

Tec 07

J2EE versus .NET

Michael Stal
Siemens AG, Corporate Technology



Agenda

- **Motivation**
- **Comparison**
 - Overall Vision: Java and .NET
 - Layer-by-Layer comparison of the infrastructures
- **Summary**

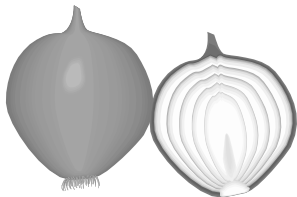
Goal

- **This is intended to be an objective comparisons of the two platforms**
- **In will contain criteria to base a decision which platform to use**
- **Interoperability issues**

Some Issues

- **If we are going to compare .NET and Java, what do we mean?**
 - Will we compare Java, the language, with C#?
 - Will we compare Java, the platform, with .NET, the platform?
 - Will we compare the Java Virtual Machine with the CLR (Common Language Runtime)?
- **In this talk we will try to cover most aspects.**

Layer-By-Layer Comparison



The Naive Approach Hello World Example

- **In C# and Java:**

```
using System;
namespace MyNameSpace {
    public class MyClass {
        public static void Main(String [] args) {
            Console.WriteLine(„Hello, C#!“);
        }
    }
}
```

```
package MyPackage;
import java.lang.*;
public class MyClass {
    public static void Main(String [] args) {
        System.out.println(„Hello, Java!“);
    }
}
```

Layers

- High Level Overview
- Runtime System
- Object model
- Base class libraries
- Enterprise support
 - Component model
 - Database access
 - XML
 - Server Pages
 - Remoting
 - Web Services
 - More Enterprise APIs

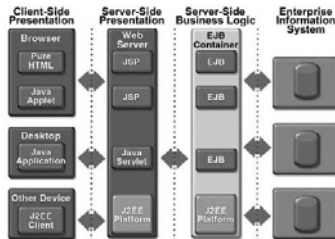
What is J2EE Technology?

• not just a set of APIs

J2EE Technology	Purpose
Platform Specification	Describes the minimum feature set of J2EE APIs and standards that vendors must provide
Reference implementation	Provides a compliant and operational J2EE platform
J2EE Blueprints	Describes how to construct J2EE applications (including JAVA Pet Store)
Compatibility Testsuite	Validates J2EE platform compatibility to guarantee code portability and eliminate vendor lock

What is J2EE Technology?

Java-based Multi-Tier Architectures: J2EE supports appropriate technologies for each tier



What is J2EE Technology?



J2EE is based on J2SE, which includes the following enterprise APIs:

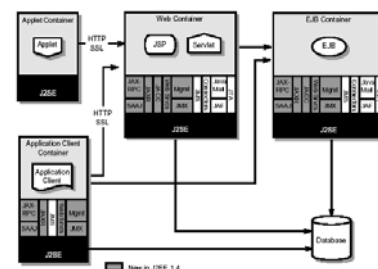
- Java IDL and RMI-IIOP
- JAAS (Java Authentication and Authorization Service)
- JDBC (Java Data Base Connectivity)
- JNDI (Java Naming and Directory Interface)
- JAXP (Java API for XML Parsing)

J2EE „standard“ Services

- Servlets
- Java Server Pages
- Java RMI-IIOP and RMI-JRMP
- JavaIDL
- JNDI (Java Naming and Directory Interface)
- JMS (Java Message Service)
- Java Authentication and Authorization Service (JAAS)
- Enterprise JAVA Beans
- JDBC (Java Data Base Connectivity)
- J2EE Connector Architecture
- Java Transaction API
- JavaMail and JAF
- Java API for XML Parsing

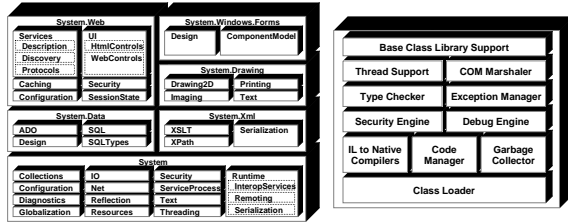
Of course I cannot cover them all.

Multi-tier Architecture Overview

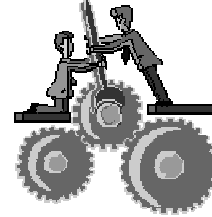


The .NET Framework Class Library

- In .NET we distinguish between .NET Framework and .NET CF

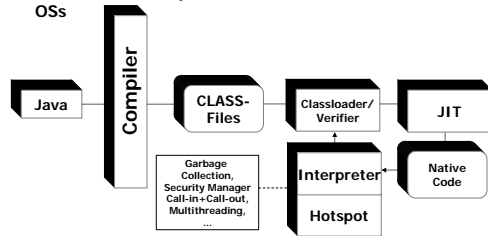


The Runtime System



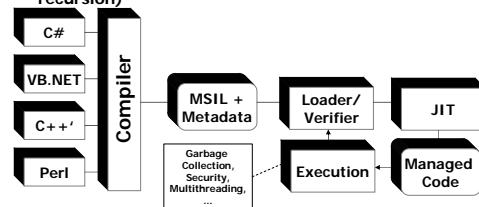
Java Virtual Machine

- The JVM targets Java and interprets Java Byte Code.
- Other languages can also be compiled to Java bytecode (e.g. Ada)
- Just-in-Time compilers exist for different environments and OSs



.NET Runtime

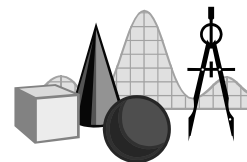
- It is called the Common Language Runtime (CLR)
- It supports all languages compiled for the MSIL
- Provides integration for several languages
- Provides support for non-OO languages (e.g. tail recursion)



Differences and Commonalities

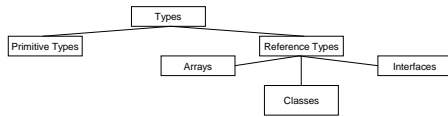
- Commonalities:**
 - Basic infrastructure is similar
- Differences:**
 - Java is intended for interpretation (e.g. type-dependent primitives i_add, ...)
 - Java allows for custom class loaders and security managers
 - .NET is optimized for compilation and is thus sometimes faster
 - .NET CLR provides a command set that also supports functional languages

The Object Model



Object Model (Java)

- Java has primitive types and classes.
 - No automatic boxing/unboxing



java.lang.Object

- Base of all Java classes

```

public class Object {
    public Object();
    public boolean equals(Object obj);
    public final Class getClass();
    public int hashCode();
    public final void notify();
    public final void notifyAll();
    public String toString();
    public final void wait() throws InterruptedException;
    public final void wait(long timeout) throws
        InterruptedException;
    public final void wait(long timeout, int nanos)
        throws InterruptedException;
    protected Object clone() throws CloneNotSupportedException;
    protected void finalize() throws Throwable;
}
    
```

Object Model (Java)

- Primitive types cannot be transparently used as an object. Special Holder classes are necessary.

