



# Java Roadmap

**Daniel Adelhardt**

Java Architekt

Sun Microsystems GmbH



# Disclaimer

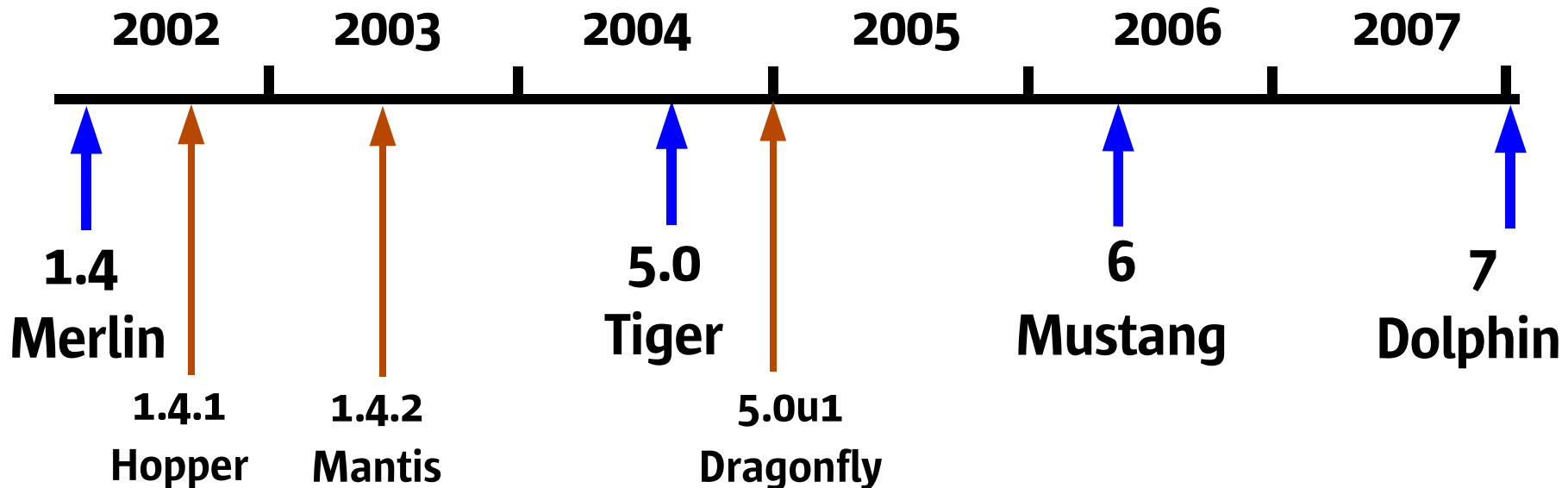
Termine und Inhalte neuer Releases sind “Subject to Change”  
und werden über den Java Community Process definiert!

# JavaONE 2005 Zusammenfassung



- JavaOne 2005 vom 27.6-30.6.2005 in SFO mit ca 15000 Teilnehmer
- Zentrale Themen
  - > Java Plattform Renaming
  - > SOA & Java Business Integration (JBI, JSR-208)
  - > Project Glassfish (Java EE 5 SDK als Open Source)
  - > Java Server Faces
  - > AJAX
  - > Scripting und Java
  - > Java und .Net Interoperability
  - > NetBeans, SwingLabs und Matisse

# Java Platform Standard Edition Roadmap



- > Major Releases mit weniger neuen APIs
- > Keine Minor (z.B. 5.1) Releases mehr, nur noch Update-Releases
- > Häufigere Releases (18 Monats Zyklus)
- > EOL Policy: Supported sind das aktuelle Release und die zwei Major Releases davor: <http://java.sun.com/products/archive/eol.policy.html>

# Project Peabody: Mehr Transparenz des Java Engineerings

- Öffentliche Entwicklung von Mustang (“Java SE 6”)
  - > Veröffentlichung von Snapshots (weekly builds des JDKs mit minimaler Qualitätssicherung)
  - > Download von Sourcen und Binaries unter <http://mustang.dev.java.net>
    - > Lizenzen: Java Research License, Java Internal Use License
  - > Von Java 6.0 unterstützte Plattformen
    - > Solaris Sparc/X86/X64, Linux X86/X64, Windows X86/X64
  
- Möglichkeit Bug Fixes zu submitten!
  - > <https://mustang.dev.java.net/collaborate.html>



# Java Platform Standard Edition 6 ("Mustang")

- Releasedatum (\*) Q2 2006
- Schwerpunkt Themen
  - > Enterprise Desktop
  - > Ease of Development
    - > JSR-269: Pluggable Annotation Processing API
    - > JSR-223: Scripting for the Java Platform
  - > XML & Web Services
    - > JSR 105 XML Digital Signature
    - > JAX-B 2.0, JAX-WS 2.0, JAX-P.Next
  - > Managability und Monitoring
  - > Misc: JDBC 4.0, Java Smart Card I/O API

