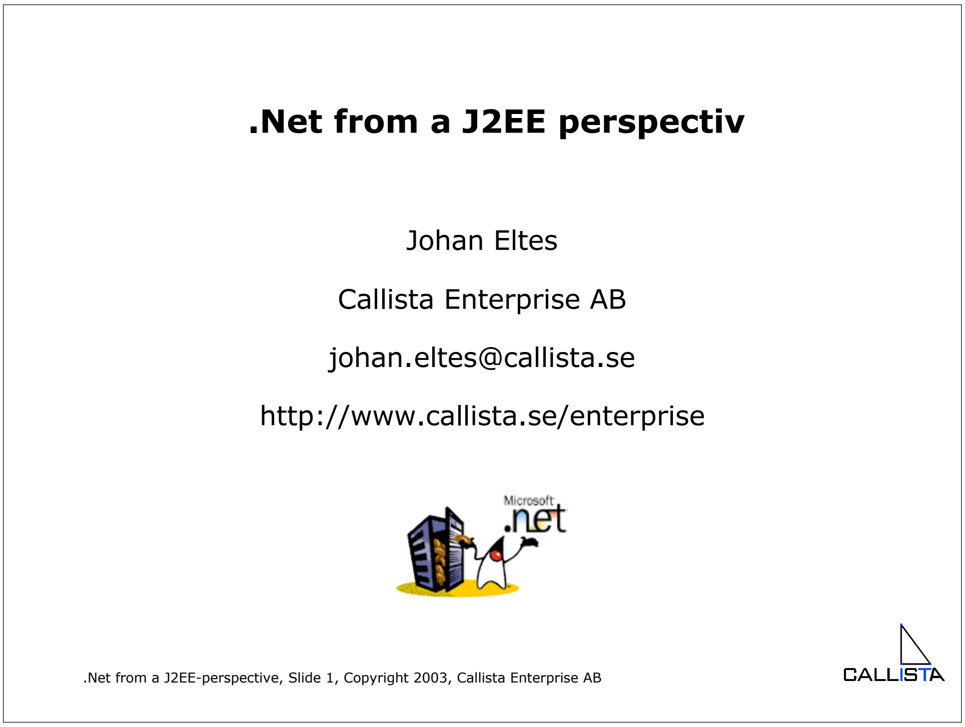


.Net from a J2EE perspectiv

Johan Eltes

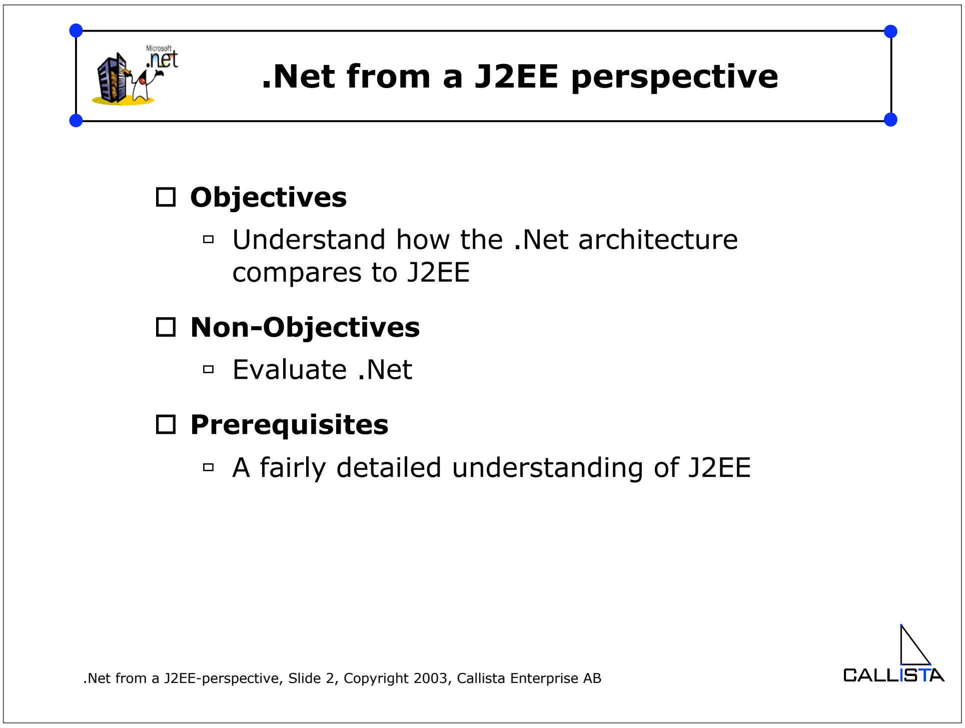
Callista Enterprise AB

johan.eltes@callista.se

http://www.callista.se/enterprise

Microsoft .net

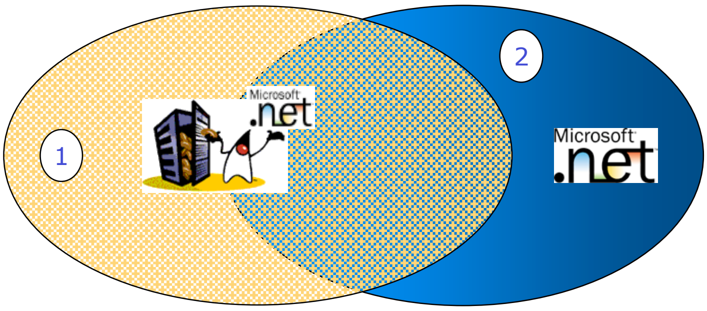
CALLISTA



.Net from a J2EE perspective

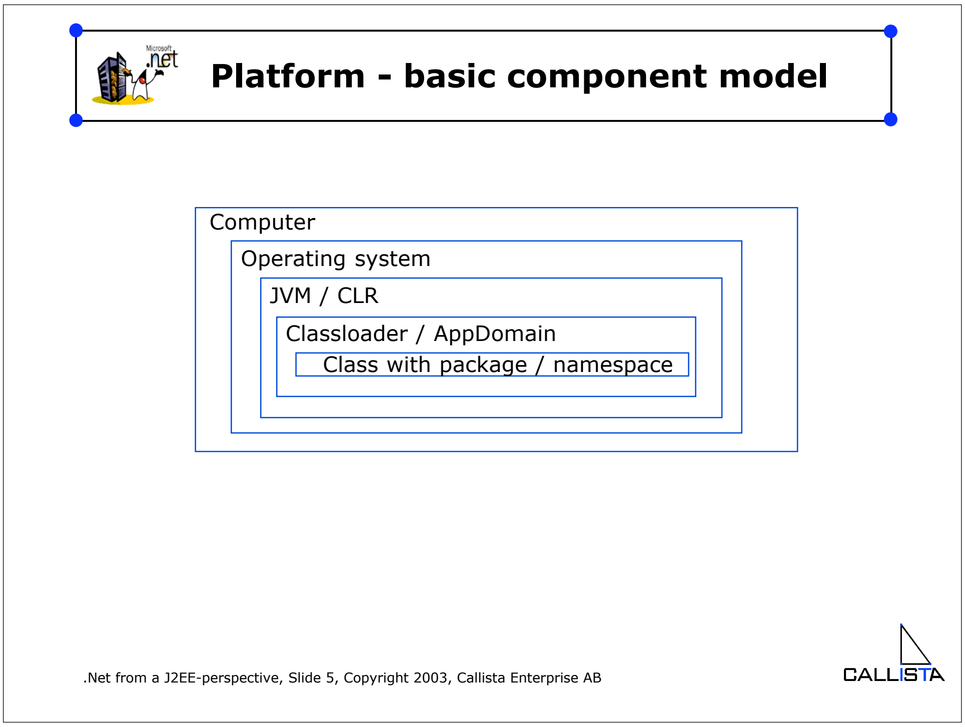
- □ **Objectives**
  - ▫ Understand how the .Net architecture compares to J2EE
- □ **Non-Objectives**
  - ▫ Evaluate .Net
- □ **Prerequisites**
  - ▫ A fairly detailed understanding of J2EE

How this presentation is organised



Agenda

- ☐ Platform, Languages
- ☐ J2EE features in .Net
- ☐ Unique enterprise .Net features
- ☐ J2EE and .Net Roadmaps
- ☐ And the winner is….



