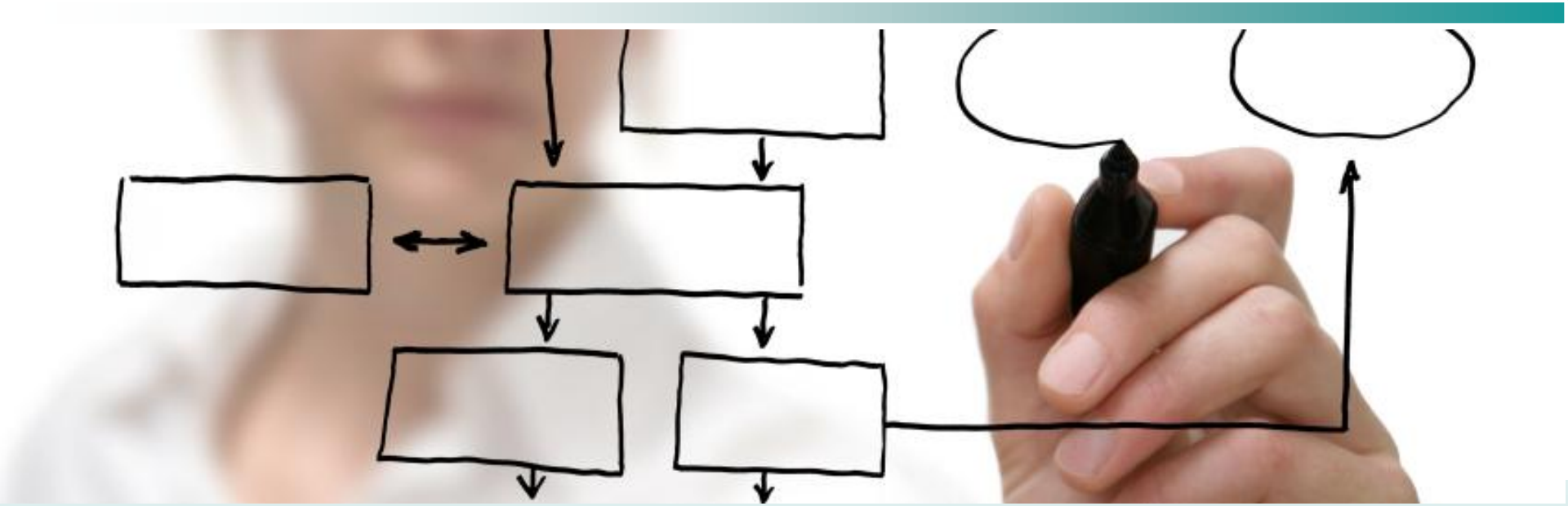
PROMATIS

ORACLE Platinum Partner



# Datenbankzugriff aus Eclipse Rich-Client-Anwendungen über das Internet

Johannes Michler, PROMATIS software GmbH
DOAG 2010, Nürnberg, 16. November 2010

# Gliederung

PROMATIS

ORACLE Platinum Partner

- Eclipse und Eclipse RCP
- Komponentenframework OSGi
- Beispielszenario Geschäftsprozessmodellierung
- Persistenz und Java
- JPA und EclipseLink
- Webservices in Java
- Single-Sourcing mit Eclipse Equinox
- Fazit

# Eclipse: Historie

- Integrierte Entwicklungsumgebung für Java
- Ursprünglich von IBM entwickelt
- Mittlerweile Open Source (Eclipse Public License)
- Seit Juni 2010: Version 3.6
- Für alle gängigen Plattformen verfügbar
  - Windows
  - Linux
  - Mac
  - …