# Java SE 6 (“Project Mustang”)

## Desktop Java Highlights

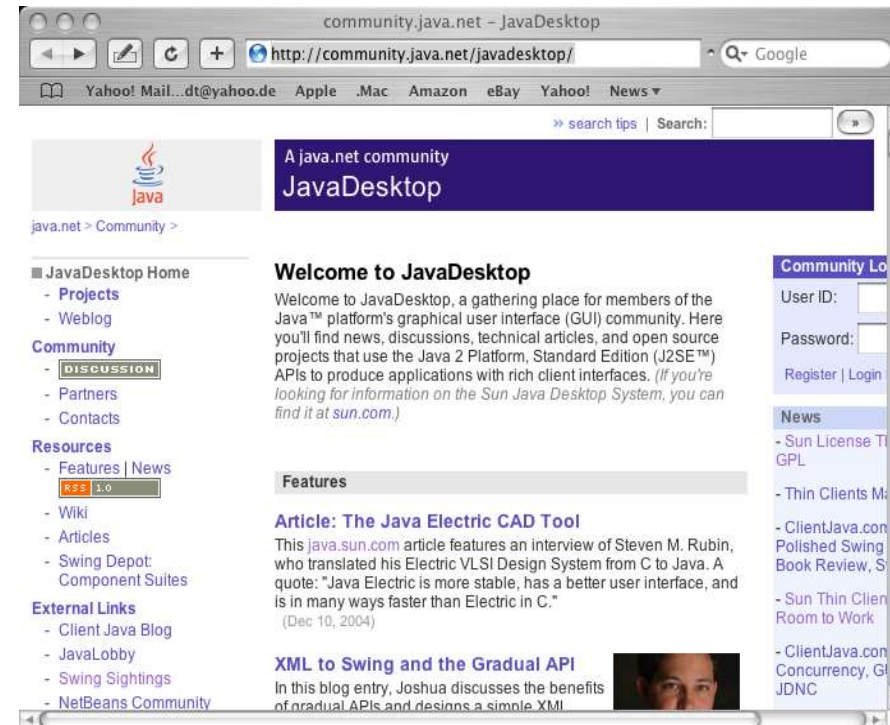
- Swing
  - > Verbesserung des Win32/GTK Look & Feels (**Avalon**), **Native Rendering**, Verbesserung des Double Bufferings, JTable Sorting/Filtering, Text Printing, **SwingWorker**
- Java 2D
  - > Bessere Hardware Beschleunigung, Single Threaded Rendering
- AWT
  - > Mixing von Heavy & Lightweight Komponenten, **Splash Screen** Support, Tray Icon Support/Start Leisten
- Deployment
  - > Helper Controls, Desktop Integration, neue Download Engine für Plugin/WebStart, HTTP Compression, Cache Management

# Java SE 6 (“Project Mustang”)

## Desktop Java

- SwingLabs auf <http://swinglabs.dev.java.net> fungiert als Inkubator für zukünftige JFC/Swing Erweiterungen
  - > Java Desktop Network Components (JDNC)
  - > Java Desktop Integration Components (JDIC)
  - > Komponenten werden sukzessive in Java SE integriert können aber auch mit J2SE 5.0 benutzt werden

**Work in Progress!**



community.java.net - JavaDesktop

http://community.java.net/javadesktop/

search tips | Search:

A java.net community  
**JavaDesktop**

java.net > Community >

JavaDesktop Home

- Projects
- Weblog

Community

- DISCUSSION
- Partners
- Contacts

Resources

- Features | News
- Wiki
- Articles
- Swing Depot: Component Suites

External Links

- Client Java Blog
- JavaLobby
- Swing Sightings
- NetBeans Community

**Welcome to JavaDesktop**

Welcome to JavaDesktop, a gathering place for members of the Java™ platform's graphical user interface (GUI) community. Here you'll find news, discussions, technical articles, and open source projects that use the Java 2 Platform, Standard Edition (J2SE™) APIs to produce applications with rich client interfaces. (If you're looking for information on the Sun Java Desktop System, you can find it at [sun.com](http://sun.com).)