Platform - basic component model

Computer

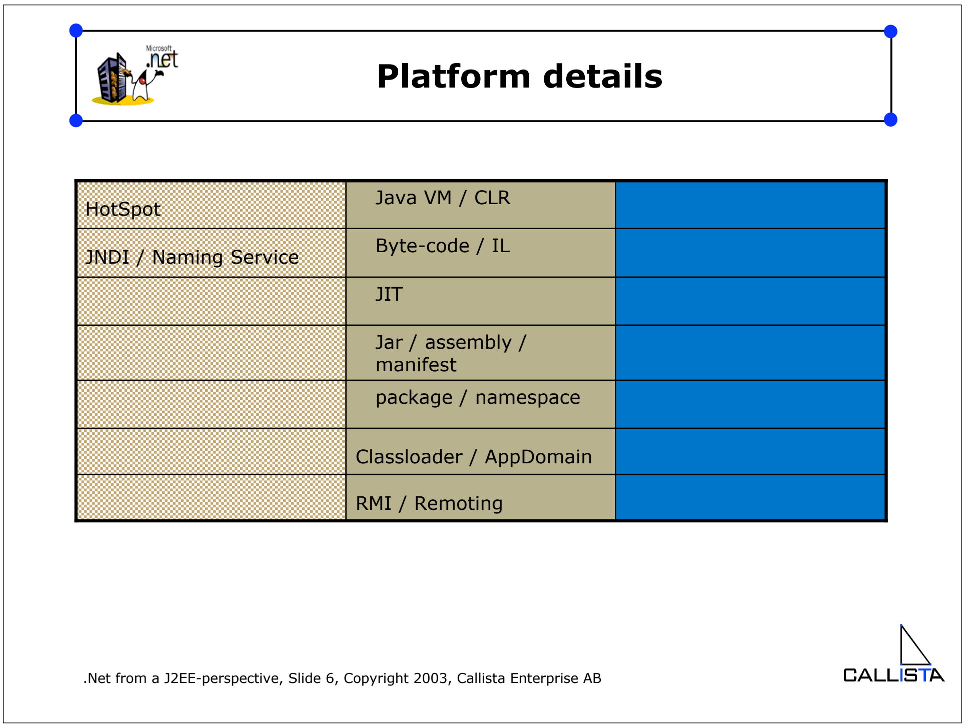
Operating system

JVM / CLR

Classloader / AppDomain

Class with package / namespace

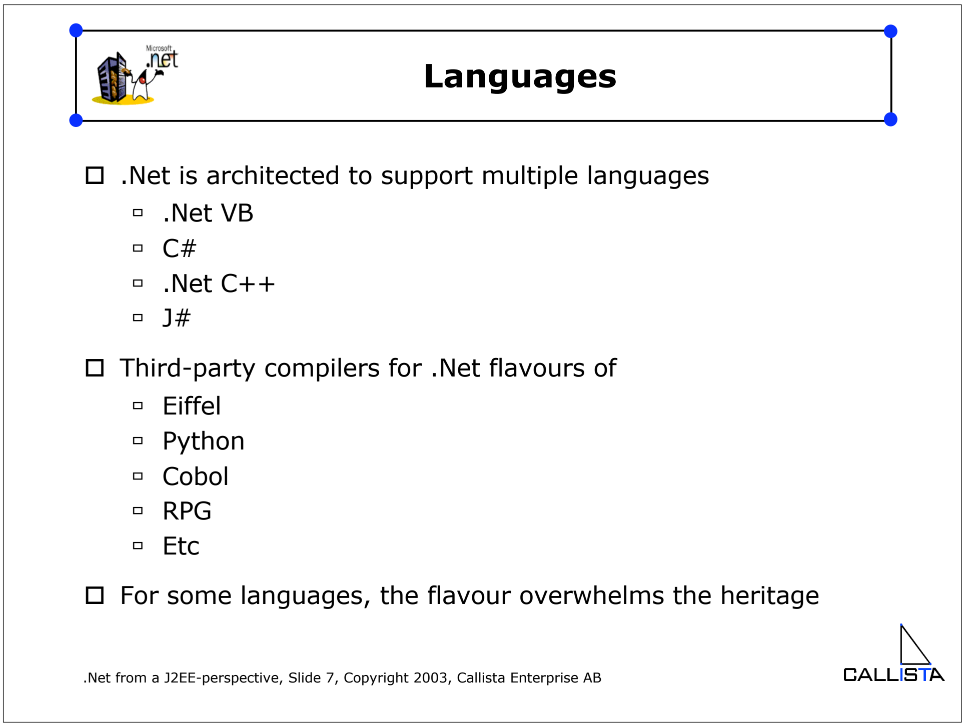
CALLISTA



Platform details

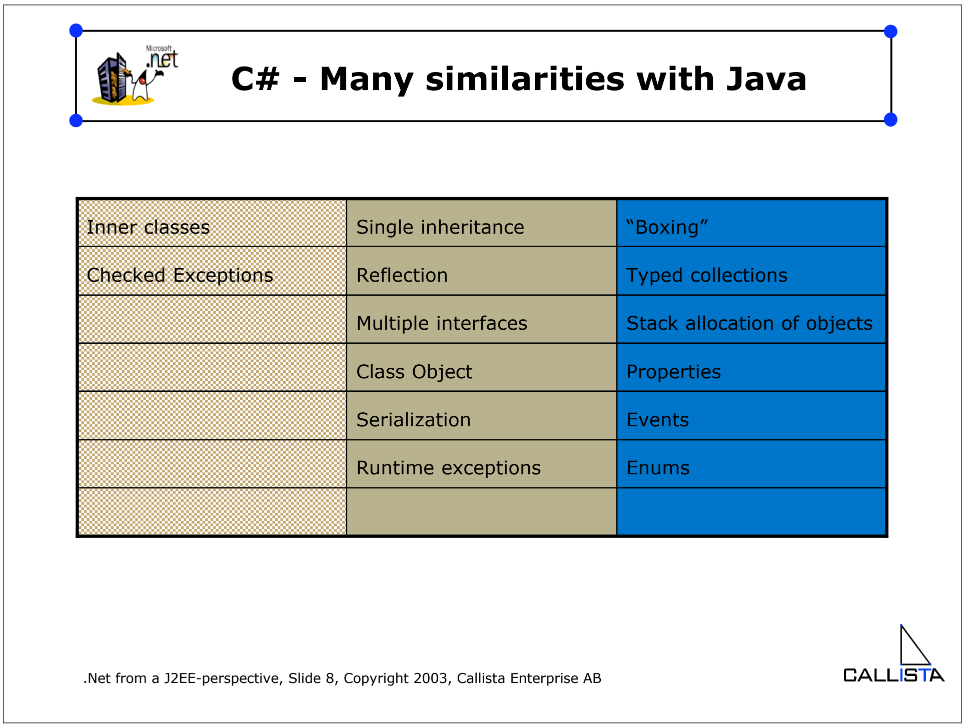
HotSpot	Java VM / CLR	
JNDI / Naming Service	Byte-code / IL	
	JIT	
	Jar / assembly / manifest	
	package / namespace	
	Classloader / AppDomain	
	RMI / Remoting	

CALLISTA



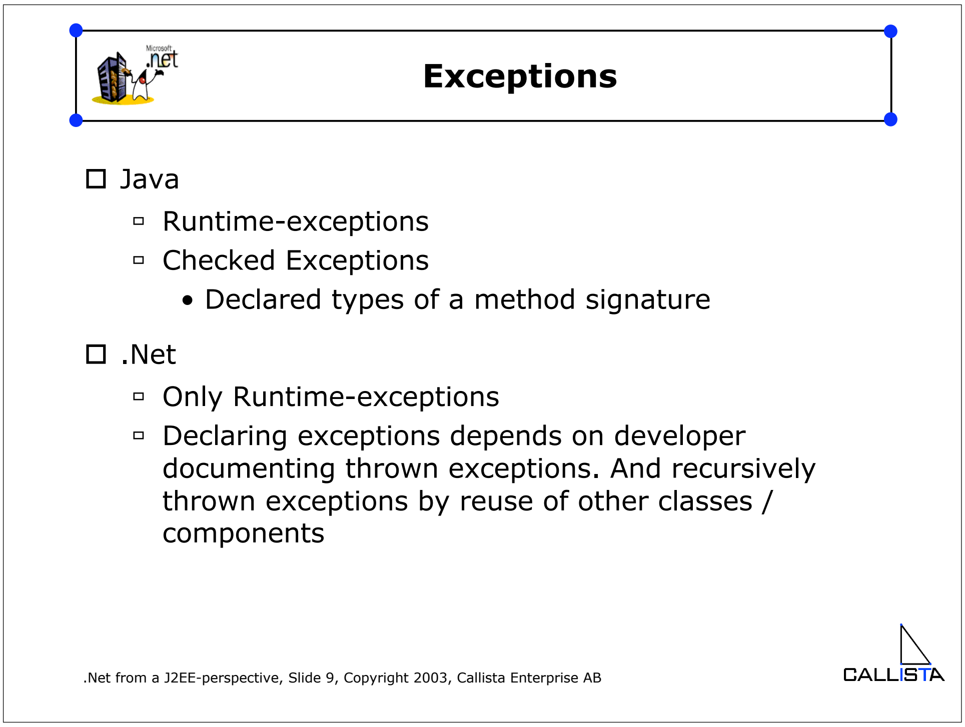
Languages

- .Net is architected to support multiple languages
  - .Net VB
  - C#
  - .Net C++
  - J#
- Third-party compilers for .Net flavours of
  - Eiffel
  - Python
  - Cobol
  - RPG
  - Etc
- For some languages, the flavour overwhelms the heritage



