



---

# Java Forum 2002

## Weblogic Server 7.0 & WebLogic Workshop

***Thomas Walter***

*Manager Enterprise Java Group*

*(Central / Eastern Europe)*

*mailto:thomas.walter@bea.com*

# Inhalt

---

- WebLogic Server 7.0
  - Einführung
  - Einige APIs: EJB, JMS, JCA und JTA
  - Betriebsaspekte
- WebLogic Workshop
  - Einfache Web Services
  - Enterprise Web Services
  - JWS: Enterprise Web Services in Java
  - Grafische IDE

# Einführung WebLogic Server

- Ältester unabhängiger Java Lösungsanbieter (Gegründet als WebLogic Inc., seit Sept. 98 bei BEA)

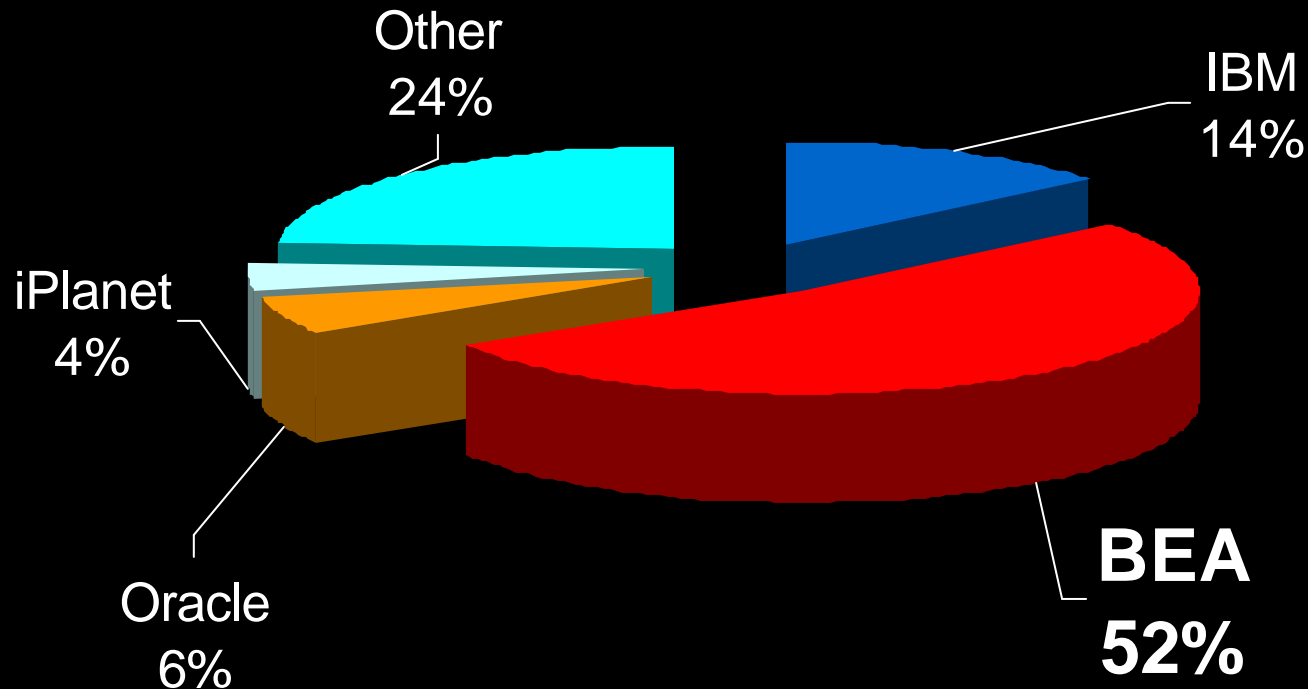


# Meta Group: J2EE /EJB Application Server Markt

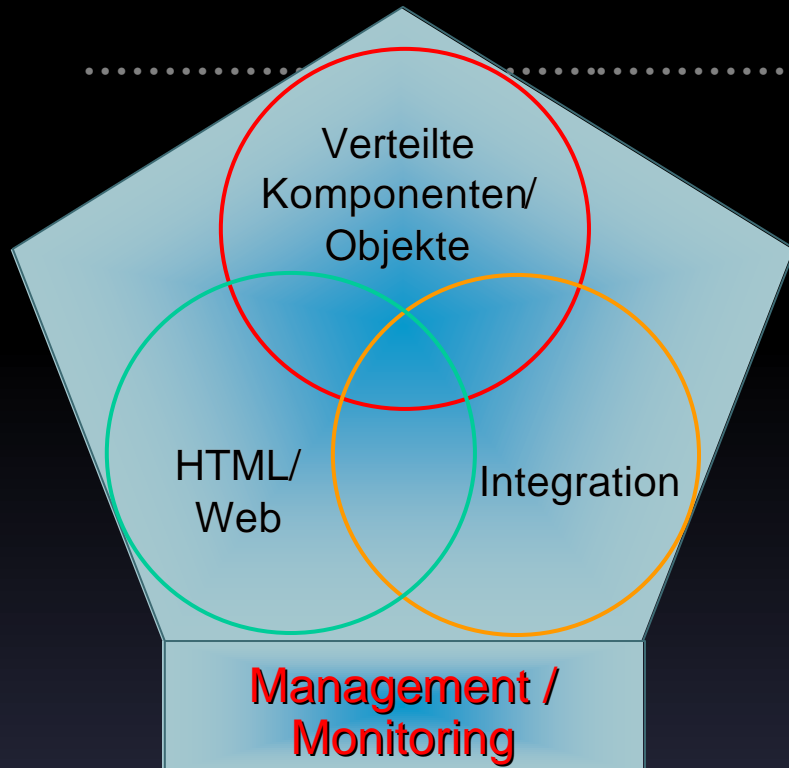
---



**J2EE Applikation-Server mit EJB in Betrieb**  
**Quelle: Meta Information Group, Sept. 2001**



# J2EE: Java auf dem Server



*Server basierend auf  
Industrie-Standard APIs*

**J2EE 1.2 & 1.3  
zertifiziert**

- **Verteilte Objekte / Komponenten**

- Enterprise JavaBeans™ 2.0 (EJB)
- Remote Method Invocation (RMI)
- Java Transaction Service & API (JTS/JTA)
- Java Naming & Directory Interface (JNDI)
- Java Messaging Service (JMS)
- Java Interface Definition Language (JIDL)

- **Web/HTML**

- Servlets
- Servlet session management
- Java Server Pages (JSP) / Java HTML

- **Integration**

- Java Database Connection (JDBC)
- Multi-tier JDBC
- Java Connector Architecture (JCA)

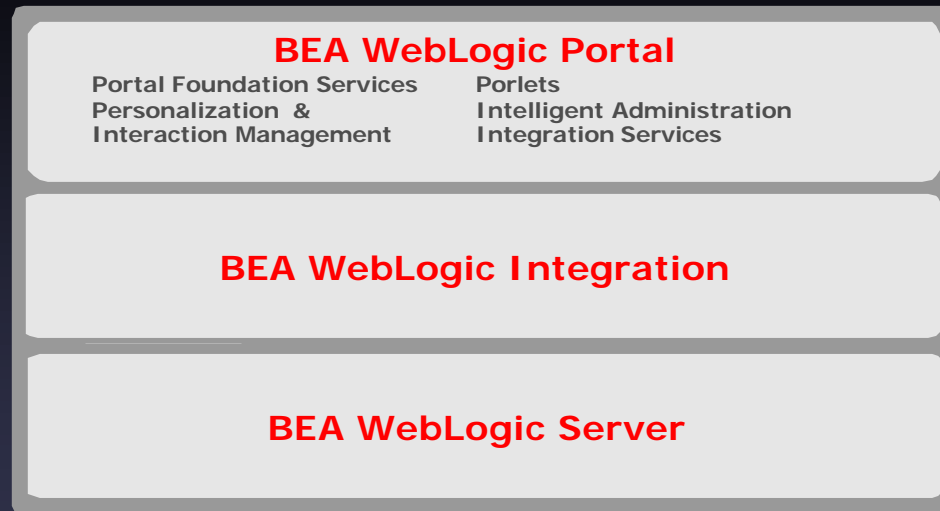
- **Management / Monitoring**

- Java Management Extension (JMX)