**Features**

**Article: The Java Electric CAD Tool**

This [java.sun.com](http://java.sun.com) article features an interview of Steven M. Rubin, who translated his Electric VLSI Design System from C to Java. A quote: "Java Electric is more stable, has a better user interface, and is in many ways faster than Electric in C."  
(Dec 10, 2004)

**XML to Swing and the Gradual API**

In this blog entry, Joshua discusses the benefits of gradual APIs and designs a simple XML

Community Login

User ID:

Password:

Register | Login

News

- Sun License T GPL
- Thin Clients M
- ClientJava.com Polished Swing Book Review, S
- Sun Thin Client Room to Work
- ClientJava.com Concurrency, G JDNC

# Java 6.0 – Desktop Java

## Java Desktop Integration Components (JDIC)

- Bessere Integration von Java Applikationen in native Desktop Umgebungen
  - > Einbettung von native Browsern in AWT/Swing GUIs über die Klasse `WebBrowser` als `AWTCanvas`
    - > Über eine `WebBrowserListener` Instanz Notifikation bei Browser Events
  - > Zugriff auf native Desktop Applikationen
    - > `Desktop.browse()`, `Desktop.mail()`, `Desktop.open()`, `Desktop.print()`, `Desktop.edit()`, Parameter jeweils URL oder File
  - > `AssociationService` für den Zugriff auf die Filehandler Informationen des OS
    - > Windows Registry, Unix: `.mime/.application` Files
  - > Tray Icon Support
  - > Screen Saver API

**Available now!**  
<http://jdic.dev.java.net>

# Java 6.0 – Desktop Java

## Java Desktop Integration Components (JDIC) Beispiel

```
JFrame frame = new JFrame("MyCustomBrowser");
```

```
WebBrowser webBrowser = new WebBrowser();
```

```
try {
```

```
    webBrowser.setURL(new URL("http://java.net"));
```

```
    } catch (MalformedURLException e) {
```

```
        ...
    }
```

```
JPanel panel = new JPanel();
```

```
panel.add(webBrowser,  
         BorderLayout.CENTER);
```

```
frame.getContentPane().  
    add(panel,  
        BorderLayout.CENTER);
```

```
...
```



# Java 6.0 – Desktop Java

## Java Desktop Network Components (JDNC)

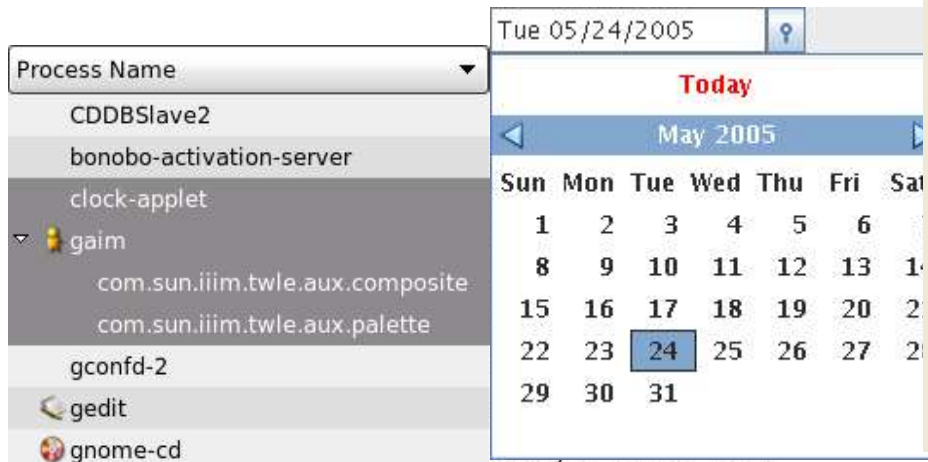
- JDNC ist ein Projekt mit dem Ziel die Entwicklung von GUIs für datenzentrische Unternehmensanwendungen zu vereinfachen
- Derzeitiger Umfang von JDNC
  - > XML Format zur deklarativen Beschreibung von UI Komponenten
  - > Databinding und Validation API
  - > High Level Komponenten (JXComponents) aufbauend auf Swing
    - > Record basiertes Datenmodell
    - > Unterstützung für Typ Konvertierung und Input-Validierung
    - > Databinding um Records mit Swing Controls zu verbinden

**Available now!**  
**<http://jdnc.dev.java.net>**

# Java 6.0 – Desktop Java

## JDNC Beispiel: JXComponents

- JXComponents sind customizable und einfach verwendbare Swing Komponenten
  - > <http://swingx.dev.java.net>
  - > Beispiele:  
 JXDatePicker, JXTable, JXTreeTable, JXEditorPane, JXPanel



Process Name

- CDDBSlave2
- bonobo-activation-server
- clock-applet
- gaim
- com.sun.iim.twle.aux.composite
- com.sun.iim.twle.aux.palette
- gconfd-2
- gedit
- gnome-cd