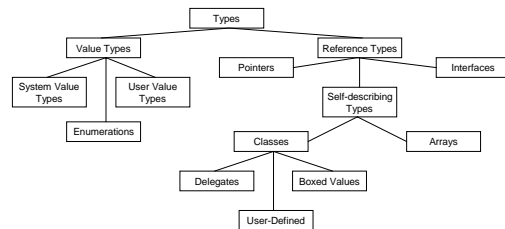
```

Integer i_ref = new Integer(7);
List l = ...
l.add(i_ref);
    
```

- There are no special function references. Java uses Observer Pattern with callback interfaces and inner classes

Object Model (.NET)

- In .NET, everything is a class



System.Object

- The „mother of all .NET classes“

```

public class Object {
    public virtual int GetHashCode();
    public virtual bool Equals();
    public virtual String ToString();
    public static bool Equals(object a, object b);
    public static bool ReferenceEquals(object a,
        object b);

    public Type GetType();
    protected object MemberwiseClone();
    protected virtual Finalize();
}
    
```

Object Model (.NET)

- .NET distinguishes between values types and reference types:
 - value types reside on the stack
 - reference types reside on the heap
- There is no difference between primitive types and classes
 - Automatic boxing/unboxing provides transparency
- Special strongly-typed function references
 - called *delegates*
 - and *events*

.NET-Types that are not available in Java

Delegates & Events:

```
class Sub {
    public void somebodyTurnedOnTheLight( int which ) {
        // do something useful
    }
    public Sub(Pub thePub) {
        thePub.OnLightTurnedOn +=
            new Pub.LightTurnedOn(somebodyTurnedOnTheLight);
    } // ...
}

class Pub {
    public delegate void LightTurnedOn(int ID);
    public event LightTurnedOn OnLightTurnedOn;
    public void SomeoneTurnsOnTheLight() {
        OnLightTurnedOn(myID);
    } // ...
}
```

.NET-Types that are not available in Java (cont'd)

Enumerations (value type):

```
enum Color : byte { RED = 1, BLUE = 2, GREEN = 3 };
```

Jagged (arrays of arrays) and unjagged Arrays:

```
int [2][] a; a[0] = new int[]{1}; a[1] = new int[]{1,2};
int [,] a = new int[2,2];
```

Structs (value types):

- Structs are implicitly sealed:

```
public struct Name {
    public String First;
    public String Last;
}
```

Commonalities and Differences

Commonalities:

- Interfaces are „completely abstract classes“
- Both support single inheritance for classes (implementation inheritance) and multiple interface inheritance
- Default-Initialization of variables
- Namespace-Concept (Java-Package and .NET-Namespace)
- Similar visibility attributes (public, private, ...)
- Generic types in .NET and Java (Generics) but .NET Generics much more flexible and available as runtime entities
- Class constructors (static initializers in Java)

Differences:

- In .NET there is no difference between primitive types and classes. Boxing to map between value and reference types. Autoboxing now also available in Java.
- Multiple language support in .NET
- In Java all methods are implicitly virtual. In .NET this is explicitly specified (virtual, override, new).
- Java maps packages to directories. .NET doesn't.

Metainformation

Java and .NET provide a reflection API

- to load, execute, and instantiate classes
- and inspect classes (introspection).

In addition, .NET allows to annotate many aspects of a system (classes, members, operations) with *Attributes*

Java Example

Accessing and using Type information:

- Note that packages and namespaces are different issues!!

```
import java.lang.reflect.*;

try {
    Class c = Class.forName(„MyPrintComponent“);
    Object o = c.newInstance();
    Method m = c.getMethod(„print“, new Class[]{ String.class });
    m.invoke(o, new Object[]{„Hallo, Java!“});
} catch (Exception e) {
    // handle it here
}
```

.NET Examples

Using an Attribute

```
[Author („Michael “)]
class MyClass { ... }
```

- There are several predefined attributes (WebService, WebMethod, ...)

Defining an Attribute

```
[AttributeUsage(AttributeTargets.All)]
public class AuthorIsAttribute : Attribute {
    private string m_Name;
    public AuthorIsAttribute(string name) { m_Name = name; }
}
```

.NET Example (cont'd)

Accessing and using Type information

```
using System;
using System.Reflection;

namespace ComponentClient {
    class Client {
        static void Main(string[] args) {
            Assembly a = Assembly.LoadFrom("Component.dll");
            Type[] allTypes = a.GetTypes();
            Type t = allTypes[0];
            object o = Activator.CreateInstance(t);
            MethodInfo mi = t.GetMethod("algorithm");
            double d = (double)mi.Invoke(o, new object[]{21.0});
        }
    }
}
```

Commonalities and Differences

Commonalities:

- Very similar APIs

Differences:

- .NET allows additional, user-defined meta information with attributes. This will also be possible in Java using a similar approach.
- Java Reflection is sometimes a bit more clumsy (because of primitive types and classes)
- .NET allows to actually create new artifacts at runtime and instantiate them or store them in assemblies.

Statements

Both platforms support basically the same language features

Some Differences:

- switch-Statement allows Strings, and prevents fallthrough.
- Different iterators: foreach(etc) to iterate container in .NET versus for (ElementType: ContainerType) in Java.
- Indexers, Operators not available in Java.
- Properties in Java are handled by coding conventions (set, get).

Exceptions

Exceptions in Java

- Exceptions have to be declared in the throws-clause. Checked versus unchecked exceptions

```
public int insert(int i) throws OverLimitException;
{ ... }
```

Exceptions in .NET

- Exceptions are not declared

```
// only way to tell you about
// OverLimitException thrown below
public int insert(int i) { ... }
```

Multithreading



Multithreading in Java

In Java there is a class „Thread“ and an interface Runnable

For synchronisation we have to use the keyword *synchronized*

```
class GlobalData {
    int m_Value;
    public synchronized int setValue { return m_Value; }
}
class Worker implements Runnable {
    GlobalData m_Global;
    public Worker(GlobalData global) { m_Global = global; }
    public void run() { m_Global.setValue(42); Thread.sleep(100); }
}
// somewhere else:
GlobalData g = new GlobalData();
Thread t = new Thread(new Worker());
t.start(); t.join();
1
```

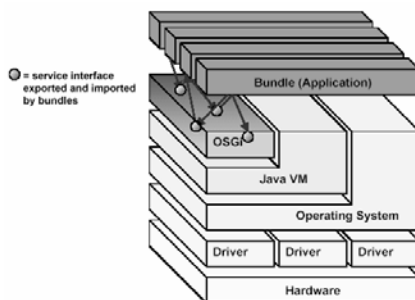

Server Components in Java cont'd

- **4 Types of Beans**
 - Stateless Session-Beans (Service Components)
 - Stateful Session Beans (Session Components)
 - Entity-Beans (Entity Components)
 - Message-Driven Beans (asynch. Service Components)
- **An EJB bean is in theory portable across containers (Application Servers) from different vendors.**

OSGi

- **Additional Frameworks for small devices are available such as OSGi, Pico, or Spring.**
- **OSGi Framework:** a general-purpose, secure, managed Java execution environment that supports the deployment of extensible and downloadable service applications called bundles.
- **Bundle:** a software component plugged into the framework that provides some functionality to an end-device such as a gas meter.
- **Service:** a Java object implementing a concisely defined interface. Installed bundles can register a number of services that can be shared with other bundles under strict control of the framework.