# BEA WebLogic Server Basis der E-Business Plattform



BEA  
WebLogic  
E-Business  
Platform





# Inhalt

---

- WebLogic Server 7.0
  - Einführung
  - Einige APIs: EJB, JMS, JCA und JTA
  - Betriebsaspekte
- WebLogic Workshop
  - Einfache Web Services
  - Enterprise Web Services
  - JWS: Enterprise Web Services in Java
  - Grafische IDE

# BEAs EJB Führerschaft: Ihr Time-to-Market

---



- **Führender Enterprise JavaBeans Anbieter**
  - EJB Technologie seit Sept. 97 in Auslieferung
  - EJB 1.0 April 98
  - EJB 1.0 Produktion Juli 98
  - Clustered EJB Nov. 98
  - EJB 1.1 Mai 1999
  - EJB 2.0 Early Adopters Juni 2000
  - EJB 2.0 Generelle Verfügbarkeit Dez. 2000
- **Ausgereiftester skalierbarster EJB Application Server auf dem Markt**





# EJBs: Effektivität und Konsistenz

- Entwickler bekommen
  - Persistenz
    - Relationen, Abstraktion von DBMS, Queries, Lazy Loading, verteiltes Caching, **Relationship-Caching**, **Optimistic Locking**
  - Transaktionen
    - Konsistenz in der Applikation
  - Security
    - ACLs und Laufzeitüberprüfung
  - Naming
    - Lokationstransparenz über Clients und Server hinweg
  - Lifecycle und Caching
    - “Swapping” auf Applikationsebene
  - Concurrency-Modell
    - Keine Low Level Multithread Programmierung
- **Kurzum Mannjahre gespart.....**



# EJBs: Eine hochspezialisierte Familie



## Session Beans

• "TP Monitor" Stil

• Kurzlebig, kein expliziter Schlüssel

• 1-zu-1 Beziehung zu Client

• Expliziter DB Zugriff

• Bsp. E-Commerce

• Server, Bank-Transaktion

## Entity Beans

• "Datenbank" Stil

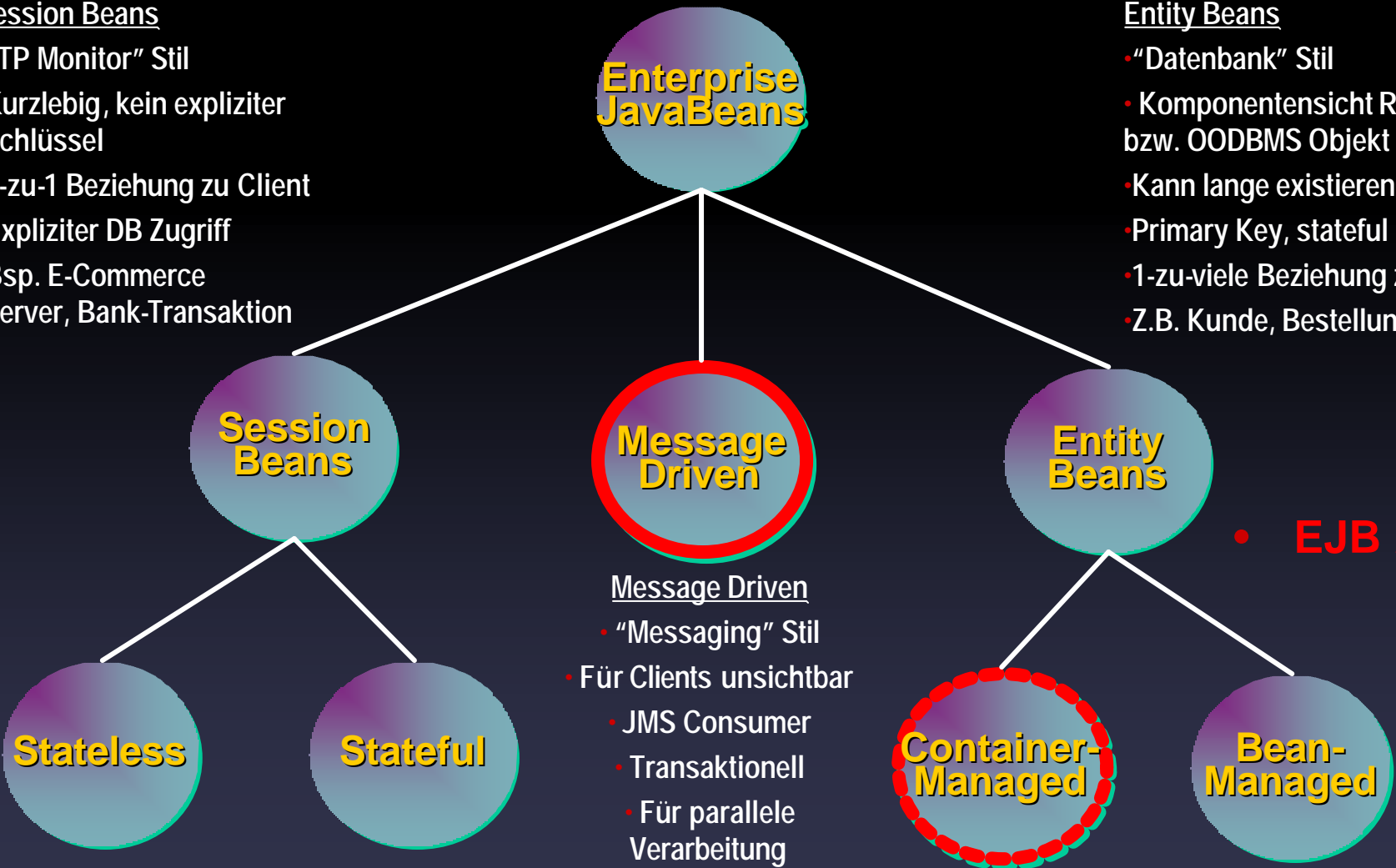
• Komponentensicht RDBMS S  
bzw. OODBMS Objekt