Tue 05/24/2005

Today

May 2005

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				



Login

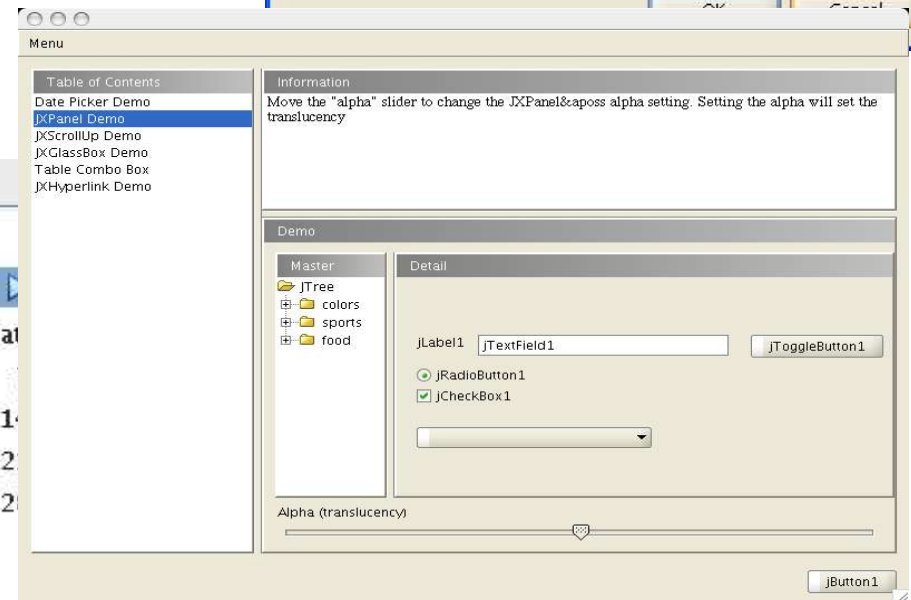
## Login

Enter your user name and password

Name:

Password:

Remember Password



Menu

- Table of Contents
- Date Picker Demo
- JXPanel Demo
- JXScrollUp Demo
- JXGlassBox Demo
- Table Combo Box
- JXHyperlink Demo

Information

Move the "alpha" slider to change the JXPanel's alpha setting. Setting the alpha will set the translucency

Demo

Master

- JTree
- colors
- sports
- food

Detail

jLabel1:

JRadioButton1

JCheckBox1

Alpha (translucency)

# Java Platform Standard Edition 6

## Scripting for the Java Platform

- Mustang beinhaltet JSR 223 (“Scripting for the Java Platform”)
  - > Bidirektionale Integration von Java und Scripting Engines
    - > Script Aufrufe durch Java und Aufruf von Java Objekten durch Scripts
    - > Bootstrapping und Discovery von Scripting Engines über **ScriptEngineFactory**
    - > Assoziation von ScriptEngines mit Mime-Types
  - > Web Tier Integration nicht enthalten
  - > Sun Java SE 6 Distribution beinhaltet eine JavaScript Engine
    - > Rhino Java Script Engine von Miozilla (<http://www.mozilla.org/rhino/>)

# Java SE 6/Java EE 5 – XML Web Services

- Java SE 6 enthält umfangreichen Web Service Support
  - > JAX-WS 2.0 (ehemals JAX-RPC)
  - > XML Signature und Encryption APIs
  - > Streaming API for XML (StAX)
    - > XML Pull Parser mit höherer Performance als DOM/SAX basierte Parser
  - > JAX-B 2.0 (Java API for XML Databinding)
    - > Vollständige Unterstützung für XML Schema,
    - > Verwendung von Annotations, Enums, Generics und Autoboxing in generierten Klassen

## JAXB-Annotated Class

```

@XmlType public class Trade {
    public int getQuantity() {...}
    public void setQuantity() {...}
    @XmlAttribute
    public String getSymbol() {...}
    public void setSymbol() {...}
  
```



## Generated XML schema

```

<xs:complexType name="trade">
  <xs:sequence>
    <xs:element name="quantity" type="xs:int"/>
  </xs:sequence>
  <xs:attribute name="symbol" type="xs:string"/>
</xs:complexType>
  
```

# Weitere Mustang Inhalte

- Compiler : Split Verifier und Compiler API
- Tracing, Monitoring und Diagnosability
  - > OutOfMemory Error Detection inkl. HeapDump Analyse
  - > JConsole Verbesserungen (L&F, Deadlock Analyse)
  - > JStack auf Remote JVMs
  - > DTrace Probes für JVM
- Common Annotations (JSR-250)
  - > Standardisierung von Annotation Tags für J2SE/J2EE
    - > Z.B. für RMI, JNDI, JAX-RPC und JAX-B
- Garbage Collection: Parallel Old Generation Collector und Client Ergonomics

# Java SE 7 (“Project Dolphin”)

## Erstes Brainstorming...

- Native XML Support (java.lang.XML)
  - > Mögliche Syntax: `<XML var>/#<element>`
- Neues Byte Code Format für Scriptsprachen
- Weitere Spracherweiterungen
  - > Method References, Friends
- Neues Packaging Format
  - > Java Modules mit besserer Versionskontrolle und Deployment Repository
- Mehr NIO Features
- Virtual File Systems

