschaft möglich
    - Nicht nur auf Java beschränkt
  - Bettet sich nahezu nahtlos in das .Net Remoting Framework ein
- ✔ Unterstützung von COS Naming ist vorhanden
- Nutzung bestehender CORBA Services wie COS Transaction und COS Notification ist möglich, aber mit Entwicklungsaufwand verbunden.
- ✗ Schlecht geeignet für Szenarien, in denen Firewalls durchdrungen werden müssen

Hessian 1/5

- Binäres Web Service Protokoll für RPC
- Spezifikation von Caucho
  - http://hessian.caucho.com/
  - Binär-codiert Nachrichten
  - Transportprotokoll HTTP
- RPC basierte Kommunikation
- Implementierungen u. a. für
  - Java (http://hessian.caucho.com/, 3.2 oder Alternative Hessian4J 1.0)
  - .Net (C#) (http://www.hessiancsharp.org/, 1.3.3, LGPL)
- Keine explizite Schnittstellen Beschreibungssprache

Hessian 2/5

- Hessian C# Client und Tomcat-basierter Hessian Service



Hessian 3/5

- Beispiel Schnittstelle in C# und Java

```
1:   namespace HessianClient {
2:      interface IdemoService {
3:         int meinIntTest(int wert);
           string meinStringTest(string zeichenkette);
           Komponent echo(Komponent component);
4:      }
5:   }
```

```
1:   package service;
2:   public interface DemoService {
3:      int meinIntTest(int wert);
        String meinStringTest(String zeichenkette);
        Komponent echo(Komponent component);
4:   }
```

Hessian 4/5

C# Client

```
1:   string url = "http://"+host+":8080/hessian/DemoService";
2:  CHessianProxyFactory factory = new CHessianProxyFactory();
3:  IDemoService demoService =
       (IDemoService)factory.Create(typeof(IDemoService), url);
4:  demoService.meinStringTest("Hallo, ein .Net Client");
```

Java Service

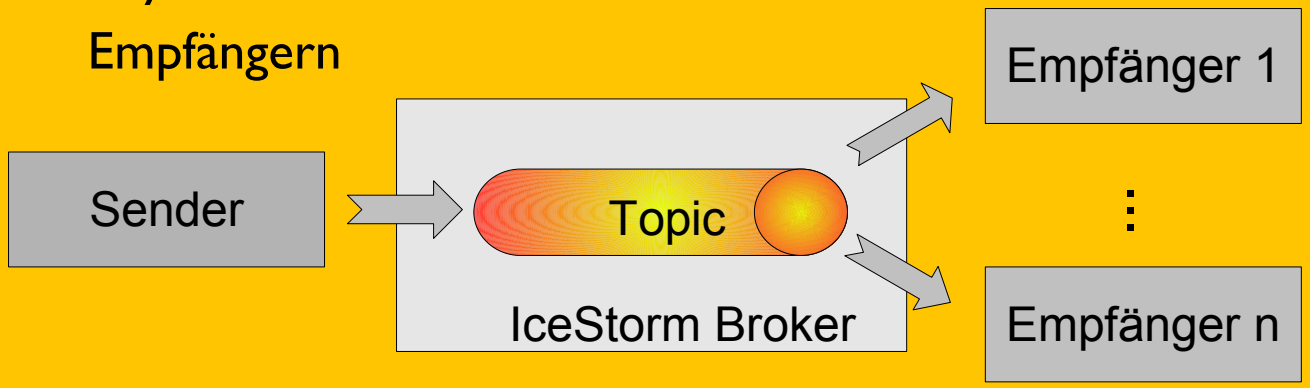
```
1:  public class DemoServiceImpl extends HessianServlet
                                 implements DemoService {
2:      public int meinIntTest(int wert) {
           System.out.println("Integer '"+wert+"' empfangen.");
           return wert++;
        }

3:      public String meinStringTest(String zeichenkette) {
           System.out.println("String '"+zeichenkette
                              +"' empfangen.");
           return "Hallo Client, server hat '"+zeichenkette
                  +"' von Ihnen empfangen";
        }
4:  }
```

Hessian 5/5 Vor- und Nachteile

- ✔ Integrationsaufwand
  - Einfaches Programmiermodell
  - Integration in Java Web Container
- ✔ Eignet sich für Szenarien, in denen Firewalls durchdrungen werden müssen
- ✔ Sicherheit über HTTPS
- Lastverteilung lässt sich mit mäßigem Aufwand realisieren
- ✗ Kein einheitliches (Server-seitiges) Programmiermodell
- ✗ Keine Unterstützung für Namensdienste/Registries, Transaktionen und Reliable Messaging

IceStorm 1/3

- IceStrom unterstützt ausschließlich Publisher-Subscriber
  - Asynchrone Kommunikation zwischen einem Sender und mehreren Empfängern



- Federation von IceStrom Broker ist möglich
- Nachrichten sind **nicht** persistent
- Topics und Subscriber können persistent sein
- **Keine** Transaktionen wie bei JMS

IceStorm 2/3

- Ice(Storm) beschreibt Schnittstellen mittels SLICE

```
1:   module Demo {
2:      interface DemoService {
3:         void melde(string wert);
4:      };
5:   };
```

Ice Publisher in Java

```
1:    IceStorm.TopicManagerPrx manager =
        IceStorm.TopicManagerPrxHelper.checkedCast(
         communicator().propertyToProxy("TopicManager.Proxy"));
2:   IceStorm.TopicPrx topic = manager.retrieve("DemoTopic");
3:   DemoService demo =
                DemoServicePrxHelper.uncheckedCast(publisher);
4:   demo.melde("Hallo eine Nachricht");
```

IceStorm 3/3


```
1:    IceStorm.TopicManagerPrx manager =
       IceStorm.TopicManagerPrxHelper.checkedCast(
         communicator().propertyToProxy("TopicManager.Proxy"));
2:   IceStorm.TopicPrx topic = manager.retrieve("DemoTopic");
3:   Ice.ObjectAdapter adapter =
       communicator().createObjectAdapter("Demo.Subscriber");

4:  Ice.Identity subId = new Ice.Identity(id,
                                    Ice.Util.generateUUID());

5:  Ice.ObjectPrx subscriber = adapter.add(new ServiceImpl(),
                                           subId);
6:  topic.subscribeAndGetPublisher(qos, subscriber);
7:  adapter.activate();

8:   public class ServiceImpl : DemoServiceDisp_ {
         public override void melde(string wert,
                                    Ice.Current current)
         {
             Console.WriteLine(wert);
         }
     }
```

Ice Subscriber in C#