• Kann lange existieren

• Primary Key, stateful

• 1-zu-viele Beziehung zu Client

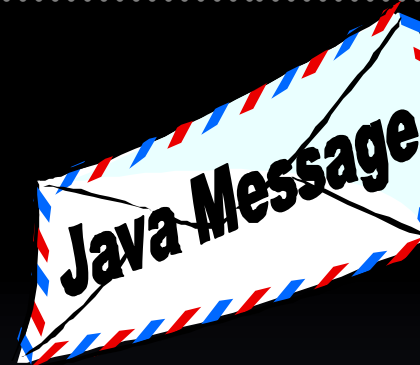
• Z.B. Kunde, Bestellung



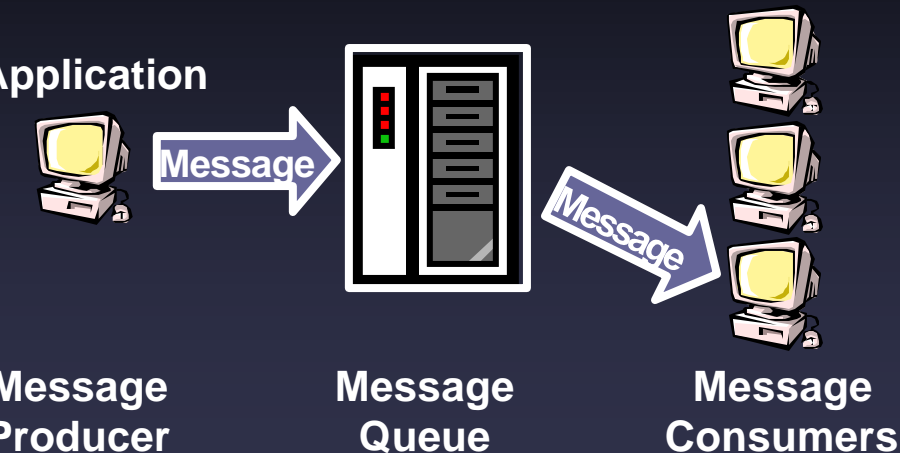
# WebLogic Java Message Service Der Unternehmens Message Bus



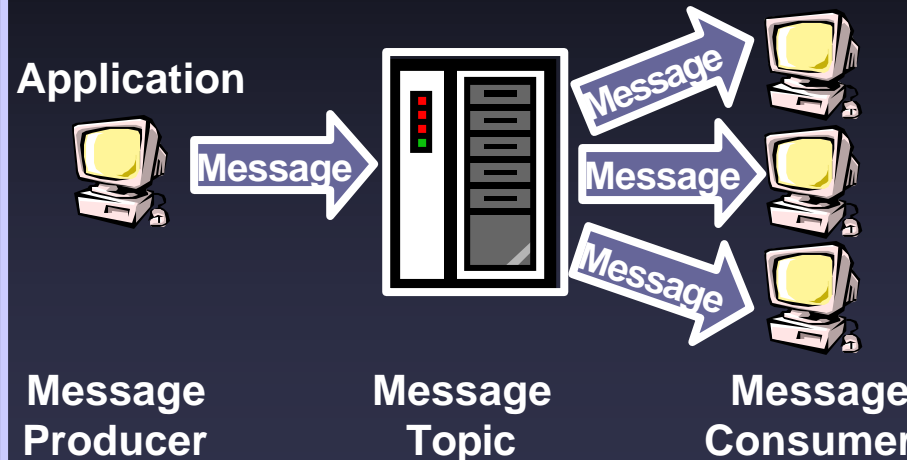
- Standard JavaSoft API (Version 1.0.1) für
  - Publish / Subscribe Model (Pub/Sub)
    - Durable möglich
  - Point-To-Point Model (PTP)
- Optionale Server Session Pools (serverseitiges parallele Verarbeitung von Messages)



*PTP - Modell*



*Pub / Sub - Modell*



# WebLogic Java Message Service Quality of Service

---



- Messages können sein
  - NON-PERSISTENT oder
  - PERSISTENT (DBMS, File)
- JMS Operationen können Teil einer Transaktion sein
  - *Transacted* JMS Sessions zum Mischen von EJB und JMS (optional in Spezifikation)
  - *Non-transacted* JMS Sessions benutzen JMS Acknowledgement
- Message Filterung
  - SQL-artige Syntax:
  - “(product like ‘WebLogic%’ or product like ‘%T3’)
  - BEA XML-Path Erweiterung



# WebLogic Java Message Service Quality of Service

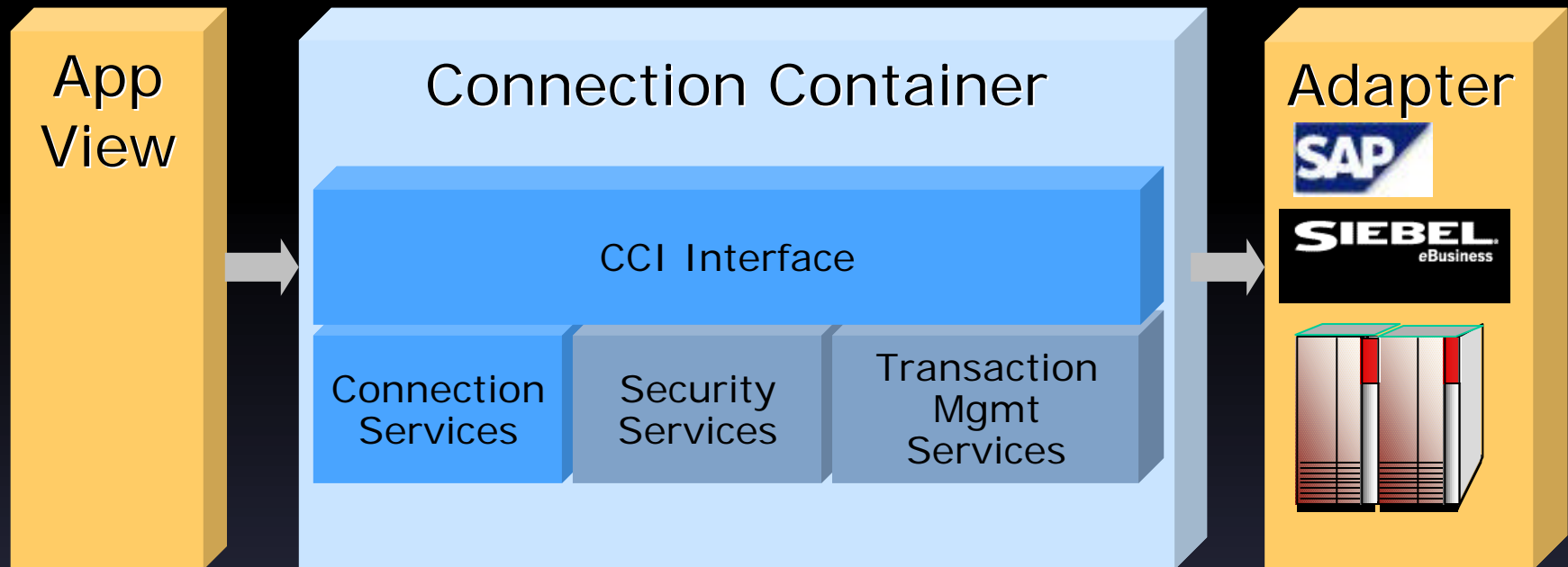
---



- Hochverfügbar
  - Virtuelle Destinations mappen auf mehrere physische Lokationen
  - Loadbalancing innerhalb einzelner Destinations
- BEA High-Performance Delivery
  - Point-to-Point TCP/IP oder Multicast
- **Messaging Bridge für 3<sup>rd</sup> Party MOMs**
  - Mapping von Destinations
  - Reconnection Policies
  - 3 Arten von TX support: XA, Local, None
  - Adapter für IBM MQSeries im Lieferumfang