# Java Enterprise Edition 5

- Java EE 5 ist für das erste Halbjahr 2006 geplant
  - > Early Access als Projekt Glassfish verfügbar!
- Schwerpunkte von J2EE 5.0 (“Ease of Development”)
  - > PoJo basiertes Programmiermodell
    - > Vereinfachung von EJB und Web Service Entwicklung
  - > Exzessive Verwendung von Annotations
  - > Größtmöglicher Verzicht auf Deployment Deskriptoren
  - > Resource Injection (“Inversion of Control”)
  - > Java Server Faces als neues Web Framework
  - > Umstellung auf Java SE 5.0
  - > Einige neue APIs und Bereinigungen der Plattform

# Java EE 5 Inhalte im Detail

- JSP Standard Tag Library (JSR-52)
- StAX (JSR-173)
- Web Services Metadata (JSR-181)
- JAXB 2.0 (JSR-222)
- Common Annotations (JSR-250)
- JavaServer Faces 1.2 (JSR-252)
- Java Persistence API (JSR-220)

*New APIs*

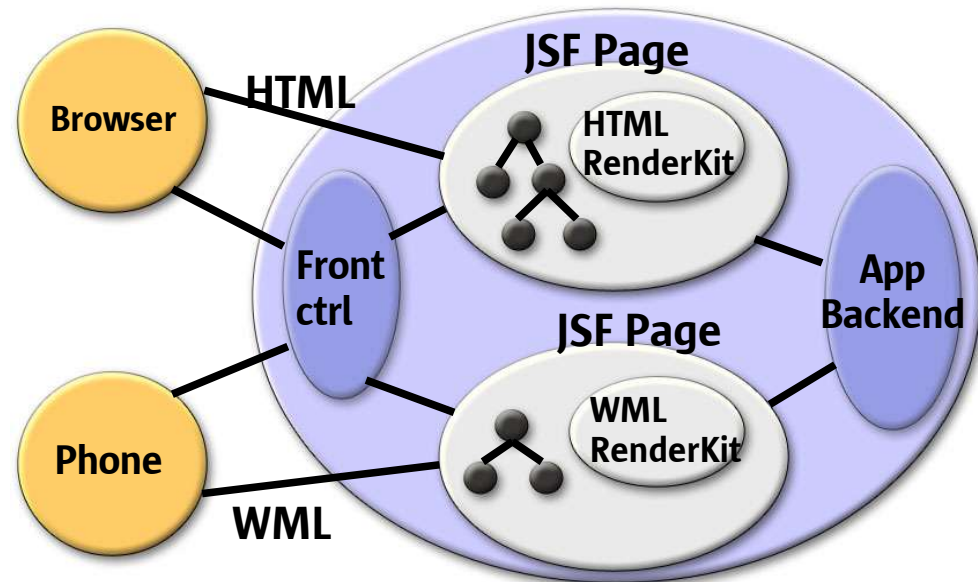
- EJB 3.0 , JAX-WS 2.0, JSP 2.1, Servlet 2.5, JSR-109, Java Mail

*Updated*

# Java EE 5.0

## Java Server Faces 1.2 Architektur und Zielsetzung

- JSF ist ein MVC Framework für Web Applikationen
  - > Umfangreiche Hierarchie von Server seitigen `UIComponents` für die Abbildung des User Interfaces
  - > Tag Libraries für die Einbindung der Komponenten in JSP Pages
  - > Renderer, die das spezifische Markup für eine `UIComponent` erzeugen (z.B. HTML)
- Bestandteile der API
  - > User Interface Komponenten
  - > Input Validierung
  - > Typ Konvertierung
  - > Page Navigation
  - > Event Handling



# Java EE 5.0

## Web Services mit JAX-WS 2.0

- Web Services für Java EE 5 werden über JSR-181 (Web Service Metadata) und JSR-224 (JAX-WS 2.) definiert
  - > Ausgang für Entwicklung: WSDL oder Java Klasse
    - > 'apt' Tool generiert WSDL aus Annotations bzw. 'wsimport' generiert Service Endpoint
    - > Deployment in Servlet Container oder Ausführung direkt mit Java SE
  - > WSDL Generierung über weitere Annotations parametrisierbar
    - > Aufruf Semantik (z.B. OneWay), Namen von Parametern, SOAP Encoding etc

```

@WebService (
    name = "HelloService",
    targetNameSpace="http://foo.com/hello")
public class Hello {
    @WebMethod
    public String sayHello(String param) {
        return "Hello " + param;
    }
}

```

# Java EE 5.0

## Enterprise Java Beans 3.0 – Generelle Themen

- Das Programmiermodell von Enterprise Java Beans wird radikal vereinfacht
  - > Evolution in Richtung POJO Beans
    - > Erhöhung der Produktivität durch Reduktion der zu entwickelnden Klassen/Interfaces
    - > Vereinfachung der APIs durch neues EJB Programmiermodell mit Annotations
  - > Testbarkeit von EJBs auch außerhalb eines Containers!
  - > Zugriff auf Environment über Annotations und Injection
  - > Größtmöglicher Verzicht auf Deployment Deskriptoren
    - > Verwendung von Defaults z.B. für Transaction oder Security Settings
    - > Überschreiben der Defaults möglich
  - > Rückwärtskompatibilität zu EJB 2.1