# Eclipse: Java IDE

PROMATIS

ORACLE Platinum Partner

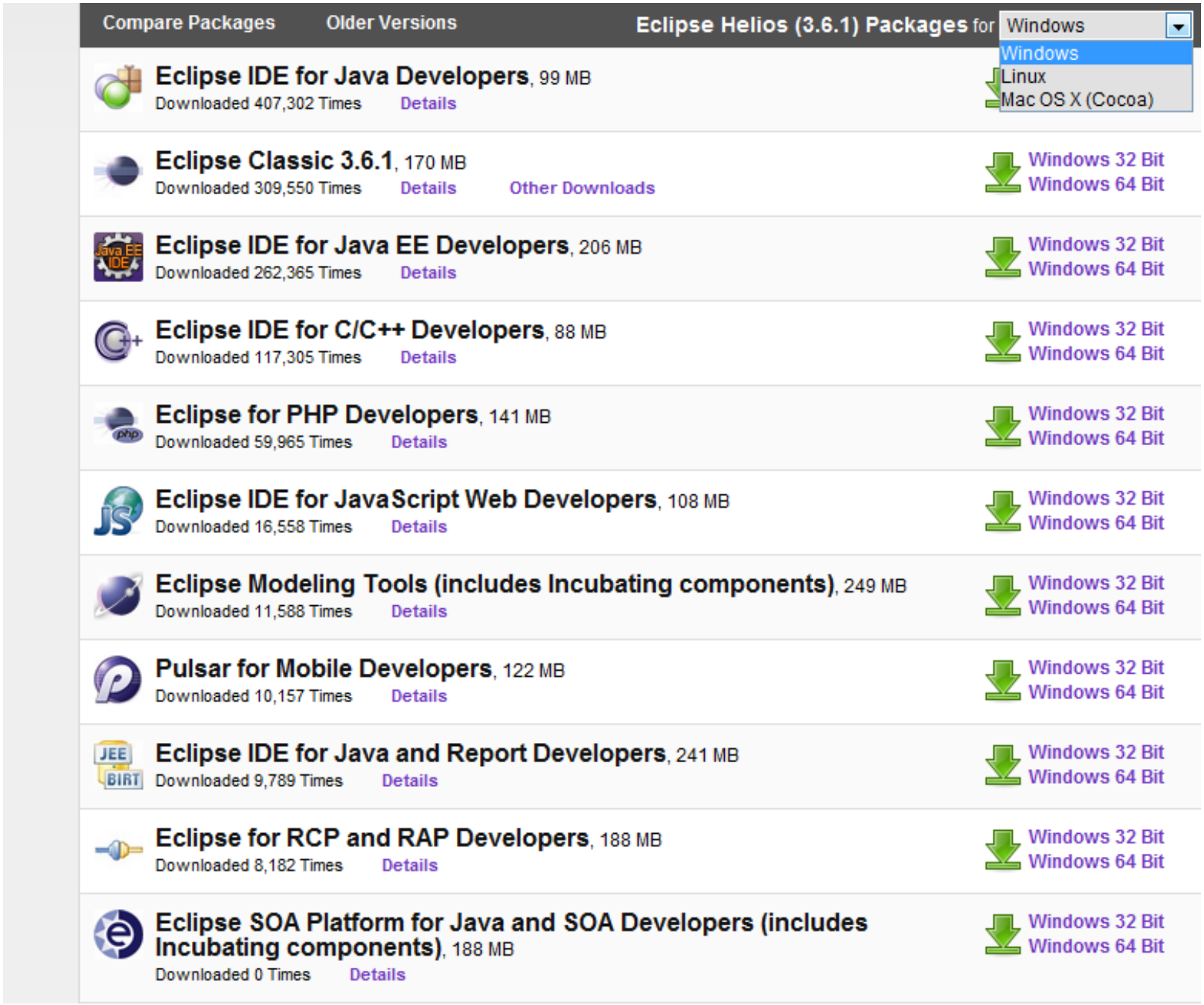
- Java Entwicklungsumgebung
  - Syntax-Überprüfung und –Hervorhebung
  - Code-Vervollständigung und -Formatierung
  - Vielfältige Restrukturierungsmittel
    - Automatische Umbenennung von Paketen, Klassen, Methoden
    - Extraktion von Klassen, Methoden
    - Generierung von Konstruktoren, Get- und Set-Methoden
    - Unterstützung bei der Internationalisierung
  - Komfortabler Debugger

# Eclipse als Plattform

PROMATIS

ORACLE Platinum Partner

- Vielzahl von Erweiterungen vorhanden
  - Java EE und Webanwendungen
  - PHP
  - C/C++
  - Modellierung
  - Reports (BIRT)
  - Mobile Anwendungen
  - Ajax Anwendungen (RAP)
  - Datenbanken (Data Tools Platform)
  - …
- Insgesamt über 1500 Plugins (http://eclipse-plugins.2y.net/)