C# - Many similarities with Java

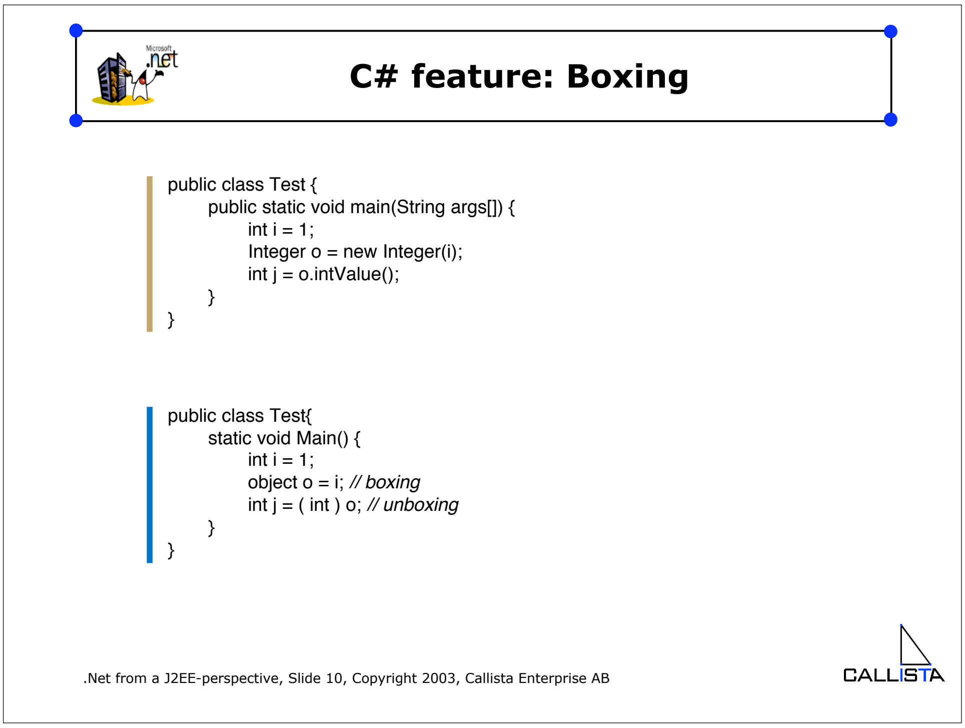
Inner classes	Single inheritance	“Boxing”
Checked Exceptions	Reflection	Typed collections
	Multiple interfaces	Stack allocation of objects
	Class Object	Properties
	Serialization	Events
	Runtime exceptions	Enums



Exceptions

- □ Java
  - Runtime-exceptions
  - Checked Exceptions
    - Declared types of a method signature
- □ .Net
  - Only Runtime-exceptions
  - Declaring exceptions depends on developer documenting thrown exceptions. And recursively thrown exceptions by reuse of other classes / components

CALLISTA

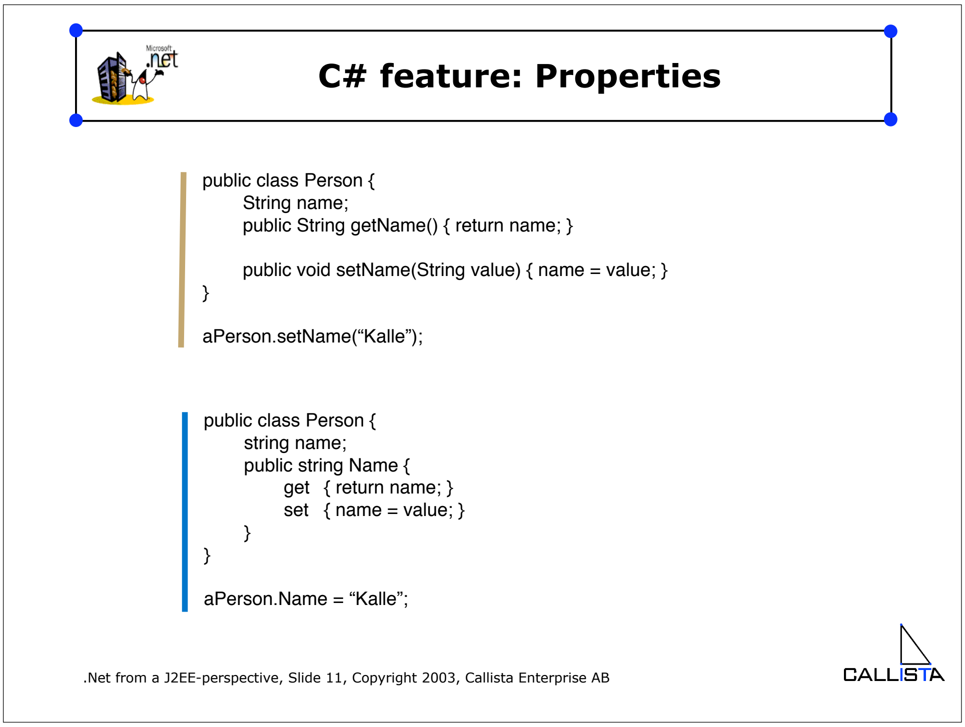


C# feature: Boxing

```
public class Test {
      public static void main(String args[]) {
            int i = 1;
            Integer o = new Integer(i);
            int j = o.intValue();
      }
}
```

```
public class Test{
      static void Main() {
            int i = 1;
            object o = i; // boxing
            int j = ( int ) o; // unboxing
      }
}
```

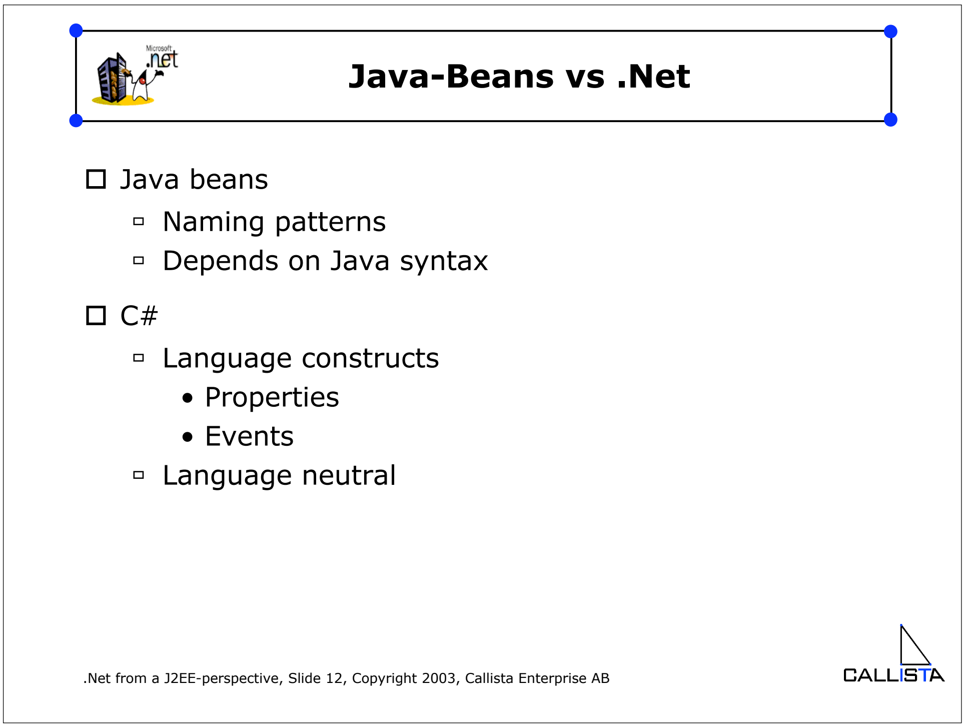
CALLISTA



C# feature: Properties

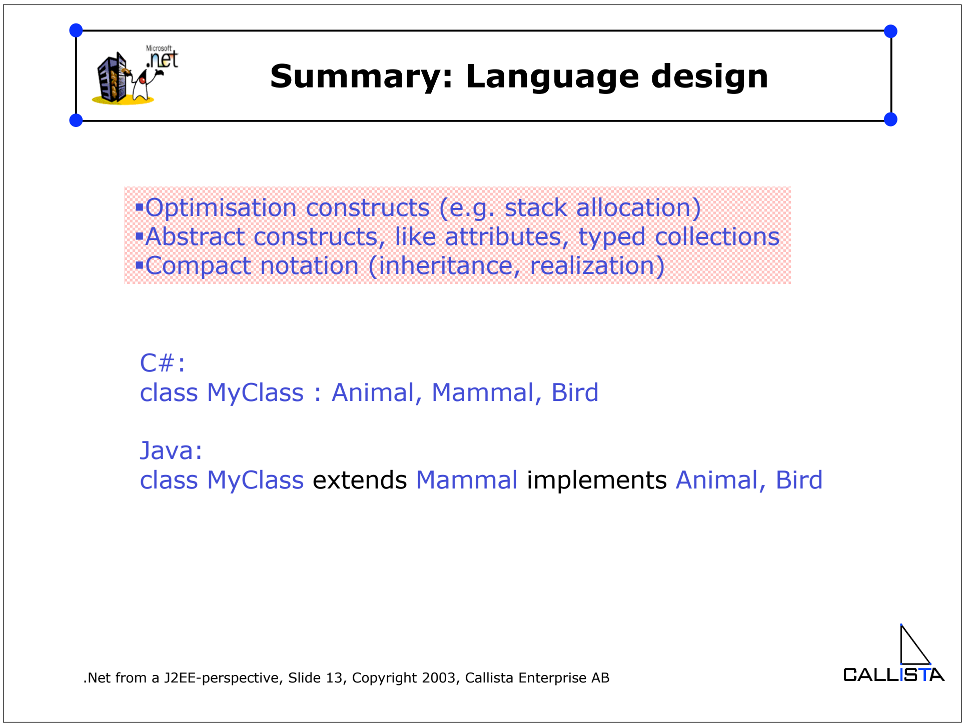
```
public class Person {
        String name;
        public String getName() { return name; }

        public void setName(String value) { name = value; }
}

aPerson.setName(“Kalle”);
```

```
public class Person {
        string name;
        public string Name {
                get   { return name; }
                set   { name = value; }
        }
}

aPerson.Name = “Kalle”;
```