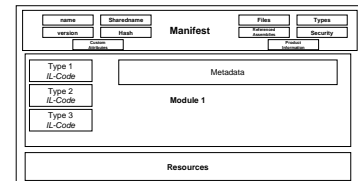
OSGi Architecture



.NET

- **Component=Assembly=Set of Types**

- Note that this notion of a component is **very** different from that used with EJB or COM+

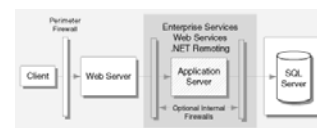


Component Model in .NET

- **Private Assemblies are typically only useful by the owning application**
- **Shared Assemblies are stored in a global assembly cache and can be used by several applications.**
 - They are signed by a key
 - They are versioned!!
- **Runtime uses Application Domains as an abstraction for isolated apartments within a process.**

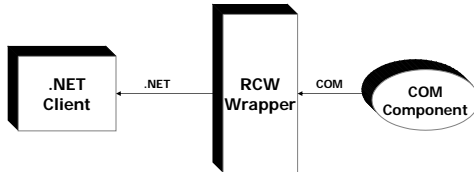
Server-Side Components

- **Now *Component* is used like in EJB**
- **To use container-provided services like synchronisation or transactions COM+ services can be used**
- **COM+-Interop provides these features.**



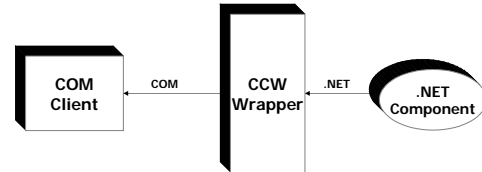
Using COM Components in .NET

- For a COM.DLL it is possible to create a RCW (Runtime Callable Wrapper).
- This is a wrapper to access COM components (with typelibs) from .NET:



COM Callable Wrapper

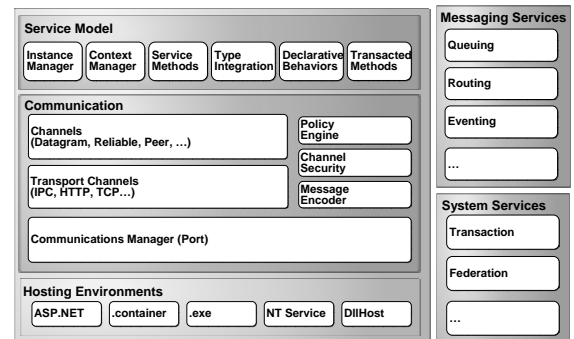
- .NET creates a CCW for exporting .NET classes as COM servers:



Future of Distributed .NET: Indigo

- Indigo (part of Longhorn) provides common framework for connected systems (SOA)**
 - Integrates Web services, .NET Remoting, MSMQ
 - Extensible Architecture
 - Vertical functionality such as security integrated in Indigo
- Service layer completely decoupled from protocol layer.
- For more details browse to <http://msdn.microsoft.com/Longhorn/understanding/pillars/Indigo/default.aspx>

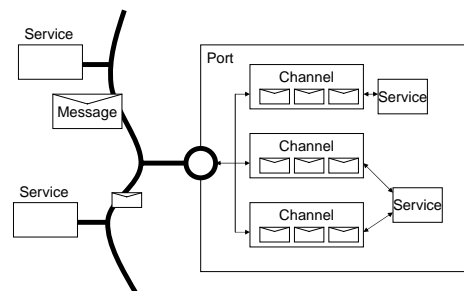
Indigo Architecture



Indigo Connectors

- A Connector is based upon 4 entities:**
 - Messages are in-memory envelopes (SOAP)
 - Services are the targets of messages
 - Ports are representations of network addresses
 - Channels allow sending or receiving through ports
- Applications rely on these concepts directly or indirectly

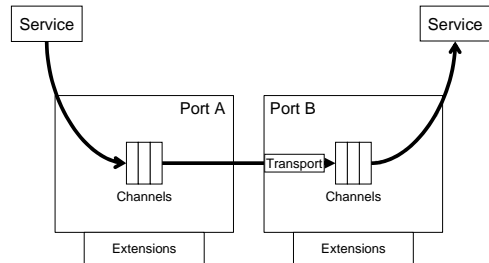
Ports, Channels, Services, And Messages



Ports And Channels

- **A port is a location in network space**
 - Resides in a single AppDomain
 - Has one or more affiliated transport channels
 - Provides a base URI for all addresses
- **Messages flow through a port via channels**
 - Channels impose a message exchange pattern
 - Channels may add additional processing code

Port/Channel Architecture



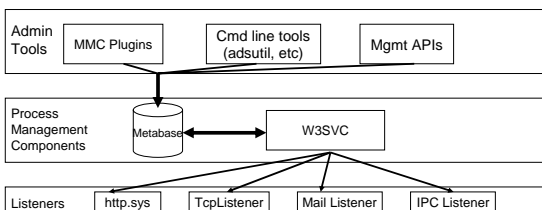
Services And Addresses

- **ServiceReferences are used to identify message recipients**
 - Absolute URI of service + fixed headers
 - Fully interoperable
 - ServiceAddress is relative version
- **“Indigo” uses the IService interface to model message recipients**
 - Maps ServiceAddress to in-memory object

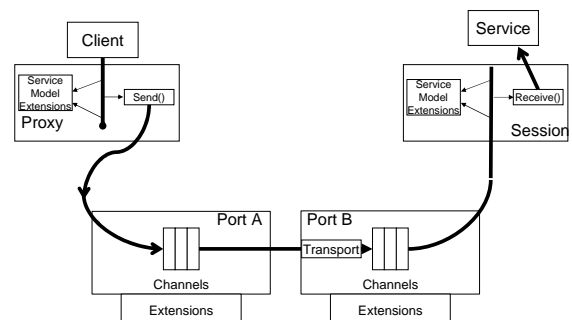
Hosting

- **Services can be self-hosted or activated on demand via ASP.NET**
- **“Indigo” shares activation with ASP.NET**
 - Process/AppDomain Startup/Shutdown/Cycle
 - Health monitoring
 - Management
- **URI namespace partitioning relies on metabase**
 - Independent of transport (HTTP, TCP, etc.)