Compare Packages
Older Versions
Eclipse Helios (3.6.1) Packages for Windows
Windows
Linux
Mac OS X (Cocoa)
Eclipse IDE for Java Developers, 99 MB
Downloaded 407,302 Times Details
Eclipse Classic 3.6.1, 170 MB
Downloaded 309,550 Times Details Other Downloads
Windows 32 Bit
Windows 64 Bit
Eclipse IDE for Java EE Developers, 206 MB
Downloaded 262,365 Times Details
Eclipse IDE for C/C++ Developers, 88 MB
Downloaded 117,305 Times Details
Eclipse for PHP Developers, 141 MB
Downloaded 59,965 Times Details
Eclipse IDE for JavaScript Web Developers, 108 MB
Downloaded 16,558 Times Details
Eclipse Modeling Tools (includes Incubating components), 249 MB
Downloaded 11,588 Times Details
Pulsar for Mobile Developers, 122 MB
Downloaded 10,157 Times Details
Eclipse IDE for Java and Report Developers, 241 MB
Downloaded 9,789 Times Details
Eclipse for RCP and RAP Developers, 188 MB
Downloaded 8,182 Times Details
Eclipse SOA Platform for Java and SOA Developers (includes Incubating components), 188 MB
Downloaded 0 Times Details

# Beispielszenario: Modellierungswerkzeug

PROMATIS

ORACLE Platinum Partner

- Geschäftsprozessmodellierungs-Tool „Horus Business Modeler“
- Basis: Eclipse RCP
  - Nutzung von Eclipse-Plugins wie Eclipse GMF, GEF
  - Damit: Einfache Entwicklung des Editors
- Thema heute: Persistierung der Modelle
- Anforderungen:
  - Speicherung in Oracle-DB (XML-Modelle und Reports mit XMLDB)
  - Zugriff auf Modelle auch über das Internet
  - Breite Skalierung bzgl. der Nutzerzahl:
    - Nur 1 Benutzer mit lokal installierter Datenbank
    - Wenige Benutzer mit eigener Benutzer-Infrastruktur
    - Integration in restliche IT (LDAP-Anbindung)
  - Bei mehreren Benutzern: Zugriff muss synchronisiert werden

# Eclipse: Architektur

- Leichtgewichtiger Kern basierend auf Eclipse Equinox (OSGi)
- Darin eingebunden: Menge von Plugins
- Diese bilden beispielsweise die Java IDE

- Erweiterungsmöglichkeiten
  - Erstellen eigener Plugins für Eclipse
  - Bauen eigenständiger Anwendungen (RCP)
    - Diese bestehen wiederum aus Plugins

# Eclipse-Plugins: Details

PROMATIS

ORACLE Platinum Partner

- Plugin=Jar-File mit MANIFEST.MF
- Manifest spezifiziert:
  - Versionierungseigenschaften
  - Benötigte Laufzeitumgebung
  - Abhängigkeiten
    - Benötigte Bundles/Java-Packages
    - Exportierte Java-Packages
- Equinox sorgt dann für
  - Korrektes Auflösen der Abhängigkeiten
  - Laden der richtigen Klassen => Keine „Classloader-Hell“
  - Dynamisches Nachladen

# Datenbankanbindung

PROMATIS

ORACLE Platinum Partner

- SQL-Verbindungen
  - Öffnen einer JDBC Verbindung
  - Absetzen von Select, Update, Insert,… Anweisungen
- Nachteile:
  - Oft mühsam
  - Manuelles und Fehlerträchtige Erstellung der SQL-Anweisungen
  - Gefahr von SQL-Injection
  - Stored Procedures oft plattformspezifisch
- Vorteile:
  - Sehr flexibel: beliebige SQL Anweisungen, Aufruf von PL/SQL
  - Theoretisch optimale Performance

# Objekt-Relationale Abbildungen

PROMATIS

ORACLE Platinum Partner

- (Automatische) Abbildung zwischen
  - Java-Objekten und
  - Relationalen Tabellen

- Java Persistence API (JPA)
  - Persistenzteil von EJB 3
  - Offizielle Brückentechnologie Java – relationale Datenbanken
  - Standard der mehrere Implementierungen besitzt:
    - Hibernate
    - Apache OpenJPA
    - EclipseLink (ehemals Oracle TopLink)