# Java EE 5.0

## Enterprise Java Beans 3.0 – Session Beans

- Session Beans als Plain Old Java Objects
  - > Kein Home, Remote, Local oder SessionBean Interface
    - > (Optional) Business Interface als einfaches Java Interface
    - > Life Cycle Callbacks per Annotation ( @PostConstruct, @PreDestroy und @PostActivate, @PrePassivate) deklariert

```
@Stateless @Remote public class MySLSessionBean {  
    @Resource public DataSource mydb;  
    public void myBusinessMethod(...){...}  
}
```

```
@Statefull public class MySFSessionBean {  
    private String username;  
  
    @Init public void myInit(String s){...}  
    @Remove public void myRemove(...){...}  
    public void myBusinessMethod(...){...}  
}
```

# Java EE 5.0

## Enterprise Java Beans 3.0 – Interceptors

- Interceptoren wirken auf Business Methoden einer Bean ein
  - > 1-n Interceptor Klassen pro Bean per `@Interceptor`, `@Interceptors` deklariert
  - > Methode mit `@AroundInvocation` Annotation in der Bean bzw. der Interceptor Klasse

```
@Stateless
@Interceptors ({ "foo.Logger", "foo.Audit" })
public class MySLSessionBean {
    public void myBusinessMethod(...){...}
}
```

```
public class Logger {
    @AroundInvoke
    public void log(InvocationContext ctx){
        System.out.println(ctx.getMethod().getName());
        inv.proceed();
    }
}
```

# Java EE 5

## Enterprise Java Beans 3.0 – Entity Beans

- Design Ziele der “Entity Beans”
  - > Einfaches POJOs Bean Modell wie bei Session Beans
  - > Besseres Domain Modelling: Vererbung
  - > Testbarkeit und Verwendbarkeit ausserhalb eines EJB Containers
  - > Eliminierung von “Anti”-Patterns wie Transfer-Objekten
- Persistente Entitäten wird in der Java Persistence API (“Common Persistence API”) spezifiziert.
  - > “Gemergte” EJB3/JDO2 Expertengruppe
  - > API verwendbar für Java SE und Java EE

# Java EE 5

## Enterprise Java Beans 3.0 – Entity Beans

- Entity Beans Programmiermodell
  - > Annotiertes POJO mit Annotations für Relationships (inkl Lazy und Eager Loading), IDs, etc.
  - > O/R Mapping per Annotations, Defaults oder XML Descriptor
    - > Logisches Object Model: @OneToMany, @ManyToMany, @ManyToOne, @OneToOne
    - > Physisches DB Model: @Table, @JoinColumn, @AssociationTable
  - > Erweiterungen bei EJB-QL: Sub Selects, Aggregations, Functions, Dynamic Queries, Update/Delete
  - > “Detached Instances” - Instanzen können auch ausserhalb einer Transaktion benutzt und modifiziert werden
  - > EntityManager als Home Ersatz
    - > Bietet LifeCycle Operationen wie merge/flush/create etc. sowie an und arbeitet als Factory für Queries

# Java EE 5

## Enterprise Java Beans 3.0 – Entity Bean Beispiel

```
@Entity @Table("MyTable")
public class MyEntity {
    private Long id;
    private String myatt1;
    private Hashmap myrelation = new Hashmap();

    @PK(generated=true) public Long getID() {
        return id;
    }
    private void setID (Long id) {
        this.id = id;
    }

    @OneToMany(CASCADE=ALL)
    public Set<MyDepEntity> getMyDependentEntities(){
        return this.myrelation;
    }
}
```

# Java EE 5: Resource Injection

- Abhängigkeit zu einer Resource deklariert per Annotation eines Feldes oder einer Methode
  - > Container fügt Resource zur Laufzeit ein
    - > Unterstützt bei Servlets, JSPs, JSF managed Beans, JAX-WS Endpoints, EJBs
    - > Ressourcen sind z.B. Web Services References, Resource Manager Connection Factories (JDBC, JavaMail, JMS, URL), ORB, UserTransaction, EJB References oder einfache Environment Parameter
    - > Annotations können **optional** durch Deployment Descriptor überschrieben werden
    - > Alternativ Dynamischer Lookup per EJBContext.lookup("<myresource>")

```
//Setter Injection
@Stateless public class MySLSB{
private DataSource ds;
@Resource
public void setDS(DataSource db){
    this.ds = db;
}}
```

```
//Resource Field Injection
@Resource int myThreshold;
@EJB private MyFacade facade;
@Resource javax.sql.DataSource mdb;
@Resource javax.jms.Queue myQueue;
@Resource UserTransaction tx;
```