On-Demand Activation



Services In Action



[Service]

- Service-Oriented Programming Model
- Opt-In Contract
- Schema-based integration
- Broad interop

```
[Service]
public class Hello
{
    [ServiceMethod]
    private string Greeting(string name)
    {
        return String.Format("Hello, {0}!", name);
    }

    public string Salutation (string name)
    {
        return String.Format("Howdy, {0}!", name);
    }
}
```

[RemoteObject]

- Object-Oriented Programming Model
- Contract based on public visibility
- DLL-based integration
- No Interop
- Like .NET Remoting, CLR-focused

```
[RemoteObject]
public class RemObj : MarshalByRefObject
{
    public RemObj() {
        Console.WriteLine(
            "Object (0) has been created.",
            this.GetHashCode().ToString());
    }

    public string Hello(string name) {
        return
            String.Format("Hello, {0}!", name);
    }

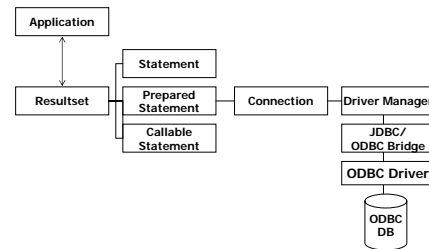
    ~RemObj() {
        Console.WriteLine(
            "Object (0) is being torn down.",
            this.GetHashCode().ToString());
    }
}
```

Commonalities and Differences

- **Commonalities:**
 - Assemblies and JAR files provide „deployment“ components
 - Server Components are available (Assemblies + COM+, EJB). Available component types in COM+ restricted
 - Interop with legacy components in .NET using COM+, in Java using CORBA)
- **Differences:**
 - EJB are a more mature and proven model
 - Special APIs in J2EE to connect to legacy systems (Java Connector Architecture)
 - Much better versioning support in .NET (side-by-side execution)
 - Future Outlook: Indigo will provide SOA container, EJB 3.0 will introduce the POJO approach to simplify development

Database Access in Java

- Java provides JDBC to access relational data



Java Example

```
import java.sql.*;
// without error handling;
Class.forName(„sun.jdbc.odbc.JdbcOdbcDriver“);
Connection
    con=DriverManager.getConnection(„jdbc:odbc:stocks,“,““);
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery(„SELECT * from stocks“);
while (rs.next()) {
    System.out.println(rs.getString(„COMPANYNAME“));
}
rs.close();
stmt.close();
con.close();
```

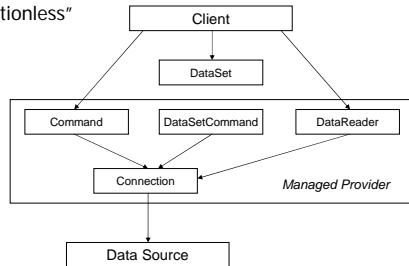
Database Access in Java

- **There are several other APIs:**
 - Embedded SQL with SQLJ (uses JDBC internally)
 - Proprietary ODBMS APIs
 - Standardized JDO API to provide (more or less transparent) persistence for Java Objects
 - XML is handled differently!
 - Java Connector API provides access to other „connection oriented“ legacy systems (such as SAP R3)

Database Access in .NET

▪ In .NET there is ADO.NET

- "connectionless"



ADO.NET

▪ ADO.NET is XML based (XML Infoset)

- DataSet dynamically builds an XML schema inside to store the data
- Relational data and XML data can be handled in a similar way!!

▪ ADO.NET works offline once the data is fetched

- Updating is partly automatic using DataSets

▪ Currently there are three Managed Providers:

- SQL Server
- ADO
- Oracle

Commonalities and Differences

▪ Commonalities:

- Decoupling of the concrete data model and the user (using DataSets and ResultSets)

▪ Differences:

- ADO.NET uses XML extensively, JDBC has a more relational flavor (like ADO)
- JDBC is mainly connection oriented, ADO.NET always works non-connected, or offline
- .NET DataSets are a kind of In-Memory-Database-Cache.
- In Java additional O/R solutions such as JDO or SQLJ without .NET counterpart.

XML



XML und Java

▪ There are several tools available

- DOM, SAX
- Xerces/Xalan, JDOM
- JAX{M,B,R,RPC}
- Castor

▪ However, Java's libraries have not been designed with XML as a basis (Java's too old ☹)

▪ JAXP (Java API for XML Parsing) supports DOM and SAX).

▪ Currently under development

- JAXM (Java API for XML Messaging)
- JAXB (Java API for XML Data Binding)
- JAXR (Java API for XML Registries)
- JAX/RPC (Java API for XML based RPC)

XML and .NET

▪ .NET is very XML-centric

- Web Services (SOAP)
- Configuration Files
- Result sets of a database access (ADO.NET)
- XML processing itself

▪ Note that formally, many .NET features are based on the XML infoset („XML semantics“) and do not necessarily require megabytes of text data!!

XML and .NET (cont'd)

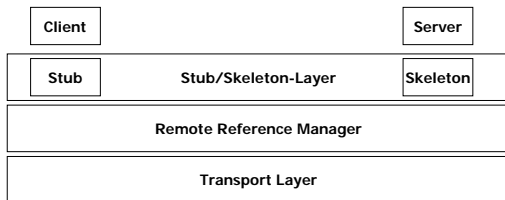
- The **System.Xml Namespace** provides a whole lot of classes
 - DOM processing using Xml Node & Sons
 - XPath and Xsl Transform
 - Xml TextReader und Xml TextWriter similar to SAX, but using a Pull-Modell (Stream-Based)
 - Schema support
- **Two approaches: one that is closer to the programmer's object model and another one that is closer to the DOM.**

Remoting



Remoting in Java

- **Several possibilities: RMI/CORBA/SOAP**
 - RMI can use JRMP or IIOP as a transport protocol
 - Not easily pluggable – changes in the code are necessary



Some Facts about RMI

- Registry tool as naming service, started manually or automatically.
- On demand activation possible.
- Changing lease time for DGC possible (leasing time = time without connection).
- Usage of RMI Security Manager to allow class loading across computer boundaries.
- Object by value out-of-the-box.
- Other protocols by implementing proprietary Socket factories.