Java-Beans vs .Net

- □ Java beans
  - ▫ Naming patterns
  - ▫ Depends on Java syntax
- □ C#
  - ▫ Language constructs
    - Properties
    - Events
  - ▫ Language neutral

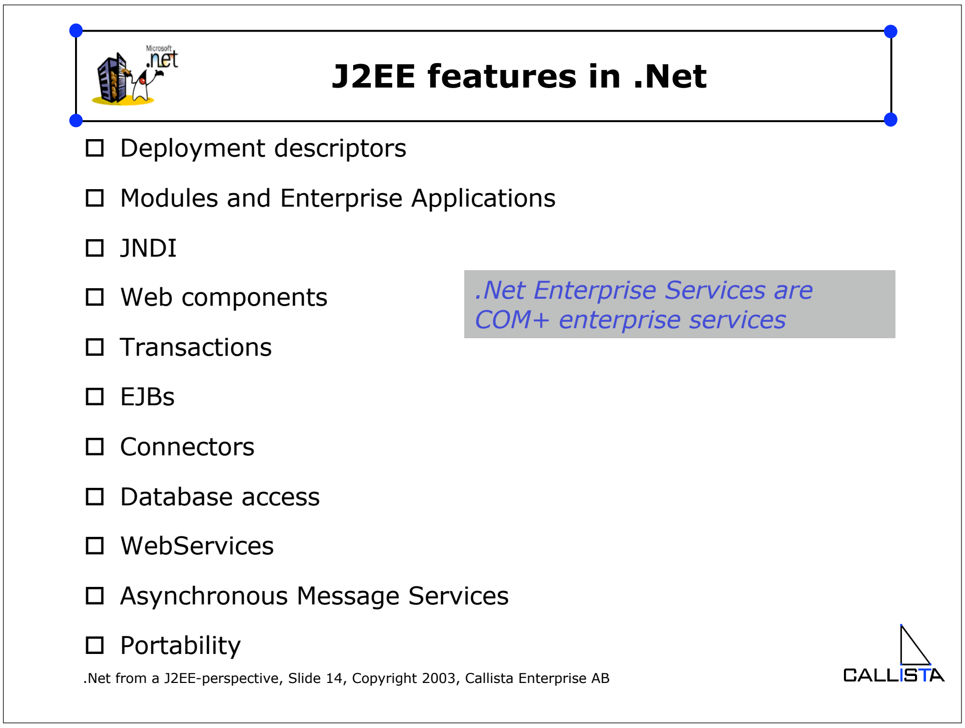


Summary: Language design

- Optimisation constructs (e.g. stack allocation)
- Abstract constructs, like attributes, typed collections
- Compact notation (inheritance, realization)

C#:
class MyClass : Animal, Mammal, Bird

Java:
class MyClass extends Mammal implements Animal, Bird



J2EE features in .Net

- ☐ Deployment descriptors
- ☐ Modules and Enterprise Applications
- ☐ JNDI
- ☐ Web components
- ☐ Transactions
- ☐ EJBs
- ☐ Connectors
- ☐ Database access
- ☐ WebServices
- ☐ Asynchronous Message Services
- ☐ Portability

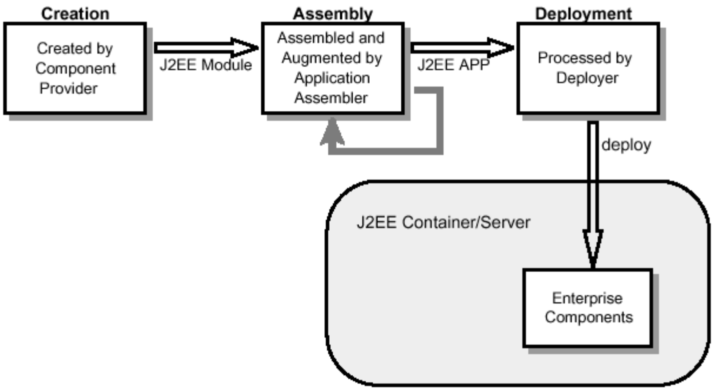
.Net Enterprise Services are COM+ enterprise services

Deployment descriptors

Enterprise Application archive	Module archive	Attributes in code + parts of web.xml
Logical resource references	Manifest	DD-info accessible at run-time
DD separate from Code		
Assembly vision		

.Net from a J2EE-perspective, Slide 15, Copyright 2003, Callista Enterprise AB

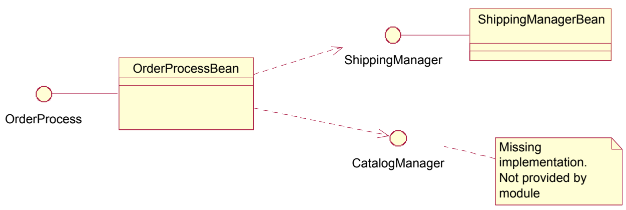
J2EE assembly vision



.Net from a J2EE-perspective, Slide 16, Copyright 2003, Callista Enterprise AB

J2EE Assembly task - example

- Each component defines its dependencies to other *interfaces*
- A component may suggest an implementation defined by the same or another module

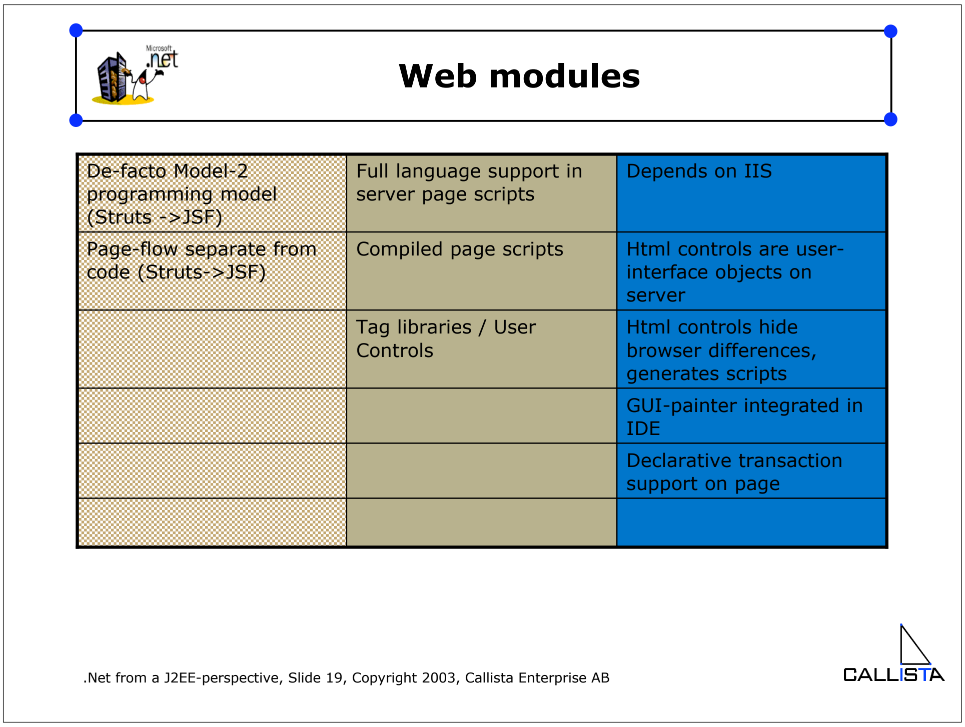


Interface dependency. Implementation available in module.

Interface dependency. Assembler MUST provide implementation

JNDI

Component Naming Service		Windows Registry for serviced components
Multiple naming domains on single server		.Net Remoting for "in-process naming" (a'la RMI)

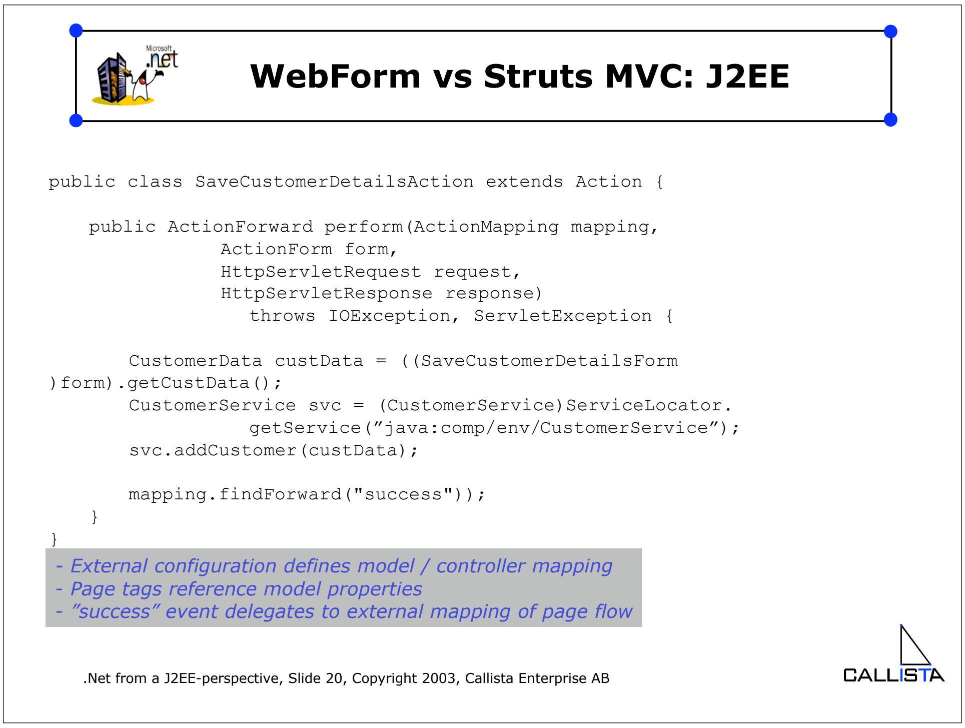


Web modules

De-facto Model-2 programming model (Struts ->JSF)	Full language support in server page scripts	Depends on IIS
Page-flow separate from code (Struts->JSF)	Compiled page scripts	Html controls are user-interface objects on server
	Tag libraries / User Controls	Html controls hide browser differences, generates scripts
		GUI-painter integrated in IDE
		Declarative transaction support on page