# Java EE 5: Verschiedene Änderungen

- Security Änderungen
  - > Gleichsetzung der Security Permissions von Web und EJB Container
- Unterstützung für shared Libraries in Enterprise Archiven
  - > *Bundled Libraries*
    - > Über **Class-Path** header im Manifest der JAR Files oder über **<library-directory>** Element mit Angabe eines Directories für shared .jar Files im .EAR
  - > *Installed Libraries*
    - > Spezifikation von Abhängigkeit zu installierten .jar Files per **Extension-List** Attribut im Manifest eines Archivs

```

META-INF/MANIFEST.MF:   App1.ear
Extension-List: util
util-Extension-Name: com/sun/util
util-Specification-Version: 1.4
  
```



```

META-INF/MANIFEST.MF:   util.jar
Extension-Name: com/sun/util
Specification-Title: My Util
Specification-Version: 1.4
Specification-Vendor: Sun Micro
Implementation-Version: 0.1a
  
```

# Java EE Announcements

**JavaOne**  
Sun's 2005 Worldwide Java Developer Conference™

- Project Glassfish - Open Source J2EE 5.0 Application Server
  - > Öffentliche Entwicklung der J2EE 5.0 Referenz Plattform (Sun Java System Application Server 9.0 PE)
  - > Download und CVS Zugang unter <http://glassfish.dev.java.net>
  - > CDDL License
- Common Persistence API (EJB 3.0 JSR-220)
  - > Oracle wird Co-Speclead der EJB 3.0 Spec und liefert die Referenzimplementierung der Persistence API ("TopLink")
  - > Wird Bestandteil des Glassfish Projektes



# SOA meets J2EE

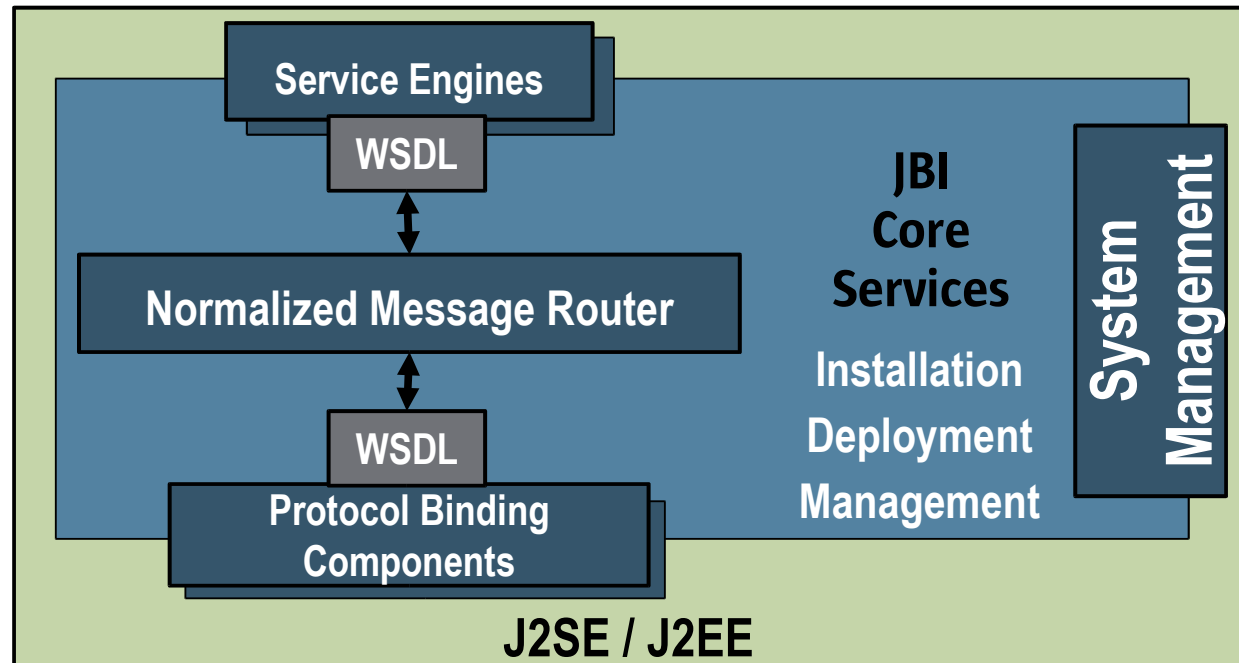
## Java Business Integration (JSR-208)