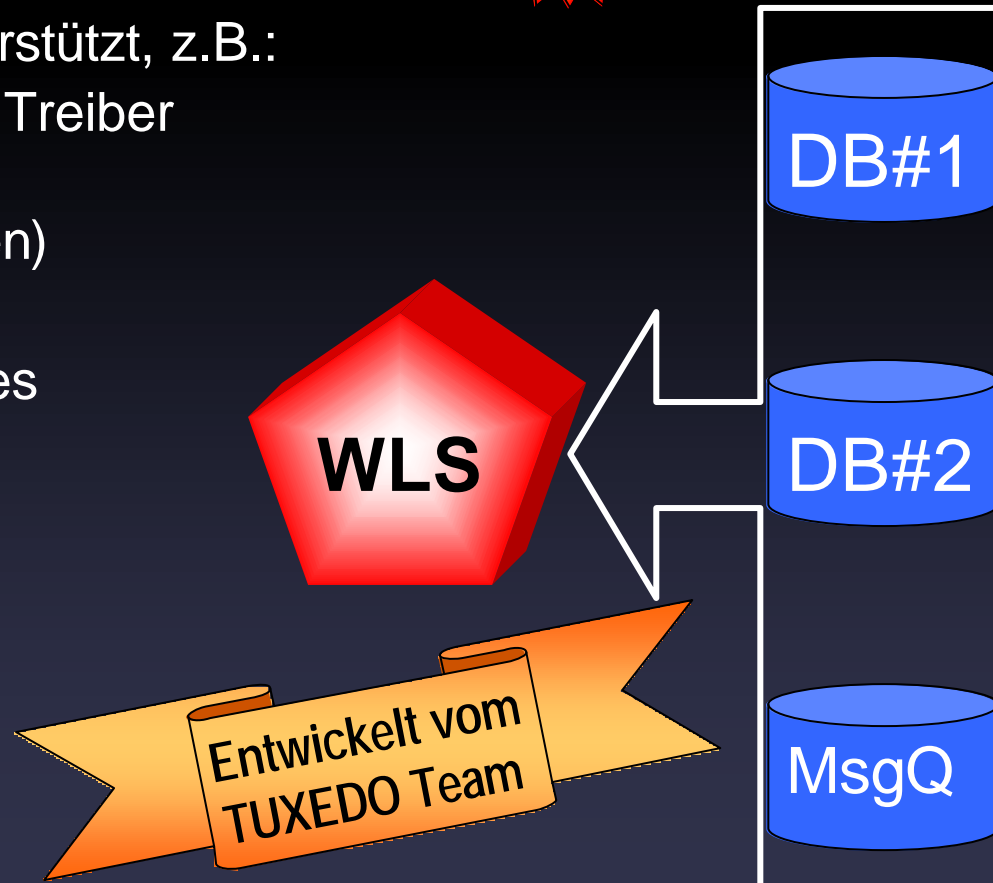
# Das neue EAI Paradigma: Java Connector Architecture



- Teil des J2EE Industriestandards
- Unterstützt Pooling, Security und Transaction Management
- BEA Adapter erhältlich für
  - SAP R/3, Mainframe (CICS / IMS) & Siebel in Vorbereitung
  - Mailadapter & DBMS Adapter als Quelltext

# Garantie für Konsistenz: Distributed Transaction Manager

- Komplett in Java
- 2PC mit jedem Storage Provider, der XA-Resource Interface unterstützt, z.B.:
  - DBMS via JDBC 2.0 XA Treiber (Oracle OOTB)
  - JMS (alle Persistenzarten)
  - EJBs
  - Externe Message Queues (MQ-Series)
  - JCA Adapter
- Transaktionsinfektion
  - über mehrere Server
  - mit & ohne Cluster



# Inhalt

---

- WebLogic Server 7.0
  - Einführung
  - Einige APIs: EJB, JMS, JCA und JTA
  - Betriebsaspekte
- WebLogic Workshop
  - Einfache Web Services
  - Enterprise Web Services
  - JWS: Enterprise Web Services in Java
  - Grafische IDE



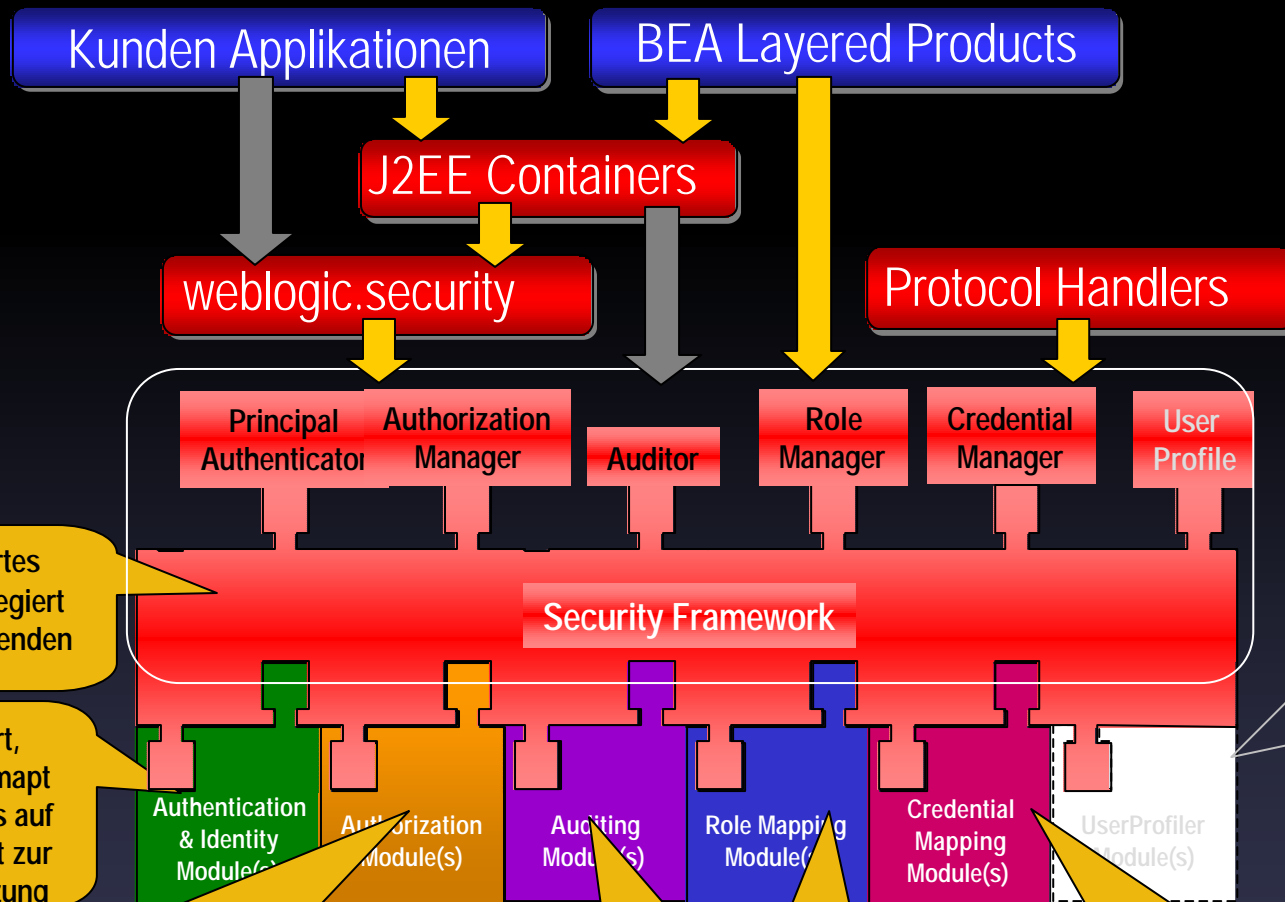


# Modulare Security Infrastruktur

---

- Offenes Set an APIs für:
  - Authentifizierung, Autorisierung, Auditing und PKI Management (JAAS enthalten)
  - 3<sup>rd</sup> party Security Hersteller können sich direkt ins WebLogic Server Framework einhängen
- Zugriffs-Schema
  - wird angewendet auf J2EE und Nicht-J2EE Ressourcen
  - Nicht limitiert auf J2EE DD-basiertes Modell
  - Rollen-basiert
  - Regel getrieben
  - GUI Security Administrations Tool im Lieferumfang
    - Definition von Security Policies
    - Definition von Regeln
    - Keine Code-Änderungen an Applikation