.Net from a J2EE-perspective, Slide 19, Copyright 2003, Callista Enterprise AB

CALLISTA



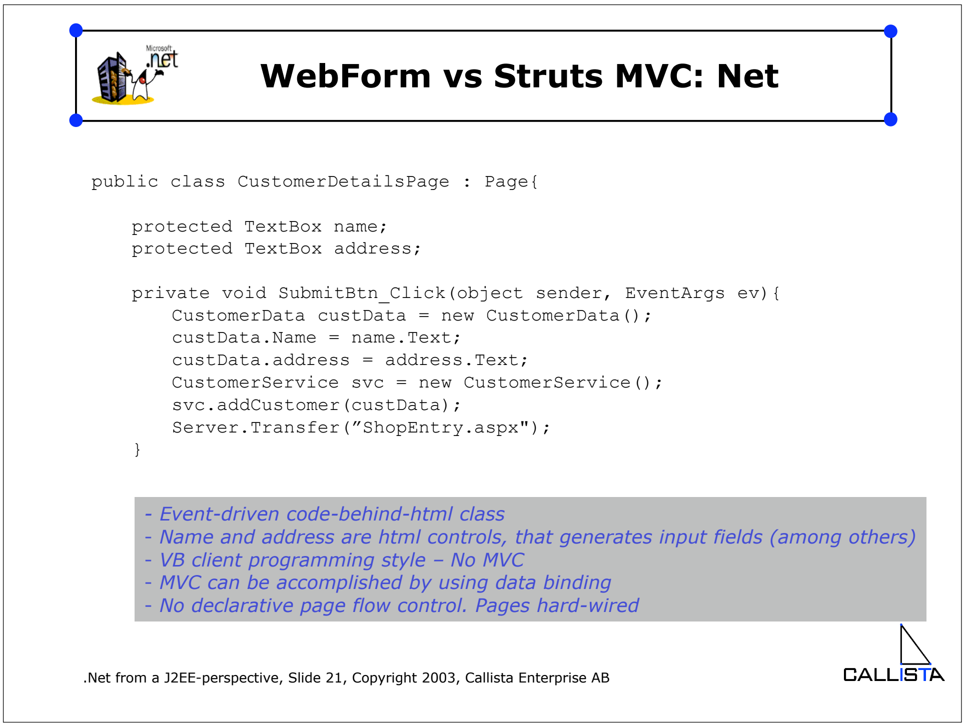
WebForm vs Struts MVC: J2EE

```
public class SaveCustomerDetailsAction extends Action {

    public ActionForward perform(ActionMapping mapping,
                 ActionForm form,
                 HttpServletRequest request,
                 HttpServletResponse response)
                    throws IOException, ServletException {

        CustomerData custData = ((SaveCustomerDetailsForm
)form).getCustData();
        CustomerService svc = (CustomerService)ServiceLocator.
                    getService(”java:comp/env/CustomerService”);
        svc.addCustomer(custData);

        mapping.findForward("success"));
    }
}
```

- *External configuration defines model / controller mapping*
- *Page tags reference model properties*
- *”success” event delegates to external mapping of page flow*

.Net from a J2EE-perspective, Slide 20, Copyright 2003, Callista Enterprise AB

CALLISTA



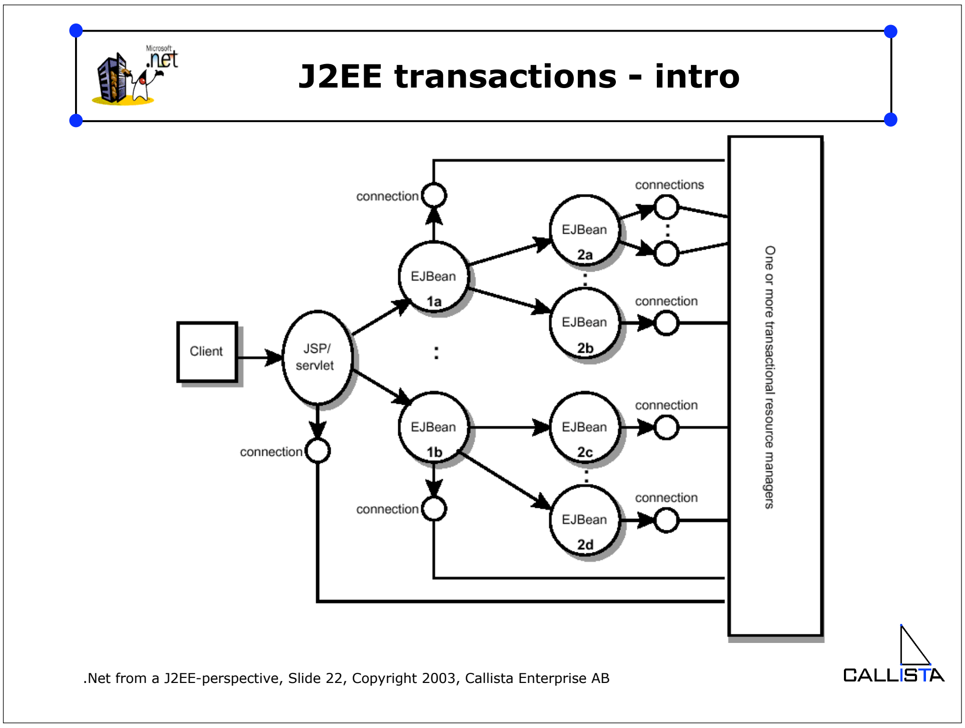
WebForm vs Struts MVC: Net

```
public class CustomerDetailsPage : Page{

    protected TextBox name;
    protected TextBox address;

    private void SubmitBtn_Click(object sender, EventArgs ev){
        CustomerData custData = new CustomerData();
        custData.Name = name.Text;
        custData.address = address.Text;
        CustomerService svc = new CustomerService();
        svc.addCustomer(custData);
        Server.Transfer(”ShopEntry.aspx");
    }
```

- Event-driven code-behind-html class
- Name and address are html controls, that generates input fields (among others)
- VB client programming style – No MVC
- MVC can be accomplished by using data binding
- No declarative page flow control. Pages hard-wired

.Net from a J2EE-perspective, Slide 21, Copyright 2003, Callista Enterprise AB

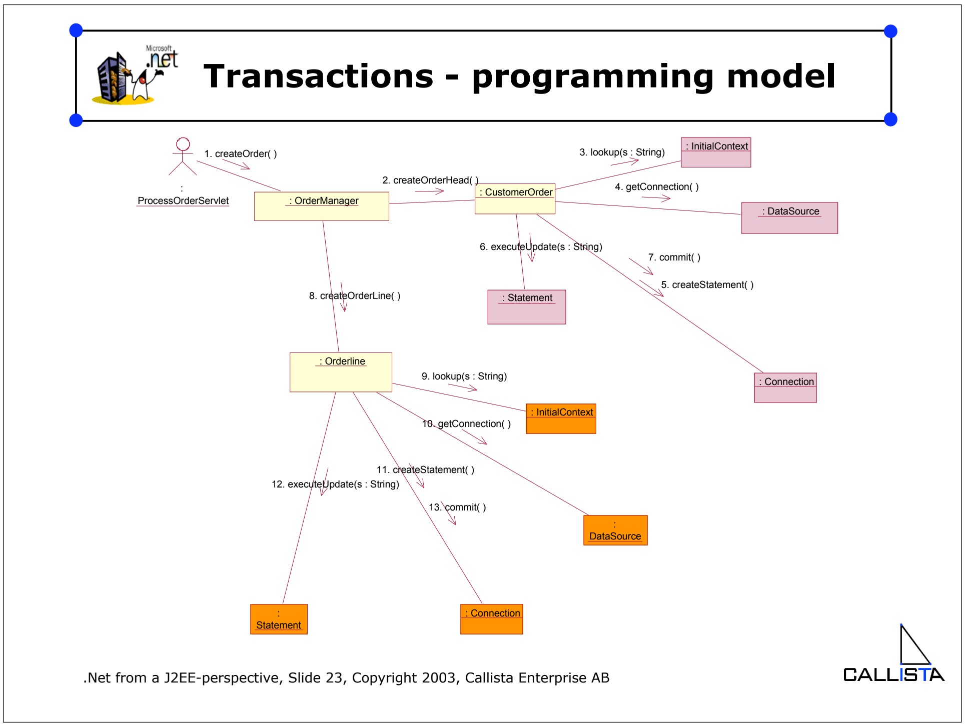
CALLISTA



J2EE transactions - intro

.Net from a J2EE-perspective, Slide 22, Copyright 2003, Callista Enterprise AB

CALLISTA



Transactions - programming model

: ProcessOrderServlet

1. createOrder( )