- Java Business Integration bildet SOA auf Java/J2EE Architekturen ab
  - > Erweiterung der Java/J2EE Plattform um eine *standardisierte Architektur* für Service orientierte Architekturen
  - > Schaffung eines Standards und Marktes für ISVs im Integrationsbereich
    - > Analogie: J2EE Standard für Application Server
  - > JSR 208 Expertengruppe: Sun (Spec- Lead), SAP, Oracle, SeeBeyond, Tibco, Sonic, WebMethods, JBoss, Apache, Deutsche Post und viele weitere JCP Teilnehmer
  - > Finale Spezifikation und SDK verfügbar unter <http://jcp.org>

<http://java.sun.com/integration>

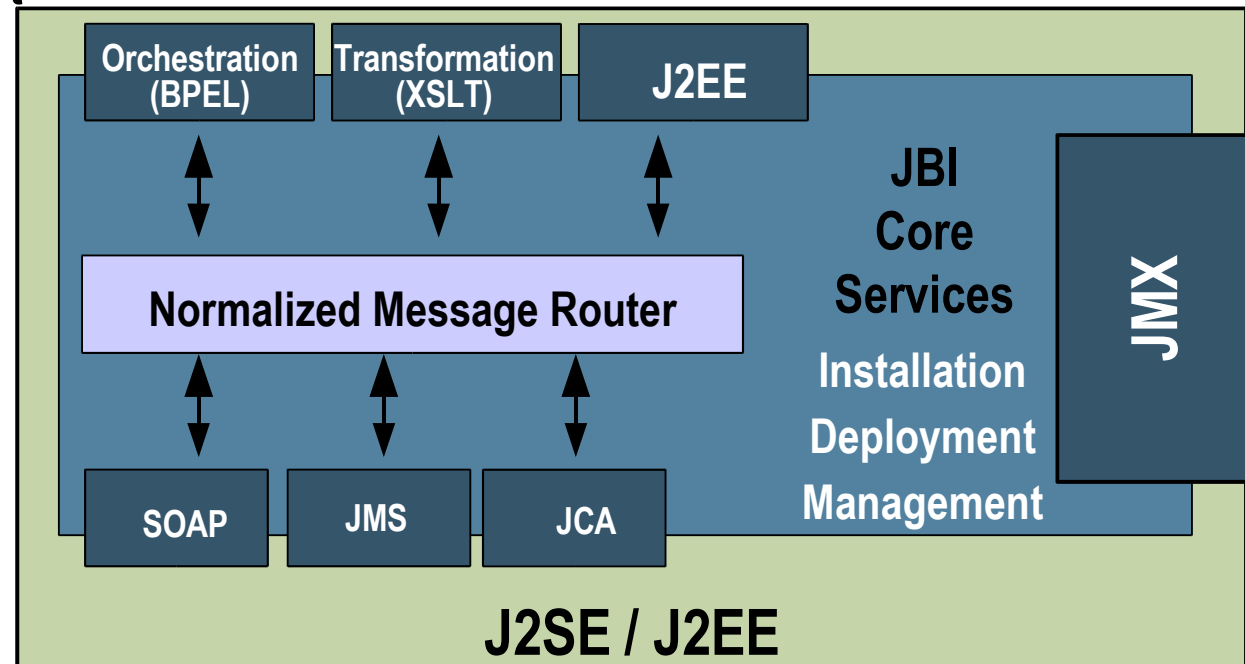
# Java Business Integration (JBI) Konzepte

- JBI definiert einen Meta-Container und die Infrastruktur für integrierte Services
  - > Normalized Message Router als Message Bus für standardisierte Nachrichten
  - > Service Engines für Abarbeitung der Business Logik und unterstützende Funktionen
  - > Binding Komponenten arbeiten als Protokoll Adapter für Zugriff auf remote Service Provider und für den Zugriff von Service Consumern



# JBI Architektur

- Service Engines sind z.B. EJB Container, BPEL Engines, Rules Engines, Transformationsdienste
- Protokoll Binding Komponenten sind z.B. WS-I BP Soap per JAX-RPC, JMS, JCA, IIOP, EDI etc.
- Management Services werden über Java Management Extensions (JMX) realisiert



# Java Business Integration

- JBI definiert eine umfassende und offene SOA Infrastruktur
  - > Life Cycle Services für Installation/Deinstallation und Monitoring von Komponenten
  - > Kommunikation zwischen Service Providern und Consumern
    - > Aufrufsemantik beschrieben über WSDL 2.0 Message Exchange Patterns (In-Only, Robust-In-Only, In-Out, In-Optional-Out...)
    - > Normalized Message Router als Mediator für Nachrichten
    - > Normalisierte Nachrichten sind Metadaten (Security Tokens, Context Informationen etc) und Payload
    - > Plugins beschreiben ihre Fähigkeiten per WSDL 2.0 (Endpoints, Operationen, Nachrichten)
    - > Service Producer und Consumer sind entkoppelt
- Open Source: <http://open-esb.dev.java.net> !!!



# Java Roadmap

[daniel.adelhardt@sun.com](mailto:daniel.adelhardt@sun.com)