# JPA: Entitäten

PROMATIS

ORACLE Platinum Partner

- Zentraler Punkt der JPA-Spezifikation
- Spezielle Form von Klassen
  - Können persistente Instanzen besitzen
  - Deklaration über @Entity Annotation
  - Benötigen argumentlosen Konstruktor
  - Dürfen nicht final sein
- Beziehungen zwischen Entitäten:
  - 1:1 @OneToOne
  - N:1 @ManyToOne
  - 1:N @OneToMany
  - N:M @ManyToMany
  - In Java: Abbildung mittels Behältertypen (Liste, Menge)
  - In der Datenbank: Fremdschlüssel oder Join-Tables

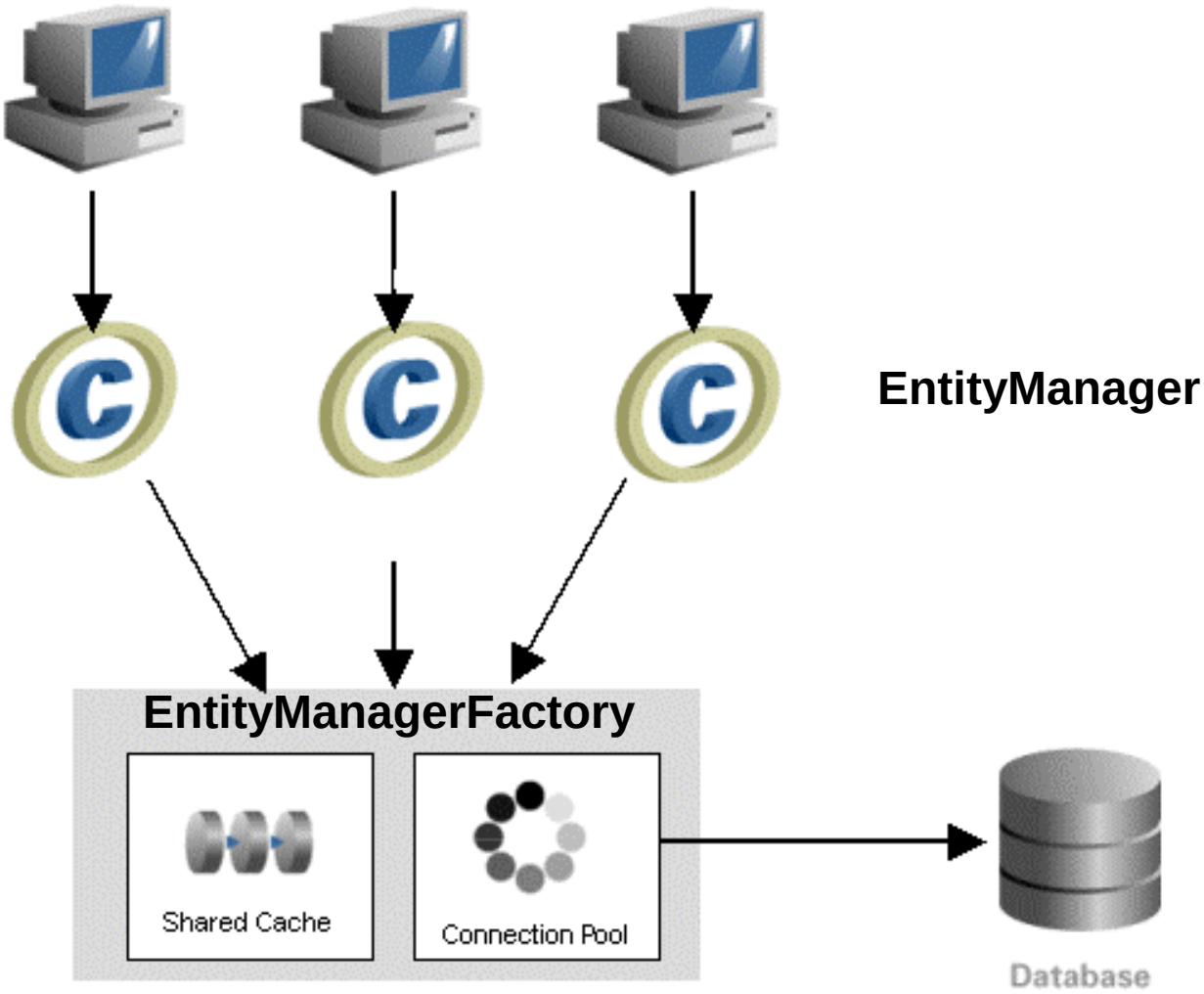
# JPA: Beispiel

PROMATIS

ORACLE Platinum Partner

```
@Entity
public class HorusUsers {
    @Id
    @GeneratedValue(generator = "HorusSeq")
    private long id;
    private String firstName;
    private String lastName;
    private byte[] passwordHash;
    private Date createDate;
    @OneToMany(cascade=CascadeType.ALL, mappedBy="m_group")
    private List<Group> groups;
}
```

# JPA: EntityManagerFactory

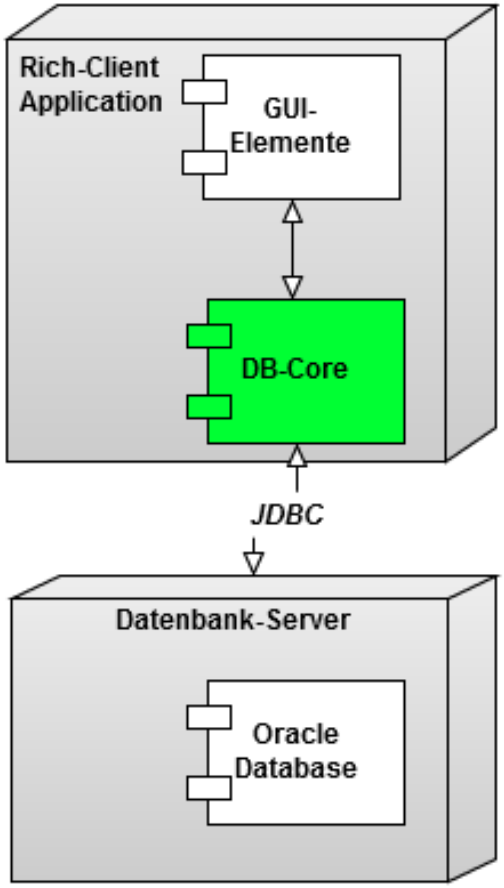


# Zwischenfazit

- Eclipse-RCP:
  - Einfaches erstellen von Rich-Client-Anwendungen
  - Modularisierung mit Hilfe von OSGi-Bundles
  - Vielfältige Basis-Plugins vorhanden
- EclipseLink:
  - Objektrelationale Abbildung JPA
  - Einfaches Persistieren  von Java-Objekten
  - Damit: Umsetzung einer DB-Zugriffsschicht
- Herausforderung:
  - Rich-Clients nicht nur im Unternehmensnetz
  - Mehrere Rich-Clients greifen gleichzeitig zu

# Ansatz 1: JDBC übers Internet

- DB-Zugriffslogik mit EclipseLink
- Zugriff von EclipseLink zur Datenbank über das Internet (TCP/IP)



# Ansatz 1: Nachteile

- EntityManagerFactory können nicht cachen
- Verteilte Transaktionen schwierig (Synchronisierung der Clients)
- Sehr viele Zugriffe über das Netz (für jedes Statement ein Zugriff)
- Dadurch übers Internet langsam
- Außerdem: Firewall erlaubt oft nur HTTP
- Sicherheit: DB-Zugangsdaten im Client
  - Damit: Jeder Benutzer hat theoretisch Zugriff auf alle Sätze

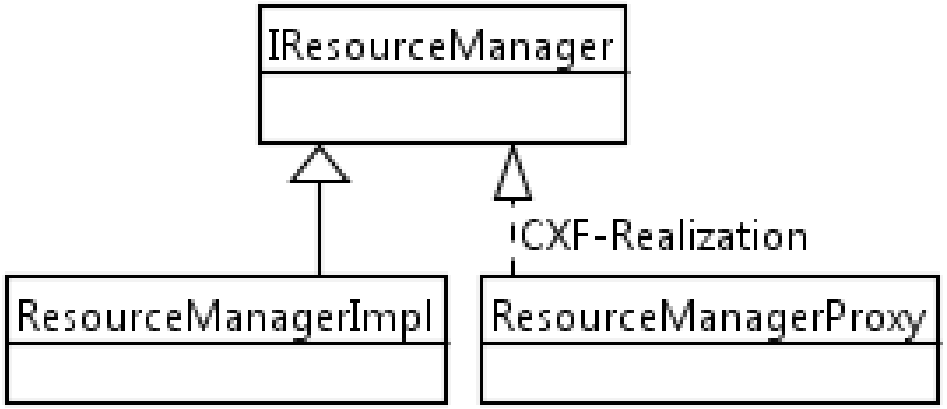
- Fazit: Eher geeignet wenn nur ein Benutzer auf der Datenbank