# Security Infrastruktur: Offenes Framework



Security  
Service

Service basiertes  
Framework, delegiert  
Anfrage an passenden  
Provider

Authentifiziert,  
verifiziert und mapt  
Security Tokens auf  
internes Format zur  
SSO Unterstützung

Autorisations-Policies für Ressourcen unter  
Einbeziehung von Business. Policies

Protokolliert alle  
Security Aktionen,  
unterstützt Nicht-

Mapt Rollen auf  
User/Gruppen in  
Abhängigkeit

Mapt Authentifizierungs-  
Credentials eines Users  
auf Legacy Applikation für

Verwaltet und  
stellt breit Profil-  
Attribute für User  
basierend auf  
Privacy Settings

# Security Infrastruktur: Policy Editor Beispiel

**Policy Condition:**

User name of the caller  
Caller is a member of the group  
Caller is granted the role  
Hours of access are between  
**Value of amount**

Add

New...

**Policy Statement:**

Caller is granted the role  
StoreManager or AssistantStoreManager  
and Hours of access are between  
08:00:00 and 19:00:00

Move Up

Move Down

**Parameter Constraint - Microsoft Internet Explorer**

**Allow access to resource:**  
Condition under which access to the resource(s) should be allowed.

If the amount is  the value of \$

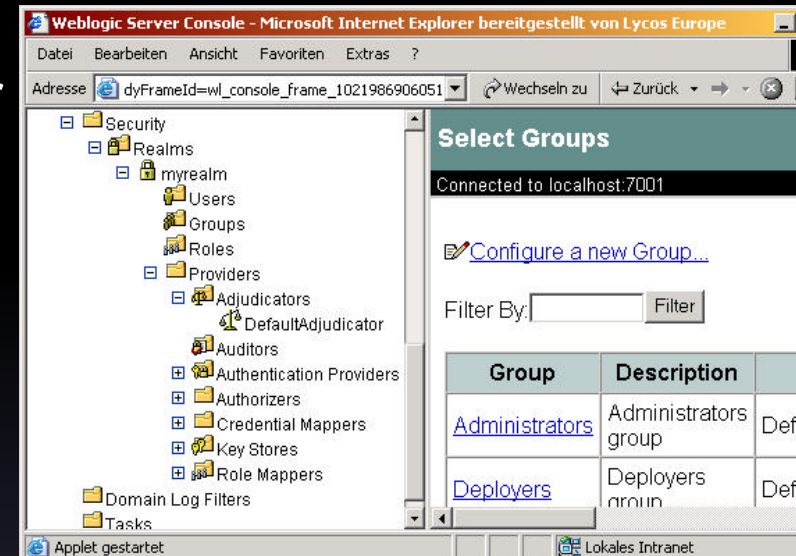
Reset Apply

OK Cancel

# Security Infrastruktur: LDAP basiert

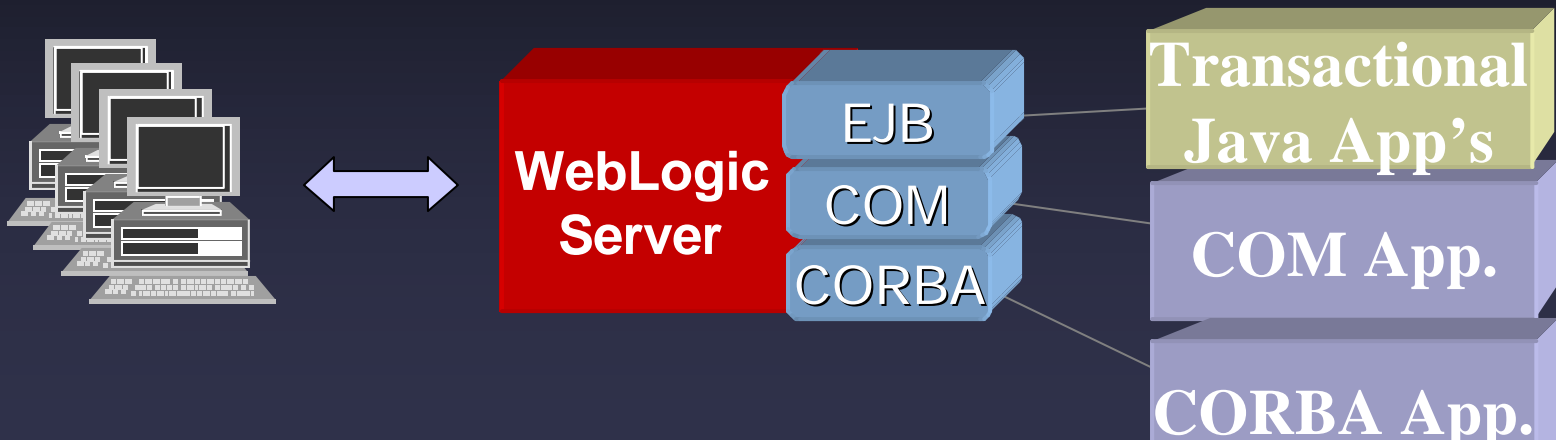


- Eingebauter LDAP v.3 Server
  - Speichert gesamte Profile & Berechtigungen (User, Group, ...)
  - Skalierbar, leseoptimiert
  - Kann als Cache Proxy für existierenden LDAP Server dienen
- Mehrere 3<sup>rd</sup> Party LDAP Server möglich
  - Erstellung eines Unified User Profils
- Ersetzt ACLs
  - Alte ACLs können zur Laufzeit automatisch konsumiert werden