: OrderManager

2. createOrderHead( )

: CustomerOrder

3. lookup(s : String)

: InitialContext

4. getConnection( )

: DataSource

6. executeUpdate(s : String)

7. commit( )

5. createStatement( )

: Statement

8. createOrderLine( )

: Connection

: Orderline

9. lookup(s : String)

: InitialContext

10. getConnection( )

11. createStatement( )

12. executeUpdate(s : String)

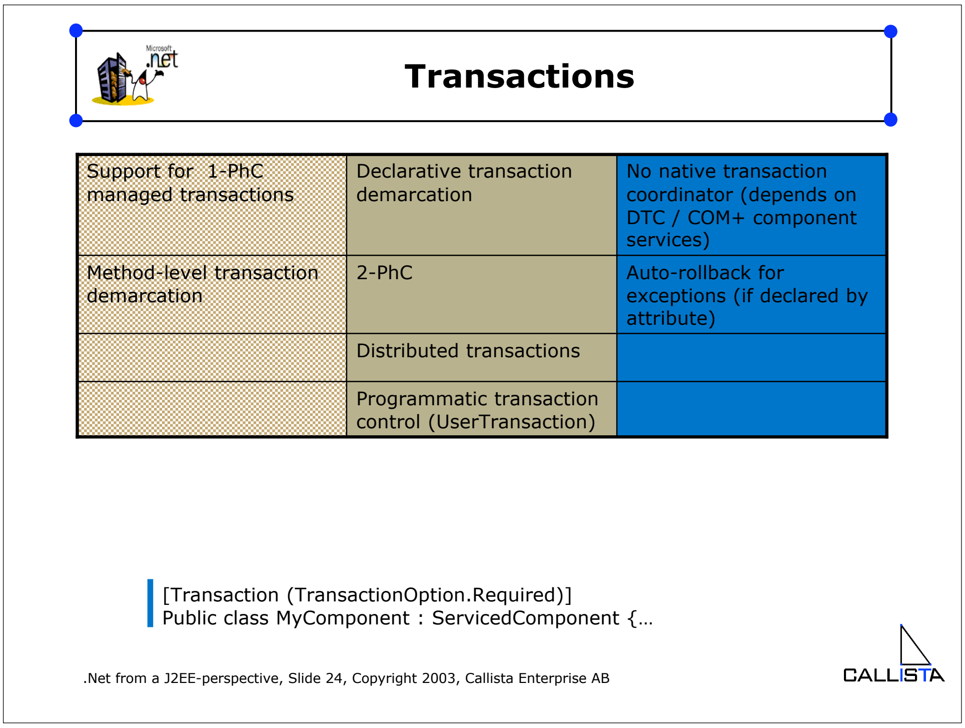
13. commit( )

: DataSource

: Statement

: Connection

.Net from a J2EE-perspective, Slide 23, Copyright 2003, Callista Enterprise AB

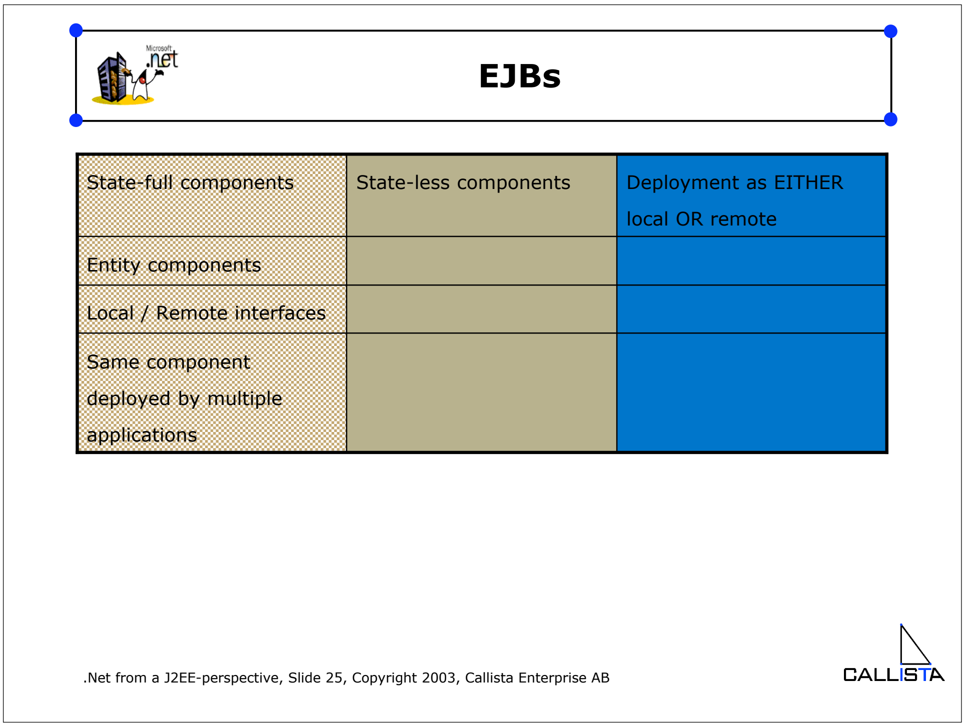


Transactions

Support for 1-PhC managed transactions	Declarative transaction demarcation	No native transaction coordinator (depends on DTC / COM+ component services)
Method-level transaction demarcation	2-PhC	Auto-rollback for exceptions (if declared by attribute)
	Distributed transactions	
	Programmatic transaction control (UserTransaction)	

[Transaction (TransactionOption.Required)]
Public class MyComponent : ServicedComponent {…

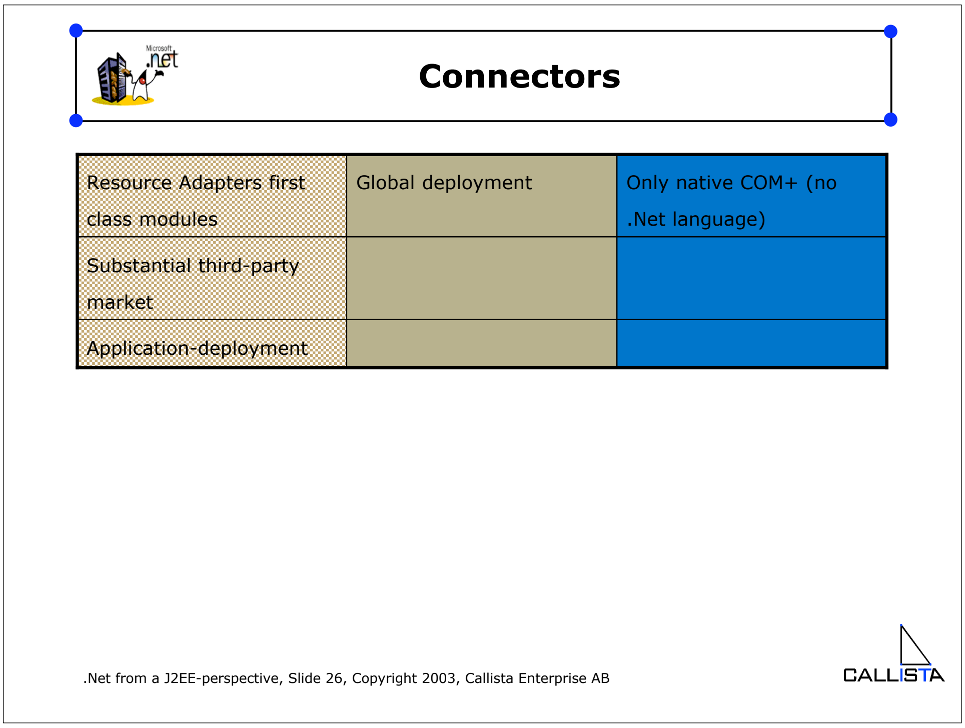
.Net from a J2EE-perspective, Slide 24, Copyright 2003, Callista Enterprise AB



EJBs

State-full components	State-less components	Deployment as EITHER local OR remote
Entity components		
Local / Remote interfaces		
Same component deployed by multiple applications		