# Exkurs: Webservices

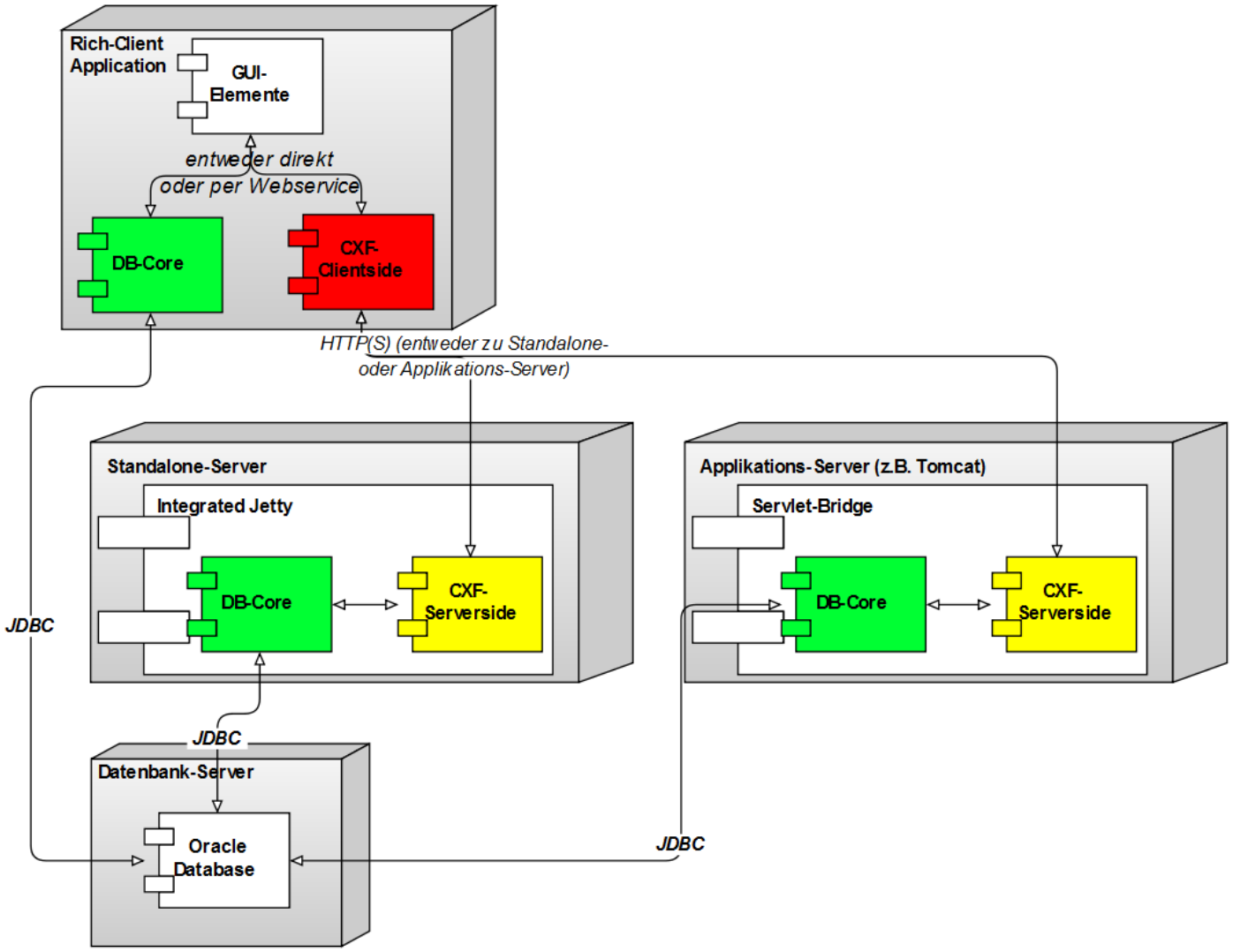
- Standardisiertes Protokoll zum Zugriff auf entfernte Dienste über HTTP
- Im Java-Umfeld: JAX-WS
- Verbreitete Implementierung: Apache CXF
- Bietet:
  - Bereitstellung eines Dienstes
  - Aufruf eines entfernten Dienstes
  - Verbreitetes Protokoll: SOAP über HTTP
- Remote-Zugriff größtenteils transparent
  - Aber: Call-By-Copy Semantik für Methodenaufrufe