# Integration

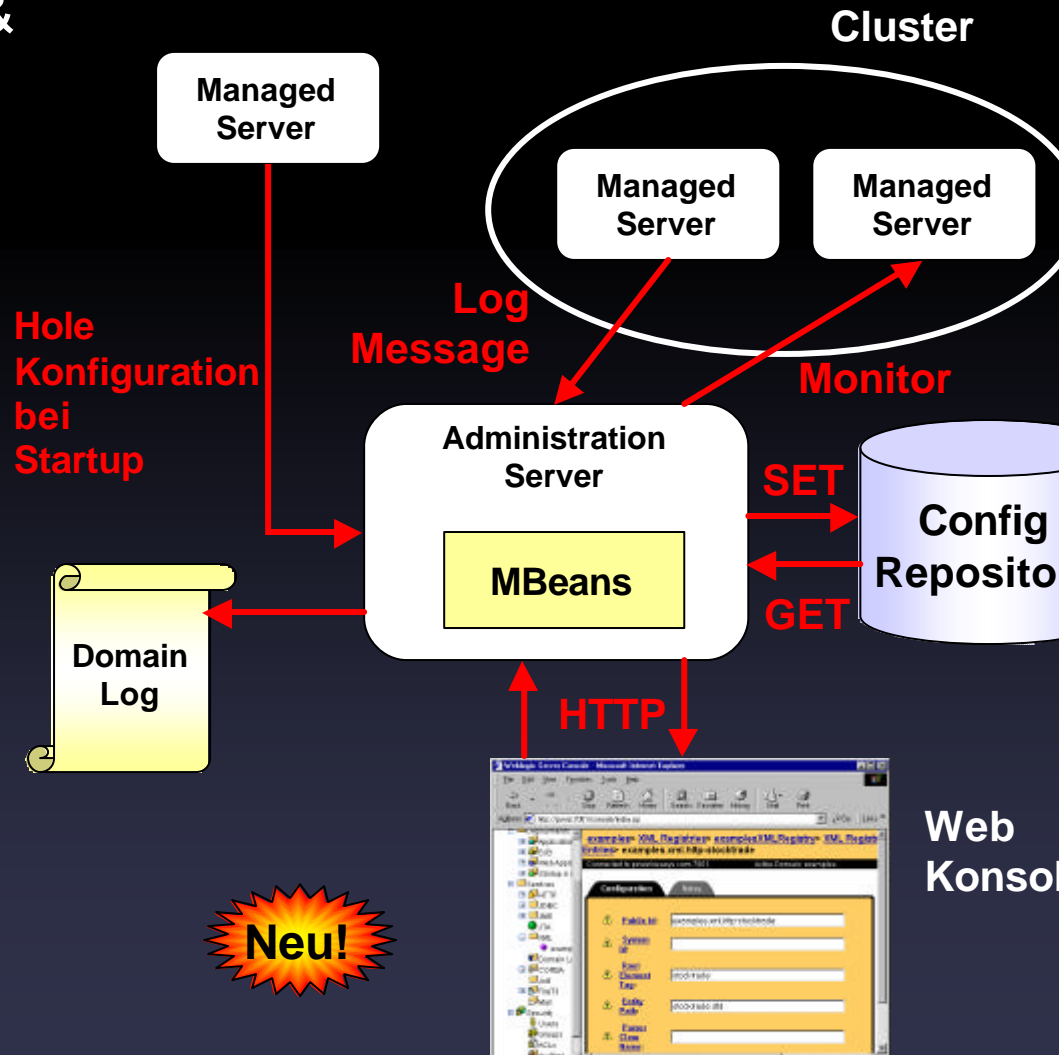
- jCom: Java zu COM bidirektionale Bridge
  - Voll integriert, im gleicher WLS JVM
- RMI /IIOP
  - Bidirektionale TX- und Security Context Propagation
- Zugriff auch von C++ Client
  - Unterstützt WLS Clustering und Fail Over



# Management und Monitoring: Basierend auf JMX Standard



- Zentrale Überwachung & Administration der Domäne
- Zentrales Repository & Log
- Hot-Deploy von EJBs / Servlets / Applications
  - Two Phase Install
- Node-Manager pro Box
  - Startup / Health
- Hot-Stand-By Server
- Basierend auf JMX
  - SNMP Agent
  - Tivoli, BMC Patrol,...



# Inhalt

---

- WebLogic Server 7.0
  - Einführung
  - Einige APIs: EJB, JMS, JCA und JTA
  - Betriebsaspekte
- WebLogic Workshop
  - Einfache Web Services
  - Enterprise Web Services
  - JWS: Enterprise Web Services in Java
  - Grafische IDE



# WebLogic Server 7.0 XML Support

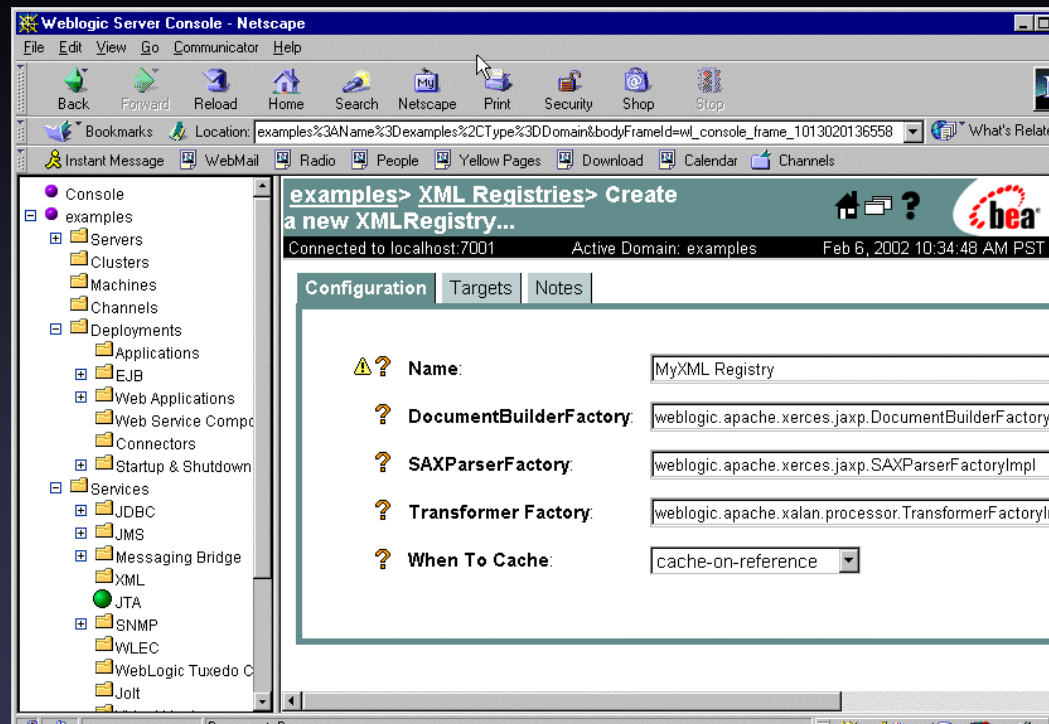
- **XML Registry**

- Austauschbare Parser durch JAXP 1.1
- Final W3C Schema Unterstützung



- **XML Parser**

- SAX V2
- DOM Level 2
- WebLogic Fast-Parser
  - Auch nur Teile parsbar







# WebLogic Server 7.0 XML Support

---

- UDDI Registry (UDDI 2.0)
  - Explorer
- JAX-RPC (0.8) für Clients
  - Automatisches Mappen einfacher SOAP Datentypen
  - Für User-Datentypen werden Serializer- / Deserializer-Klassen generiert
- Tools (Cmd-Line & Ant) für
  - ServiceGen: Macht aus EJBs oder Java-Klassen Web Services (synchron / asynchron)
  - ClientGen: Erzeugt JAX-RPC konformen Java Client
- Web Services Interzeptoren
  - Zugriff auf SOAP Nachricht vor/nach Aufruf





# Inhalt

---

- WebLogic Server 7.0
  - Einführung
  - Einige APIs: EJB, JMS, JCA und JTA
  - Betriebsaspekte
- WebLogic Workshop
  - Einfache Web Services
  - Enterprise Web Services
  - JWS: Enterprise Web Services in Java
  - Grafische IDE

# Corporate Developers

---

## Corporate Developers...

- Entwickeln Anwendungen & UI
- Verbinden diese mit den Legacy-Systemen
  - Databases, COM Komponenten, third-party Libraries, etc.
- Sind im Umgang mit Nutzern einer Anwendung
  - Kennen Anforderungen sehr gut
- Arbeiten mit prozeduralen Programmiersprachen
  - Visual Basic, Power Builder, Perl, PHP, Python, COBOL, etc.