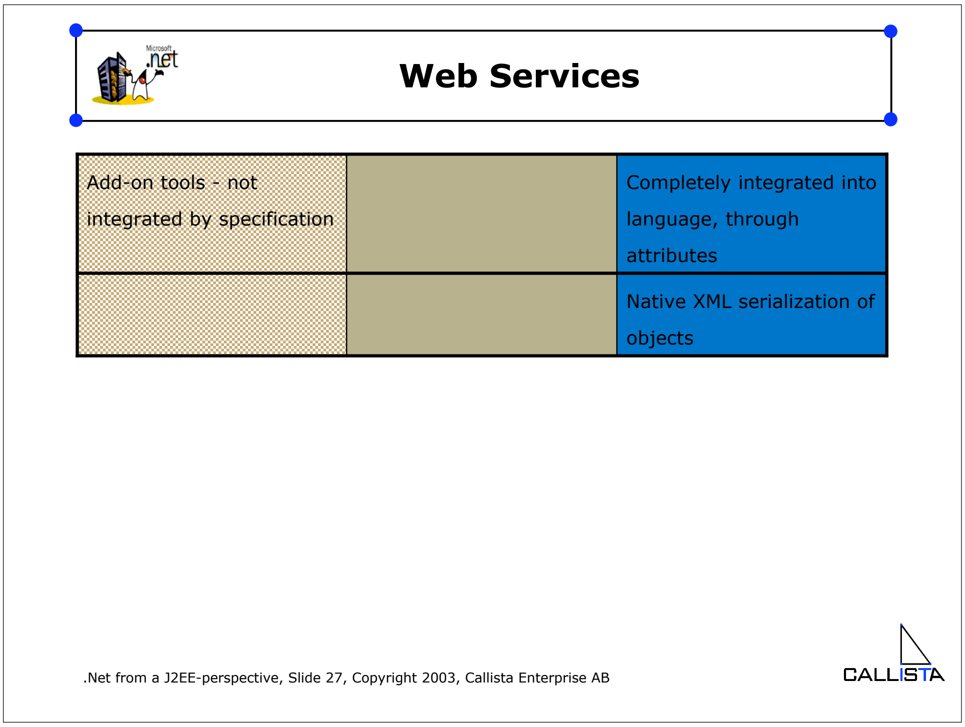
.Net from a J2EE-perspective, Slide 25, Copyright 2003, Callista Enterprise AB



Connectors

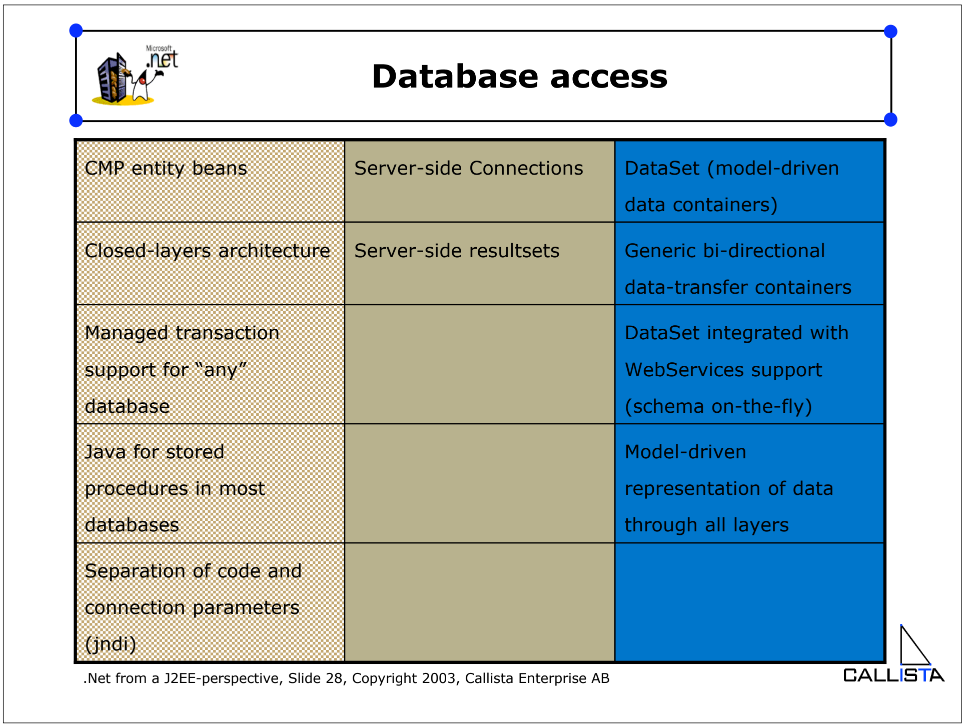
Resource Adapters first class modules	Global deployment	Only native COM+ (no .Net language)
Substantial third-party market		
Application-deployment		

.Net from a J2EE-perspective, Slide 26, Copyright 2003, Callista Enterprise AB



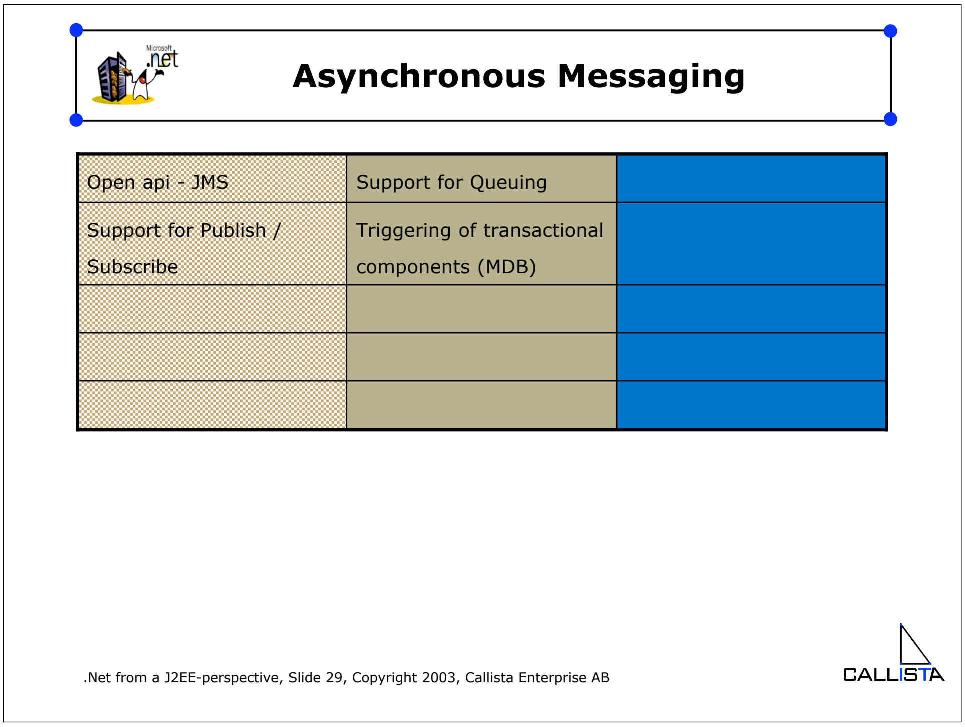
Web Services

Add-on tools - not integrated by specification		Completely integrated into language, through attributes
		Native XML serialization of objects



Database access

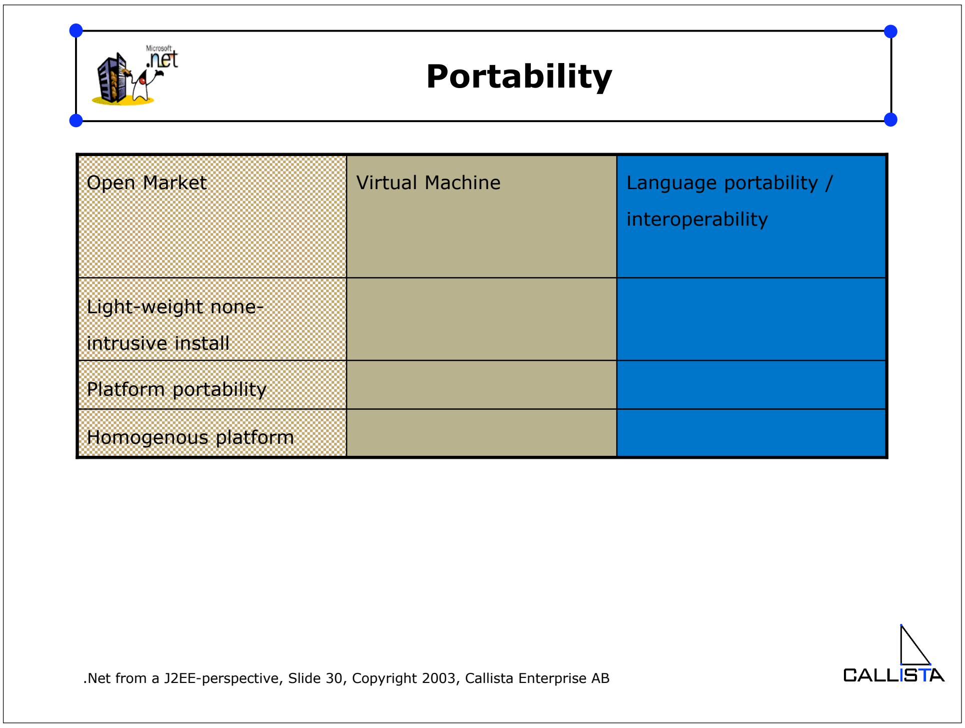
CMP entity beans	Server-side Connections	DataSet (model-driven data containers)
Closed-layers architecture	Server-side resultsets	Generic bi-directional data-transfer containers
Managed transaction support for "any" database		DataSet integrated with WebServices support (schema on-the-fly)
Java for stored procedures in most databases		Model-driven representation of data through all layers
Separation of code and connection parameters (jndi)		



Asynchronous Messaging

Open api - JMS	Support for Queuing	
Support for Publish / Subscribe	Triggering of transactional components (MDB)	