Example

- **Exceptions and interface definitions**

```
// file: BoundaryException.java
public class BoundaryException extends Exception {}
// file: Grid.java
import java.rmi.*;

public interface Grid extends Remote {
    public void setValue(int row, int col, Object val)
        throws RemoteException, BoundaryException;
    public Object getValue(int row, int col)
        throws RemoteException, BoundaryException;
    public int getColumns()
        throws RemoteException;
    public int getRows()
        throws RemoteException;
}
```

Example (cont'd)

- **Implementation of remote object:**

```
import java.rmi.*;
import java.rmi.server.*;

public class GridImpl extends UnicastRemoteObject implements Grid {
    Object[][] values; final
    static int ROWS = 20; final static int COLS = 20;
    public GridImpl() throws RemoteException {
        values = new Object[ROWS][COLS];
    }
    public boolean isOutOfBounds(int r, int c) {
        if ((r < 0) || (r >= ROWS) || (c < 0) || (c >= COLS))
            return true;
        else return false;
    }
    public Object getValue(int row, int col)
        throws RemoteException, BoundaryException {
        if (isOutOfBounds(row, col)) throw new BoundaryException();
        else return values[row][col];
    }
    ...
}
```

Example (cont'd)

▪ The Server Main

```
import java.rmi.*;
import java.rmi.server.*;

public class GridServer {
    public static void main(String args[]) throws
    Exception {
        GridImpl svr = new GridImpl ();
        Naming.rebind("grid", svr);
        System.out.println("ready");
    }
}
```

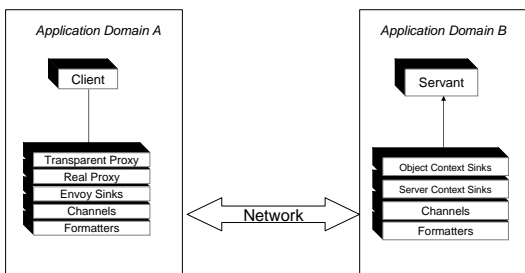
Example (cont'd)

▪ Client

```
import java.rmi.*;
import java.rmi.registry.*;

public class GridClient {
    public static void main(String args[]) {
        try {
            Grid g = (Grid)Naming.lookup("grid");
            g.setValue(4, 4, new Integer(42));
            Integer i = (Integer)g.getValue(4, 4);
            System.out.println(i);
            g.setValue(33, 33, new Integer(88));
        }
        catch (BoundaryException b) {
            System.out.println("Boundary violation");
        }
        catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

Remoting in .NET



Remoting in .NET (cont'd)

- **.NET Remoting provides pluggable transports and formatters**
 - currently TCP and HTTP transport and
 - binary and SOAP formatters
- **Contexts are automatically propagated (very neat feature!!)**
- **Only very simple lifecycle management options for servants (compared to EJB or CORBA)**
 - Singleton (one object for all calls)
 - SingleCall (new instance for each call)
 - Client-Activated based on leases

Commonalities and Differences

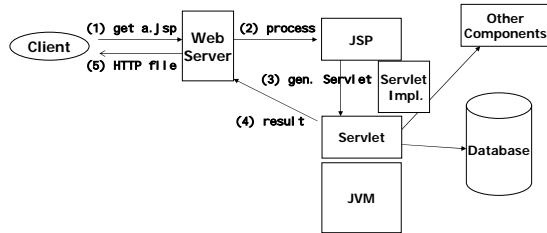
- **Commonalities:**
 - Relatively easy to use
- **Differences:**
 - .NET Remoting can be extended more flexibly
 - Java provides interoperability with CORBA
 - Asynchronous invocations not directly supported by Java
 - No activation mechanism provided in .NET Remoting

Web



Java Server Pages and Servlets

- Java allows for server-side scripting
 - JSPs are based on Servlets



Java Example



Java Example

- Bean and JSP:

```
// Datei MyPerson.java
package MyPackage;
import java.lang.*;

public class MyPerson {
    public String getFirstName() { return "Michael"; }
    public String getLastName() { return "Stal"; }
}

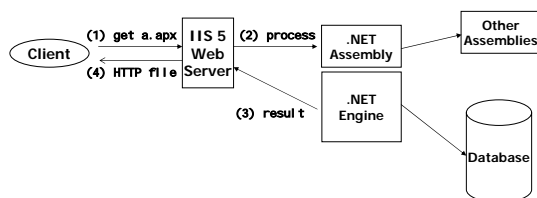
// Datei MyTest.jsp:
<HTML> <BODY>
<jsp:useBean id="person" scope="session"
class="MyPackage.MyPerson"/>
Your name is: <br>
<jsp:getProperty name="person" property="firstName"/> <br>
<jsp:getProperty name="person" property="lastName"/>
</BODY> </HTML>
```

Complementary Technologies in Java

- Servlets are server-side extensions providing functionality. Implemented by Servlet Engine.
- JSPs (Java Server Pages) are scripted Web pages transformed to servlets.
- Taglibs allow to integrate additional html/xml-like tags.
- Java ServerFaces use taglibs to provide Web components.
- Apache Struts provides a framework for implementing MVC applications.

ASP.NET (Server-Side Scripting)

- ASP.NET Architecture:



ASP.NET Example

- A simple login screen:



ASP.NET Example (cont'd)

```
<%@ Page language="c#" Codebehind="WebForm1.aspx.cs"
AutoEventWireup="false" Inherits="LoginPage.WebForm1" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
<HTML>
<body>
<form id="Form1" method="post" runat="server">
<asp:Label id="TitleLabel" runat="server">Please specify your name and
password</asp:Label> <br>
<asp:Label id="LoginLabel" runat="server">Login</asp:Label> <br>
<asp:TextBox id="LoginText" runat="server"></asp:TextBox>
<asp:RequiredFieldValidator id="RequiredFieldValidator" runat="server"
ErrorMessage="You need to specify your name"
ControlToValidate="LoginText"></asp:RequiredFieldValidator> <br>
<asp:Label id="PasswordLabel" runat="server">Password</asp:Label> <br>
<asp:TextBox id="PasswordText" runat="server"
TextMode="Password"></asp:TextBox> <br>
<asp:Button id="EnterButton" runat="server" Text="Open the entrance"
ToolTip="Press this after you have specified login and
password"></asp:Button> <br>
<asp:Label id="MessageText" runat="server"></asp:Label >
</form>
</body>
</HTML>
```

ASP.NET Example (cont'd)