# Corporate Developers

---

- Sind nicht
  - OOP / Design Pattern Experten
  - Enterprise Architektur Spezialisten
    - hochskalierbare ausfallsichere Systeme
  - Systemprogrammierer
    - Details von Communications Stacks, Parsing etc.
  - Routiniert im Umgang mit komplexen feingranularen J2EE APIs
    - JMS, JNDI, EJB, etc.
- Es gibt heute fast 5mal mehr Corporate Developers als professionelle Java Architekten
  - 9 Millionen Corporate Developers
  - 2 Millionen J2EE Architekten

# Enterprise Web Services

---

- Reichen einfache Web Services aus?
  - = einfacher Aufruf mit Rückgabewert
- Benötigt wird
  - Asynchronität
    - Zielsystem langsam (z.B. Mensch)
    - Zielsystem zeitweise nicht erreichbar (z.B. Batch)
  - Lose Kopplung
    - Änderung der Web Services Implementation darf Aufrufer nicht tangieren
    - XML ist nicht hinreichend
  - Business Level
    - Grobkörnige Interaktion



# Enterprise Web Services in BEA WebLogic Workshop

---

- Asynchronität
  - Konversationen
    - Persistenter Zustand (Entity Beans)
  - Message Buffer
    - JMS darunterliegend
- Lose Kopplung
  - XML Maps
  - Entkopplung des Kontrakts zwischen XML und Java Signatur
  - Protokolle: SOAP oder XML over HTTP, JMS
- Business Level
  - XML Dokumente



# Inhalt

---

- WebLogic Server 7.0
  - Einführung
  - Einige APIs: EJB, JMS, JCA und JTA
  - Betriebsaspekte
- WebLogic Workshop
  - Einfache Web Services
  - Enterprise Web Services
  - JWS: Enterprise Web Services in Java
  - Grafische IDE



# Ein neues Paradigma in J2EE: Deskriptive Sprachen

---

- Deskriptive Sprachen
  - beschreiben das WAS und nicht das WIE
  - Manchmal auch als 4GL bezeichnet
  - Bekanntes Beispiel: SQL
- Vorteile
  - Prägnanter, abstrakter
  - Leichter lesbar
  - Höhere Produktivität
  - Eignen sich für “Corporate Developers”





# Ein neues Paradigma in J2EE: Deskriptive Sprachen

---

- Verwendung innerhalb von J2EE
  - Prä-Compilierung in Java
  - Deskriptive Elemente werden als Java-Kommentar dargestellt
  - Ähnlich zu JSPs
    - “Deskriptive” Tag-Libraries
- Nachteil
  - Präcompilierung erschwert Debugging, wenn nicht eine Entwicklungsumgebung vorliegt, die deskriptive Sprache unterstützt



# Deskriptive Sprache für Java Enterprise Web Services: JWS

---

- JWS Files sind
  - Normale Java Klassen, die die Logik eines Enterprise Web Services enthalten
  - WLW Javadoc Kommentare zur Kommunikation
    - Konversationen
    - Puffer
    - Etc.
  - Zukünftiger Teil des J2EE Standards
    - JSR 175 vom 25.März 2002



# Leistungsfähigkeit von JWS: Ein Beispiel

```
/**
 * @jws:operation
 * @jws:conversation phase="start"
 * @jws:message-buffer enable="true"
 */
public void getQuote(int age, String gender, int policyAmount)
```

## Erzeugen des Web Services

- Objekt erzeugen (Bean)
- SOAP Marshalling
- Lesen / Schreiben WSDL
- XML Parsing
- ...

80 Lines

## Konversationen verwalten

- Zustand verwalten (Entity Beans)
- Automatische Nachrichten Korrelation
- Datenbankzugriff (JDBC)

50 Lines

## Implizite J2EE Verwendung

- Skalierbare Web Services
- Verwendung von JNDI, JMS

80 Lines

# Controls: Überblick

---

- Bekannter Ansatz aus Entwicklungsumgebungen
- Bündeln feingranulare APIs zu einem Objekt
  - Gängige Verwendungen im Vordergrund
- Control sieht immer aus wie lokales Objekt mit Methoden und Callbacks
- Das Verhalten von Controls wird durch Zuweisung von Eigenschaften bestimmt

## Controls: Technisch

---

- Jedes Control besteht aus einem Control-File
- Controls Files definieren ein Java Interface
  - Ergänzt um WLW Kommentare
- Leiten sich ab von fünf Super-Interfaces
  - Database, Web Service, Timer, JMS, EJB
- Das erzeugte Control bezeichnet eine anwendungsspezifische Ausprägung
  - Bsp. Database: Update der *Kunden*-Tabelle
  - Bsp. EJB: Aufruf des *Konto*-Beans



# Controls: Technisch

## Beispiel Timer Control

---

- Deklaration

```
/**  
 * @jws:control  
 * @jws:timer timeout="5 seconds"  
 */  
private TimerControl myTimer;
```

- Steuern

```
myTimer.start();  
myTimer.stop();
```

- Callbacks implementieren

```
private void myTimer_onTimeout(long arg0) throws Exception {  
    System.println("5 Sekunden sind um");  
}
```

# Inhalt

---

- WebLogic Server 7.0
  - Einführung
  - Einige APIs: EJB, JMS, JCA und JTA
  - Betriebsaspekte
- WebLogic Workshop
  - Einfache Web Services
  - Enterprise Web Services
  - JWS: Enterprise Web Services in Java
  - Grafische IDE



# BEA WebLogic Workshop: Die IDE

---

WebLogic Workshop besteht aus zwei Hauptkomponenten:

- Visueller Entwicklungsumgebung
  - Design View: Graphische Metaphern
  - Source View: Code
  - Änderungen werden immer synchron geladen
- Run-Time Framework
  - Führt alles als EJB, JMS, JNDI Code aus
  - Testing, Debugging

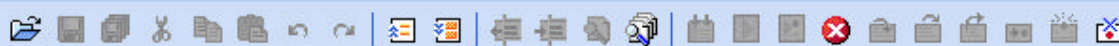




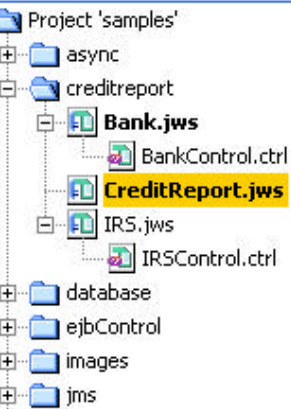
# WebLogic Workshop: Design View

CreditReport.jws - BEA WebLogic Workshop

File Edit View Service Debug Tools Window Help



Project Tree



Structure Pane

CreditReport

Design View

Source View

Add Operation

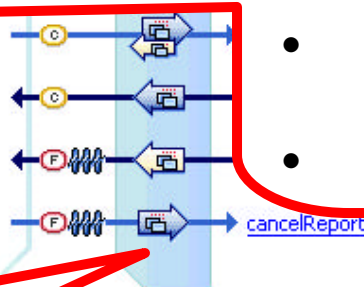
## Projekt Fenster

- Zeigt Dateien des aktuellen Projekts
- Projekt = Web-App

## Design Canvas

- Web Service in der Mitte
- Client links
- Controls rechts

CLIENT



SERVICE

SERVICE

Member Variables

String taxReport

String bankReport

CreditReport.jws

Properties - CreditReport

Name	CreditReport
target-namespace	
conversation-lifetime	
protocol	
wsdl	
schema	

Description

CreditReport Service

This is the web service that you are creating. You can add methods to the service to exchange messages with a client. You can