# Apache-CXF: Client

- Client ruft entfernten Service über eine Proxy-Klasse auf
- Proxy wird durch Apache-CXF bereitgestellt
- Implementiert Schnittstelle mit den Methoden des Dienstes
- Schnittstelle kann
  - aus der WSDL generiert werden
  - Oder auf Client und Server gemeinsam verwendet werden



# Ansatz 2: 3-Tier-Architektur



# Umsetzung mit Eclipse

- OSGi-Framework ermöglicht Wiederverwendung von „DB-Core“:
- Dasselbe DB-Core-Plugin entweder im Rich-Client oder auf Server (Keine Anpassungen oder Neu-Kompilieren nötig)
- Automatisch Abhängigkeitsauflösung durch OSGi

- Zwei Varianten möglich:
- Entweder in eigener Server-Anwendung:
  - Basierend auf Eclipse Equinox
  - Mit integriertem Jetty-Webserver
- oder als Webanwendung in einem Applikationsserver (WAR)
  - Servletbridge stellt Dienste des App-Servers bereit
  - Verwendung verschiedener Servlet-Container möglich
- In beiden Fällen: Identischer Code für CXF-Client und -Server

# Implementierung: Serverseite

PROMATIS

ORACLE Platinum Partner

```
ServiceReference sr=bndl.getServiceReference(HttpService.class.getName());

CXFNonSpringServlet srvlet = new CXFNonSpringServlet();

if (sr!=null) //Aktivierung in einem App-Server über Servlet-Bridge
 (HttpService)bndl.getService(sr).registerServlet("/",srvlet,null,null);
else{
  org.mortbay.jetty.Server server = new Server();
  Context jettyContext = new Context( server, "/");
  jettyContext.addServlet(srvlet,"/horus/*");
  server.start();
}

ServerFactoryBean factory = new JaxWsServerFactoryBean();
factory.setBus( srvlet.getBus());
JPA_DB_Access realService = new JPA_DB_Access(); //The service to expose
factory.setServiceBean( refManager);
factory.setAddress( "/dbaccess");
factory.create();
```

# Absicherung der Kommunikation

PROMATIS

ORACLE Platinum Partner

- Ziel: Vertraulichkeit und Authentizität der Daten
- Apache CXF bietet zwei Ansätze
  - Verwendung von HTTPS als Transportprotokoll
  - Verwendung von WS-Security
- Auch Zertifikate zur Client-Authentifizierung möglich

# Fazit

- Einfache Rich-Client-Entwicklung dank
  - Komfortabler Entwicklungsumgebung
  - Große Menge an wiederverwendaren Plugins (GMF,GEF,...)
- Mächtiger Datenbankzugriff mit EclipseLink:
  - JPA2-Referenzimplementierung
  - Automatisches O/R-Mapping
- Kommunikation mittels Webservices mit Apache CXF
- Entwicklungsbeschleunigung und vielfältige Wiederverwendung dank Single-Sourcing
- Eclipse-Universum bietet jetzt auch serverseitige Komponenten

# Fragen?!?

PROMATIS

ORACLE Platinum Partner

**Johannes Michler**
Dipl.-Inform.
*Software Engineer & Consultant*

**PROMATIS software GmbH**
**Pforzheimer Str. 160**
**76275 Ettlingen**

**Tel. +49 7243 2179 0**
**Fax +49 7243 2179 99**

**E-Mail:** johannes.michler@promatis.de
**Web:** www.promatis.de
www.horus.biz