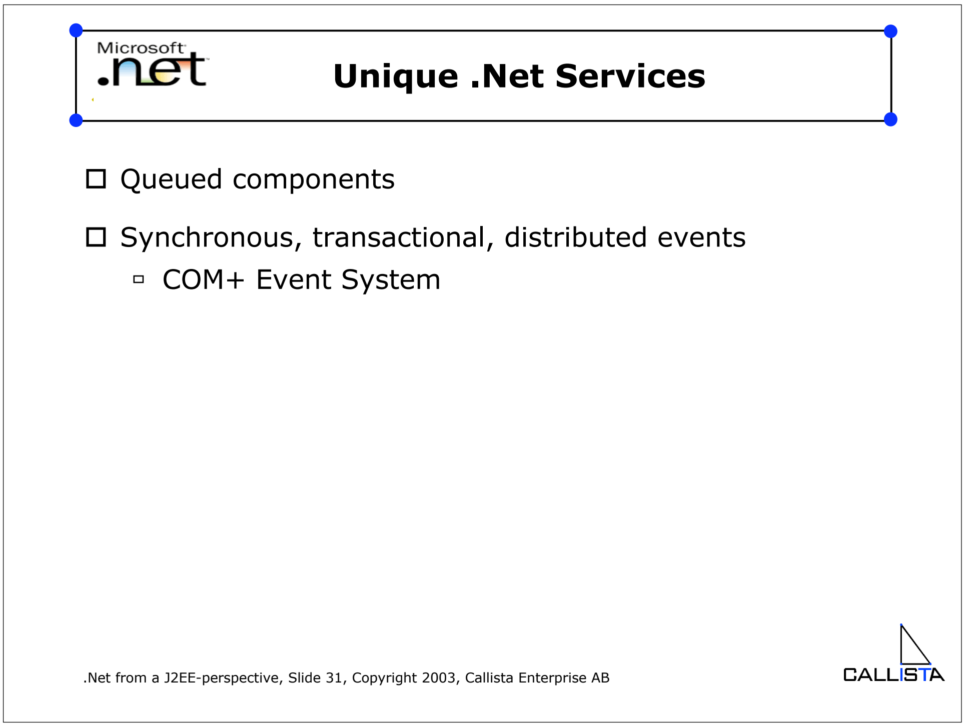
.Net from a J2EE-perspective, Slide 29, Copyright 2003, Callista Enterprise AB



Portability

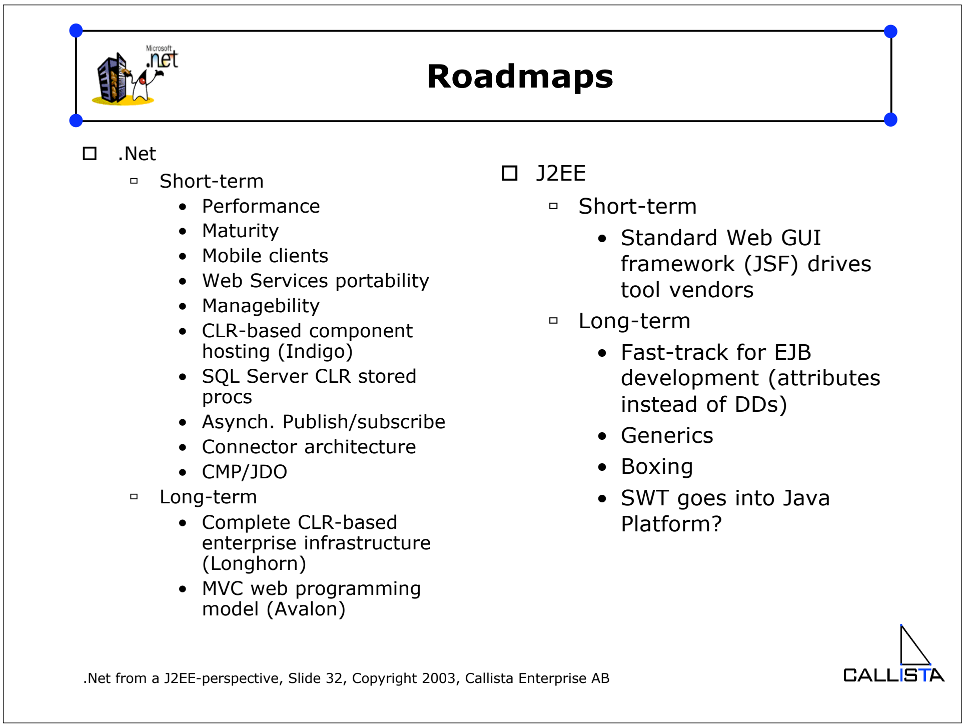
Open Market	Virtual Machine	Language portability / interoperability
Light-weight none-intrusive install		
Platform portability		
Homogenous platform		

.Net from a J2EE-perspective, Slide 30, Copyright 2003, Callista Enterprise AB



Unique .Net Services

- □ Queued components
- □ Synchronous, transactional, distributed events
  - COM+ Event System



Roadmaps

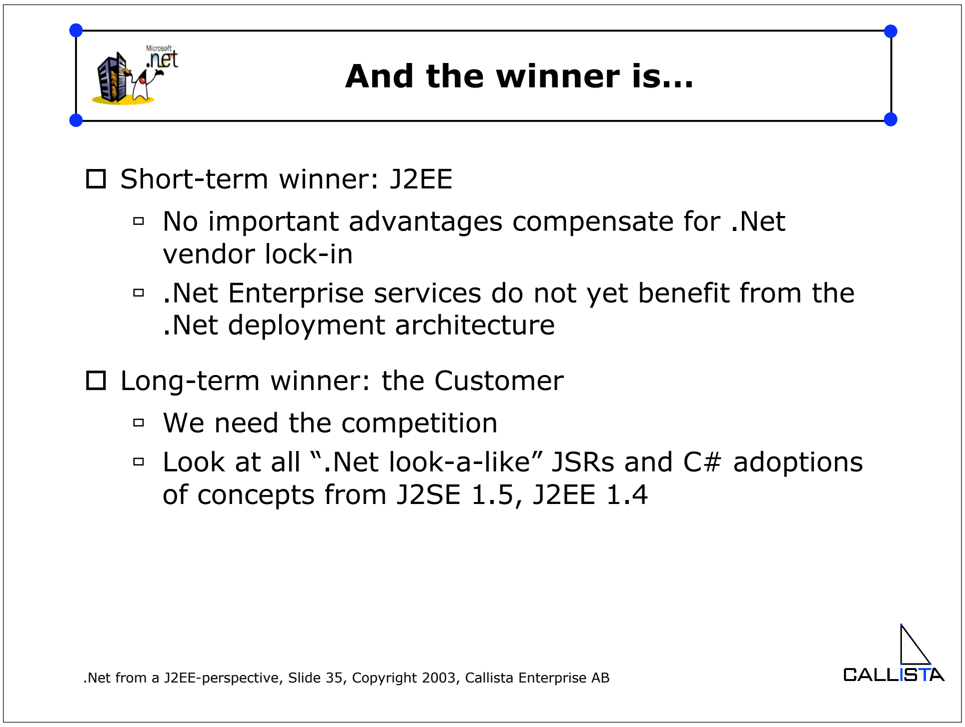
- □ .Net
  - Short-term
    - Performance
    - Maturity
    - Mobile clients
    - Web Services portability
    - Managebility
    - CLR-based component hosting (Indigo)
    - SQL Server CLR stored procs
    - Asynch. Publish/subscribe
    - Connector architecture
    - CMP/JDO
  - Long-term
    - Complete CLR-based enterprise infrastructure (Longhorn)
    - MVC web programming model (Avalon)
- □ J2EE
  - Short-term
    - Standard Web GUI framework (JSF) drives tool vendors
  - Long-term
    - Fast-track for EJB development (attributes instead of DDs)
    - Generics
    - Boxing
    - SWT goes into Java Platform?

Advantages

- □ J2EE +
  - Vendor competition
  - Mature enterprise project culture (test-driven development, refactoring, continues integration)
  - Multi-platform
  - From zOS surgery installations to lap-top one-click install of complete infrastructure
  - Very large and mature open-source community
  - Develop / Assembly / Deploy architecture supports business component market and frameworks
  - Developer environment is free
- □ .Net +
  - User interface builder for WebForms
  - Programming language improvements over Java
  - Rich client programming

Drawbacks

- □ J2EE
  - Deployment descriptors / application assembly is complex
  - Still lack of html GUI framework (JSF is VERY close!)
  - Swing sucks
- □ .Net
  - Only robust with SQL Server?
  - No Publish/Subscribe messaging
  - Low-to-mid-end applications suffer from 2PhC requirements (high-end databases, complexity)
  - Connector architecture missing
  - “dll hell” (registry, intrusive installs, ghost configurations …)
  - Checked exceptions missing - quality !?



And the winner is…

- ☐ Short-term winner: J2EE
  - ▫ No important advantages compensate for .Net vendor lock-in
  - ▫ .Net Enterprise services do not yet benefit from the .Net deployment architecture
- ☐ Long-term winner: the Customer
  - ▫ We need the competition
  - ▫ Look at all ".Net look-a-like" JSRs and C# adoptions of concepts from J2SE 1.5, J2EE 1.4