Tasks

- ♦ Add a new method
- ♦ Add a new callback

Server Running

Ln 1 Col 1



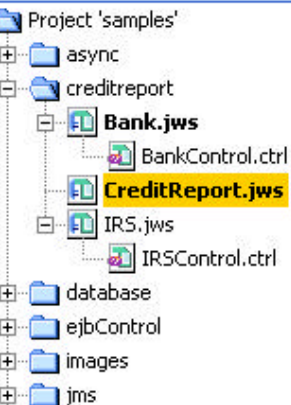
# WebLogic Workshop: Design View

CreditReport.jws - BEA WebLogic Workshop

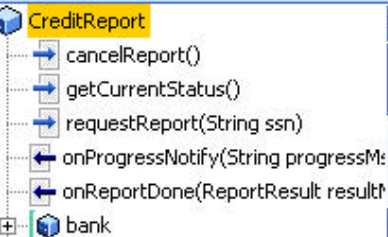
File Edit View Service Debug Tools Window Help



Project Tree



Structure Pane



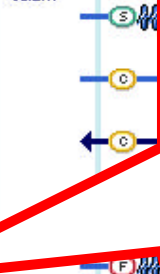
Service Running

Design View

Source View

Add Operation

CLIENT



requestTaxRes

SERVICE

SERVICE

SERVICE

## Struktur Fenster

- Zeigt Methoden, Variablen, Klassen, Controls des aktuellen Web Service

## Eigenschaften Fenster (selektiertes Objekt)

- **Properties:** Eigenschaften
- **Description:** Beschreibung
- **Tasks:** Mögliche Aktionen

CreditReport.jws

Properties - CreditReport

Name	CreditReport
target-namespace	
conversation-lifetime	
protocol	
wsdl	
schema	

Description

CreditReport	Service
This is the web service that you are creating. You can add methods to the service to exchange messages with a client. You can	

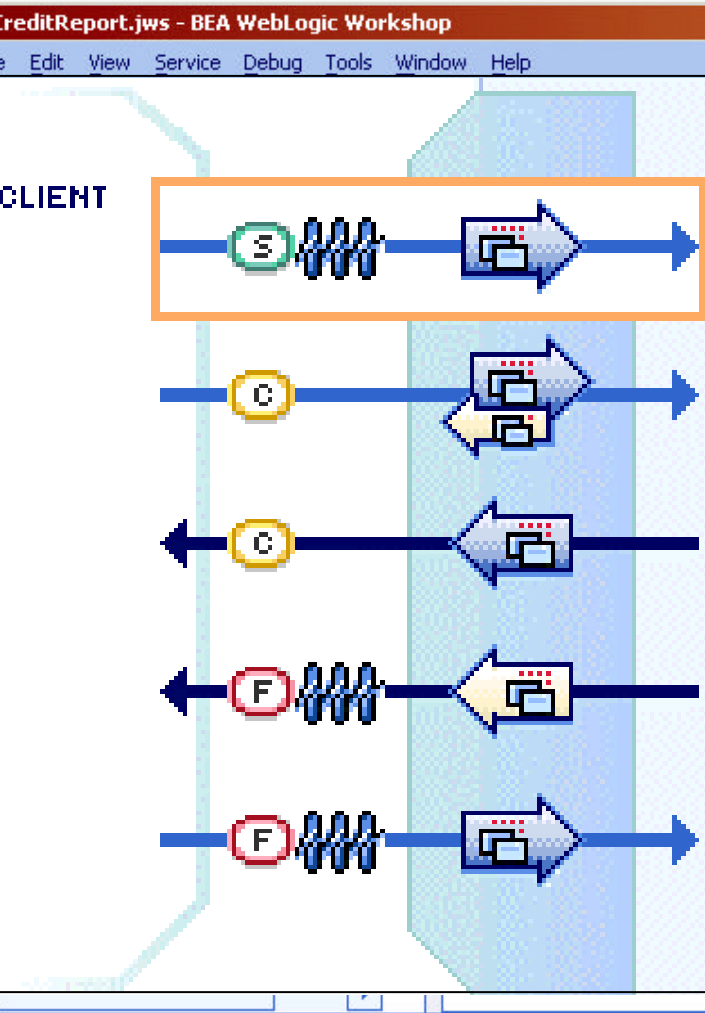
Tasks

- ♦ Add a new method
- ♦ Add a new callback

Ln 1

Col 1

# WebLogic Workshop: Konversationen & Asynchronität



## Edit Maps and Interface

### XML:

Parameter XML

Return XML

This output XML message maps from the return value of the operation.

Map Type: ☐ Default ☒ Custom

```
<creditReport>
  <bankReport>{return.bank}</bankReport>
  <taxReport>{return.tax}</taxReport>
</creditReport>
```

### Java:

```
public ReportResult getCurrentStatus()
```

Help

OK

Cancel

# WebLogic Workshop: Source View

## Source View

- Syntax-Helper
- Debugging
- Breakpoints
- Watch-Variables
- U.v.m

creditReport.jws - BEA WebLogic Workshop

Edit View Service Debug Tools Window Help

Design View Source View

CreditReport (Definition)

```

154 {
155     // Request information from the bank
156     bank.
157     // ba
158
159     // Re
160     irs.r
161
162     retur
163 }
164
165 /**
166  * <p>Req
167  *
168  * <p>For our impatient customers, we can give them
169  * partial results whenever they ask for them.</p>

```

interface creditreport.BankControl

Method	Return Type
Callback	class
cancelAnalysis()	void
equals(Object)	boolean
getClass()	Class
getInputHeaders()	Element[]
getPassword()	String
getUsername()	String
hashCode()	int

Errors

File	Line	Message
CreditReport.jws	156	No variable irs defined in interface creditreport.BankControl.

Build complete - 1 error(s), 0 warning(s)

Errors Find in Files

Server Running Ln 156 Col 14



# WebLogic Workshop: Test View

- Zusammenfassung aller Web Service Operationen
- Automatische Proxies zum Aufrufen
- Logs aller Nachrichten in und out
- U.v.m

## Setup.jws Web Service

Overview Console Test Form

http://localhost:7001/eBike/Solutions/Exercise04/Setup.jws

**Public Information**  
about Setup.jws Web Service [See other services in this project](#)

### Web Service Description Language files

[Complete WSDL](#) This WSDL file describes the complete public contract of Setup.jws, including both operations and callbacks.

[Callback WSDL](#) Some software cannot automatically handle the callbacks described in the complete WSDL. This callback WSDL describes the service to be implemented by clients that wish to receive callbacks by implementing a callback listening service.

### Client source code

[JWI](#) Source code for a Service Control JWI that can be used by a Cajun client to communicate with this service.

[Java Proxy](#) Source code for a pure Java proxy that can be used to communicate with this service.

### Service Description

This web service implements the following operations:

<a href="#">start</a>	The <b>first</b> sentence of javadoc describing Setup.jws, if any. <b>Need to extend data structures to store this info.</b>
<a href="#">progress</a>	The <b>first</b> sentence of javadoc describing Setup.jws, if any. <b>Need to extend data structures to store this info.</b>
<a href="#">finish</a>	The <b>first</b> sentence of javadoc describing Setup.jws, if any. <b>Need to extend data structures to store this info.</b>
<a href="#">nukeTables</a>	The <b>first</b> sentence of javadoc describing Setup.jws, if any. <b>Need to extend data structures to store this info.</b>

This web service sends the following callbacks:

### Useful links

For more details on WSDL, see the [WSDL Specification v1.1](#)  
For more details on SOAP, see the [SOAP Specification v1.1](#)  
For more details on XML namespaces, see the W3C recommendation on [Namespaces in XML](#)  
For more details on URIs, see [RFC 2396](#)





## Design Time

Run-Time



**How Business Becomes E-Business™**