```
// lot of details omitted
namespace LoginPage {
public class WebForm1 : System.Web.UI.Page {
protected TextBox PasswordText, LoginText;
protected Button EnterButton;
protected Label MessageLabel;
private void InitializeComponent() {
this.EnterButton.Click +=
new System.EventHandler(this.EnterButton_Click);
this.Load += new System.EventHandler(this.Page_Load);
}
private void EnterButton_Click(object sender, System.EventArgs e) {
if (!LoginText.Text.Equals("aladdin") &&
PasswordText.Text.Equals("sesam")) {
MessageLabel.Text = " Wrong name or password! ";
}
else {
Session["user"] = "aladdin";
Response.Redirect("UserArea.aspx");
}
}
}
}
```

Commonalities and Differences

Commonalities:

- Pages are precompiled to accelerate access
- Similar syntax and concepts
- ASP.NET provides „GUI components“ using Webcontrols, Java provides Taglibs.

Differences:

- All .NET languages can be used for ASP.NET scripting
- Servlets/JSP are available for a wide range of web servers
- Many open source implementations, frameworks and tools for Java

XML Web Services

- .. why today's Web does not solve all the problems.

Why do we care about Service-Oriented Architectures and Web Services

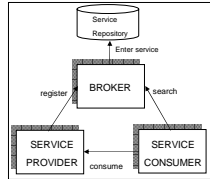
- **Cannot take single-vendor approach to IT (because of risk, pricing, etc.)**
 - Embracing heterogeneity
- **Interoperability is the only viable and economic approach**
- **Accelerating change in IT due to globalization, e-business, etc.**
- **Service-Oriented Architectures/Web Services have ubiquitous support of all major vendors**
- **Standardization**

Services

- **What is a service?**
 - A remotely accessible self-contained piece of coarse-grained software functionality with at least one unique physical address to which a service consumer can bind using a communication protocol supported.
 - A service exports functionality using standardized service interfaces. There might be multiple service interfaces denoting different types of business functionality or different types of communication protocols.
 - Services are implementation agnostic in that the implementation technology used (container, programming language, middleware, underlying server infrastructure) is not made visible.
 - Services are singletons. In contrast to CORBA, DCOM, RMI there are no instances.
 - Services are defined using a high level description language (XML)

Service-Oriented Architectures

- Basic concepts:
 - Self-contained services.
 - Messages exchanged between providers and consumers.
 - Brokers/Mediators where services are registered and located.
- A Broker in this scenario isn't necessarily an ORB! It might be a directory service or a P2P discovery mechanism.
- Components/Containers are often used for the implementation of services!



Loosely Coupled

- The central paradigm when using Service-oriented Architecture as a principle and paradigm is loose coupling.
- Loosely Coupled means (see Doug Kaye's book „Loosely Coupled – The Missing Pieces of Web Services“):
 - All interactions are basically asynchronous on the transport level. Higher abstraction layers might be introduced to support synchronous communication.
 - Messaging Style is Document, not RPC.
 - Message Paths are routed, not hard-coded.
 - Technology must support heterogeneity (interface is standardized, not code).
 - Data Types are technology independent.
 - Syntactic Definition using a schema. Contract defines structure (what documents are exchanged) and behavior (e.g., order and rules of document exchange, QoS).
 - Binding is delayed, not fixed and early.
 - Semantic Adaptation by transformation, not re-coding (e.g. use intermediaries for currency calculations).
 - Objective: More on broad applicability than on re-use or efficiency.

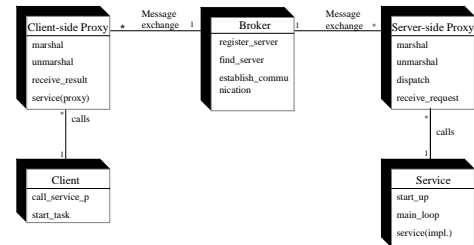
SOA: Message Passing Paradigm

- Under the hood each communication in a SOA is based upon asynchronous message exchanges:

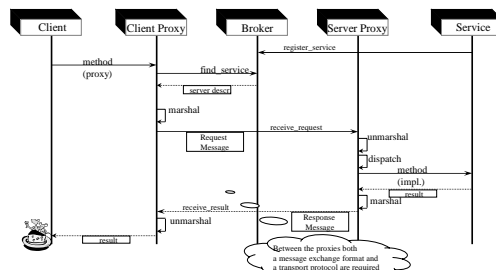


- Any other communication style / semantics can be provided using message passing, e.g.:
 - Request/Response (use correlation info to associate messages)
 - Multicast: Send same message to different peers
 - ...

Web Services as Middleware Solution: The Web is the Computer



Dynamics

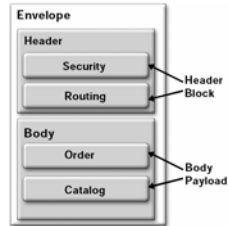


How to implement this paradigm



Step 1: Define a Transport Protocol

- A protocol defines syntax, semantics and order of messages exchanged between peers.
- For a Web-based transport protocol we need:
 - HTTP and other Internet Protocols as transport layer.
 - a self-describing data representation format using XML.
- Each request and each response is wrapped as a XML message and transferred over the wire.
 - SOAP formerly known as Simple Object Access Protocol.



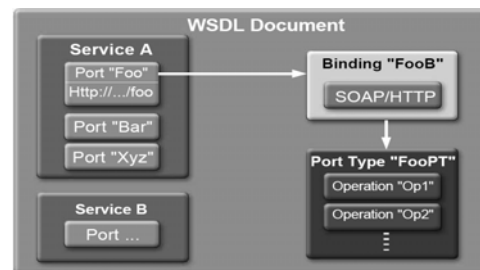
Marshalling Flavors

- SOAP bodies can be formatted
 - **RPC style:** method name and parameter names are mapped to XML elements with same names
 - **Document Style:** XSD Schema is used instead. Web service can carry any document.
- Options for parameter encoding:
 - **Literal:** Encoding using XML Schema
 - **Encoded:** Encoding using SOAP
- Conforming to the standard toolkits should use Document Style and Literal

Why is SOAP important?

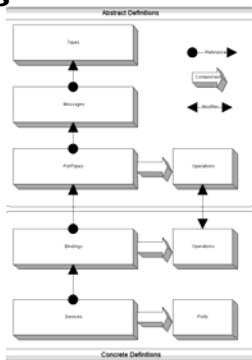
- SOAP defines a transport protocol based upon XML and HTTP (or other protocols).
- Leveraging HTTP helps clients and servers to circumvent firewalls.
- Using SOAP, web servers can be extended to full blown Broker architectures.
- SOAP can bridge disparate infrastructures and applications.

Step 2: Define a Service Description Language



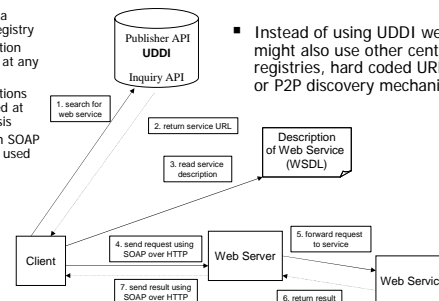
WSDL Elements

- **Port**
 - Concrete net address of Web Service (incl. URL and port)
- **Service**
 - Collection of ports
 - Physical location of end point
- **Message**
 - Format for single transfer
 - Request and Reponse are separate messages
- **PortType**
 - Logical grouping of messages to operations
- **Binding**
 - Maps PortType to implementation (e.g., SOAP)



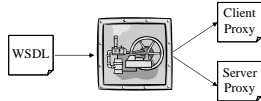
Step 3: Define a global Service Directory

- UDDI is a global registry
- Registration possible at any node
- Registrations replicated at daily basis
- Common SOAP protocol used
- Instead of using UDDI we might also use other central registries, hard coded URLs, or P2P discovery mechanisms

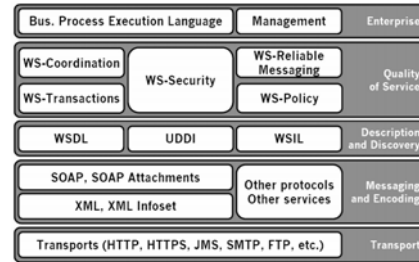


Step 4: All you need are Generators

- Introduce tools that generate glue for connecting you with the Web ORB, i.e.,
 - Client Proxies for connecting the client with services.
 - Server Proxies for seamless service deployment.



SOAP + WSDL + UDDI is not sufficient



WS Implementations and Complementary Technologies

- **Tools:**
 - .NET: ASP.NET WebServices (will be integrated in Indigo), WSE (Web Services Enhancements) includes Attachments, Security, ...
 - J2EE: JAX-RPC which is implemented by many vendors such as Apache Axis, WS is part of EJB 2.1 (Enterprise JavaBeans)
 - MS-SOAP for COM
 - All major IDEs support WS such as IBM, Microsoft, BEA, Borland, ...
- **Complementary Technologies:**
 - P2P (Peer-to-Peer Computing): Use discovery instead of central repository
 - Grid Computing: Sharing common resources on a network

Web Services in Java

- **Java (J2EE 1.4) provides a Web Service API for Java: JAX-RPC is the**
- **Currently there are different solutions**
 - Some specific SOAP toolkits:
 - Apache Axis
 - IBM Web Services Toolkit
 - GLUE
 - Some integrated with the major application servers
 - Silverstream
 - IONA
 - Weblogic
 - ...

Case Study: Web Services in Java using GLUE

GLUE

- There are a lot of different solutions for Java.
- One typical example is GLUE from „The MIND Electric“.
- GLUE allows to implement Web services in Java.
- It provide its own little HTTP server and can integrate EJB, JMS, ...

GLUE (cont'd)

- It also enables to parse WSDL files and generate proxies.
- Interoperates with most vendors such as .NET or IBM WSTK.
- Offers both command line tools and JBuilder support.

GLUE Example (1)

- Provide an interface `IHello.java`:

```
import java.*;

public interface IHello {
    public String echo(String input);
    public String hello();
}
```

GLUE Example (2)

- Implement the interface in `Hello.java`:

```
import java.*;

public class Hello implements IHello {
    public String hello() {
        return "Hello, here is GLUE";
    }
    public String echo(String input) {
        return input;
    }
}
```

GLUE Example (3)

- Implement main (`HelloMain.java`):

```
import electric.registry.Registry;
import electric.server.http.HTTP;
public class HelloMain {
    public static void main( String[] args ) // throws Exception
    {
        try {
            // start a web server on port 8004
            HTTP.startup( "http://localhost:8004/glue" );
            // publish an instance of Exchange
            Registry.publish( "hello", new Hello() );
        }
        catch(Exception e) {
            e.printStackTrace();
        }
    }
}
```

GLUE Example (4)

- Now start server
- Using a web service is also simple:
 - Use `wsdl2java` to generate a proxy implementation.
 - Instantiate and invoke it from your client application.
 - GLUE also provides tools such as `java2wsdl`.

GLUE Example (5)

```
// calling it:
IHello svc = new IHelloHelper().bind();
String res = svc.echo("Hello");
```

Web Services in .NET

- .NET provides a very comfortable and well-integrated way to build them:

```
namespace WebService1 {
    public class Service1 : System.Web.Services.WebService {
        // lot of stuff omitted
        [WebMethod]
        public double DM_to_Euro(double value) {
            return value / 1.95583;
        }
        [WebMethod]
        public double Euro_to_DM(double value) {
            return value * 1.95583;
        }
    }
}
```

Datatypes supported

String	Char	Byte
Boolean	Int16	Int32
Int64	UInt16	UInt32
UInt64	Single	Double
Guid	Decimal	DateTime
XMLQualifiedName	class	struct
XmlNode	DataSet	

Using Web Services

- Using it is also simple
 - Some steps have been omitted

```
localhost.Service1 s1 = new localhost.Service1();
double result = s1.Euro_to_DM(200);
```

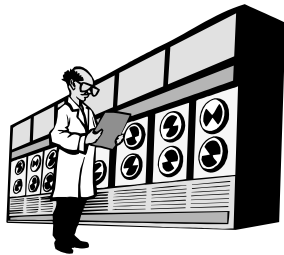
- Asynchronous „Style“:

```
localhost.Service1 s1 = new localhost.Service1();
IAsyncResult ar = s1.BeginEuro_to_DM(200, null, null);
while (!ar.IsCompleted) {
    // enjoy your coffee
}
double result = s1.EndEuro_to_DM(ar);
```

Commonalities and Differences

- **Commonalities:**
 - Both .NET and Java strive for conformance to existing standards (SOAP, WSDL, UDDI).
 - „Look & Feel“ very similar: WSDL-based parsers that generate proxies.
- **Differences:**
 - For Java there is a whole bunch of available solutions.
- **Additional remark:**
 - Standards are open to interpretation. Thus, interoperability doesn't come for free.

More Enterprise APIs



Enterprise APIs

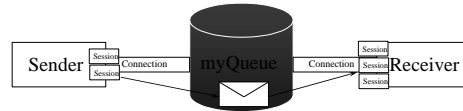
- **Naming:**
 - JNDI in Java (as an interface to CORBANaming, LDAP, ...)
 - Active Directory in .NET (Windows-specific)
- **Message-oriented Middleware:**
 - JMS in Java (JAXM is on the horizon)
 - .NET can use MSMQ.

JMS – Messaging in Java

- JMS is an API to integrate existing MOM solutions (JMS Providers).
- Point to Point approaches are supported as well as Publish/Subscribe (Message Listeners).
- Queues (~ mailbox) and Topics (~ newsgroup).
- Different kinds of message bodies are provided: stream, text, serialized object, bytes, map.

Example

- Here is an example scenario for Queuing:



Example (cont'd)

- Here is a sample peer-to-peer client:

```
// First, we need a Connection Factory:
QueueConnectionFactory factory;
Context ctx = new InitialContext(); // JNDI lookup
factory = (QueueConnectionFactory) ctx.lookup("QCFactory");
// Getting a message queue:
myQueue = (Queue) ctx.lookup("MyQueue");
// Connect:
QueueConnection con;
con = fac.createQueueConnection(); // create JMS connection
```

Example (cont'd)

```
... sample continued ...
// Establish a session:
QueueSession session
= con.createQueueSession(false, Session.AUTO_ACKNOWLEDGE);
// Get a QueueSender to send messages using <queue>:
QueueSender sender = session.createSender(queue);
// Get a QueueReceiver to receive messages using <queue>:
QueueReceiver receiver = session.createReceiver(queue);
// Creating a Text Message:
StringBuffer myText;
TextMessage message;
message = session.createTextMessage();
message.setText(myText);
```

Example (cont'd)

```
... sample continued ...
// Send the message:
sender.send(message);
// receiving response:
TextMessage newText;
newText = (TextMessage) receiver.receive();
```

- That's it!

MSMQ - Messaging in .NET

- There is also a messaging API to access MSMQ:

```
using System;
using System.Messaging;

namespace Messaging
{
    class Class1 {
        static void Main(string[] args) {
            MessageQueue mq = null;
            if (!MessageQueue.Exists(@"\\.\Private\MyQueue1")) {
                mq = MessageQueue.Create(@"\\.\Private\MyQueue1");
            }
            else
                mq = new MessageQueue(@"\\.\Private\MyQueue1");

            mq.Send("Hello");
            Message msg = mq.Receive();
            Console.WriteLine(msg.Body.ToString());
        }
    }
}
```

Accessing Directory Services

- There is a special API in .NET to access LDAP-based directories such as Active Directory.
- In Java JNDI serves as an adapter to existing directory services.
- Both very similar.
- Java much more flexible.

Legacy-Integration

- In .NET using the Microsoft Host Integration Server
- In Java using the Connector Architecture. JCA defines standard way to build connectors and integrate them with EJB containers
- Comparison: Java provides a much simpler way. Connectors can be implemented relatively straight forward.

Interoperability

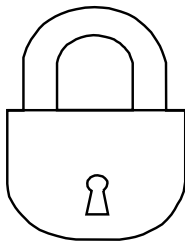
- Java provides access to C/C++ using JNI (Java Native Interface). Relatively complex call-in, call-out.
- .NET provides PInvoke (Platform Invoke):

```
class PInvokeTest {  
    [DllImport("user32.dll")]  
    static extern int MessageBoxA(int hWnd, string m, string c, int t);  
    static void Main(string[] args) {  
        MessageBoxA(0, "Hello DLL", "My Window", 0);  
    }  
}
```

Interoperability (cont'd)

- In .NET there is a way to interoperate with COM+ and COM+ Services.
- Java provides CORBA interoperability
- JMS provides integration with MoMs
- In .NET, the interoperability between .NET languages is easy
 - Use the same assemblies, class libraries...
 - Languages have to be adapted a little bit (e.g. Managed C++ does not provide multiple inheritance)

Controlling Resource Access



Access Control in Java

- In Java access control is implemented by the SecurityManager (delegates to java.security.AccessController).
- SecurityManager uses policy files. Use policytool to create policies.
- Permissions in policies control access to resources. Customized permission types possible.
- Jar-files can be signed (keytool).

Access Control in .NET

- **Code Access Security**
 - managed by CLR
 - Evidence-based security (works in tandem with Windows security).
 - Code Groups defined by membership conditions such as zone, publisher, URL, site, publisher, strong name
 - Tool caspol .exe (code access security tool) to administer code groups.
 - Code Access Permissions and Permissions Sets specifies what assemblies are allowed to do.
 - Policy Levels: User, Machine, Enterprise.

Access Control in .NET (cont'd)

- Support in .NET Framework:
 - System.Security.Permissions
 - System.Security.Policy
 - System.Security.Principal
- Permissions can be requested programmatically or declaratively.
- Configuration stored in XML-based .config files.
- **Role-Based Security**
 - Principal: user's identity can be Windows account, Passport account, ASP.NET cookie authentication
 - Roles according to technology used.

Commonalities and Differences

- **Both approaches very similar.**
- **Differences in implementation of access control:**
 - In .NET CLR collects evidence to grant or deny permissions.
 - In Java the SecurityManager is used by JVM to ask for permission.

Mobile and Embedded



Profiles and devices

- **There are the following variants of Java**
 - J2SE (Java 2 Platform Standard Edition)
 - J2EE (Java 2 Platform Enterprise Edition)
 - J2ME (Java 2 Platform Micro Edition)
 - Configurations,
 - and Profiles
 - JavaCard

Profiles and devices (cont'd)

- **.NET Universe**
 - .NET: Framework for Standard und Enterprise
 - .NET Compact Framework for Windows CE (and other embedded OS)
 - Design Goals:
 - > Resource saving
 - > Adaptability regarding device properties
 - > Compatibility with the standard IDEs
 - > Easy integration
 - > Seamless connectivity
 - No ASP.NET, Reflection.Emit, Optimized JIT
 - Depending on machine stack
 - Simplified versioning and security, but similar formats

Profiles and devices (cont'd)

- **.NET Mobile Web Framework**
 - Abstraction for the developer
 - Several markup languages (WML, HTML, ...)
 - Configurable and extendible
 - ASP.NET can be used
 - Emulators for devices are available for testing and debugging purposes
- Extends ASP.NET with special controls

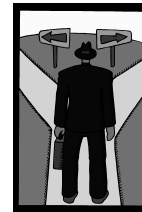
Tools

- **Microsoft .NET applications are mainly developed using Microsoft's Visual Studio .NET.**
 - Borland will also provide support (e.g., Sidewinder, a C# IDE)
 - Rational XDE support .NET
 - Language implementers such as ISE, ActiveState provide Visual Studio Plug Ins
- **Java tool support comes from many different companies and organizations:**
 - Sun, Borland, BEA, Rational, IBM, Eclipse, Compuware, ...

Open and Shared Source

- **Microsoft .NET:**
 - Shared Implementation (Rotor) available for BSD Unix and Windows.
 - Mono developed as open source to implement full .NET support for Linux; still immature
 - Dot GNU will provide another open source approach; when will it arrive?
- **Java:**
 - Countless mature open source products available
 - JBoss (EJB), OpenEJB, Eclipse, Apache (Tomcat, Struts, ...), ...

Selecting one of the two



.NET and/or Java ?

- .NET is a product, JVM/J2EE are specifications
- Both address the web (among other things)
- For „real big systems“ J2EE is better suited
- **The rule-of-thumb „Java is platform-independent, .NET is language independent“ must be considered carefully:**
 - ECMA works on the standardization of C# and parts of .NET. However, only subset standardized. JCP (Java Community Process) works very efficiently.
 - Other languages can be compiled to the JVM such as Jython, ComponentPascal, Beta, ...

.NET and/or Java ?

- **.NET language independence is not for free and not completely transparent**

```
#pragma once
using namespace System; // .NET mit C++
namespace CPPBase {
public __gc class CPPBaseCl ass {
public: virtual System::String __gc* Echo(System::String __gc *s);
};
System::String __gc * CPPBase::CPPBaseCl ass::Echo(System::String __gc *s)
{ return s; }
```

- **In a real project you might want to use only one language**
 - But sometimes... ☺

.NET and/or Java ?

- Windows Applications are better done with .NET than Java (can Eclipse SWT change this?)
- Java should be used when platform (vendor-) independence is necessary
- Java is more mature, .NET is more modern
- There is also Java for .NET
 - But syntax is not the issue here!
- Both can be used for Web services - .NET is „nicer“, J2EE is more scalable
- Migration from COM to .NET seems to be a big issue.
- Analysts foresee a fifty/fifty situation for Java and .NET

Noch Fragen?

**Development for
Professionals!**

Bemerkungen

Bemerkungen