UNIVERSITÄT LEIPZIG

Mainframe Internet Integration

Prof. Dr. Martin Bogdan
Prof. Dr.-Ing. Wilhelm G. Spruth

SS2013

WebSphere Application Server Teil 2

Schnittstellen

Container

Containers sind Software Constructe innerhalb eines Web Application Servers. Sie dienen als eine Ausführungsumgebung. In einem Web Application Server existieren 2 Arten von Containern:

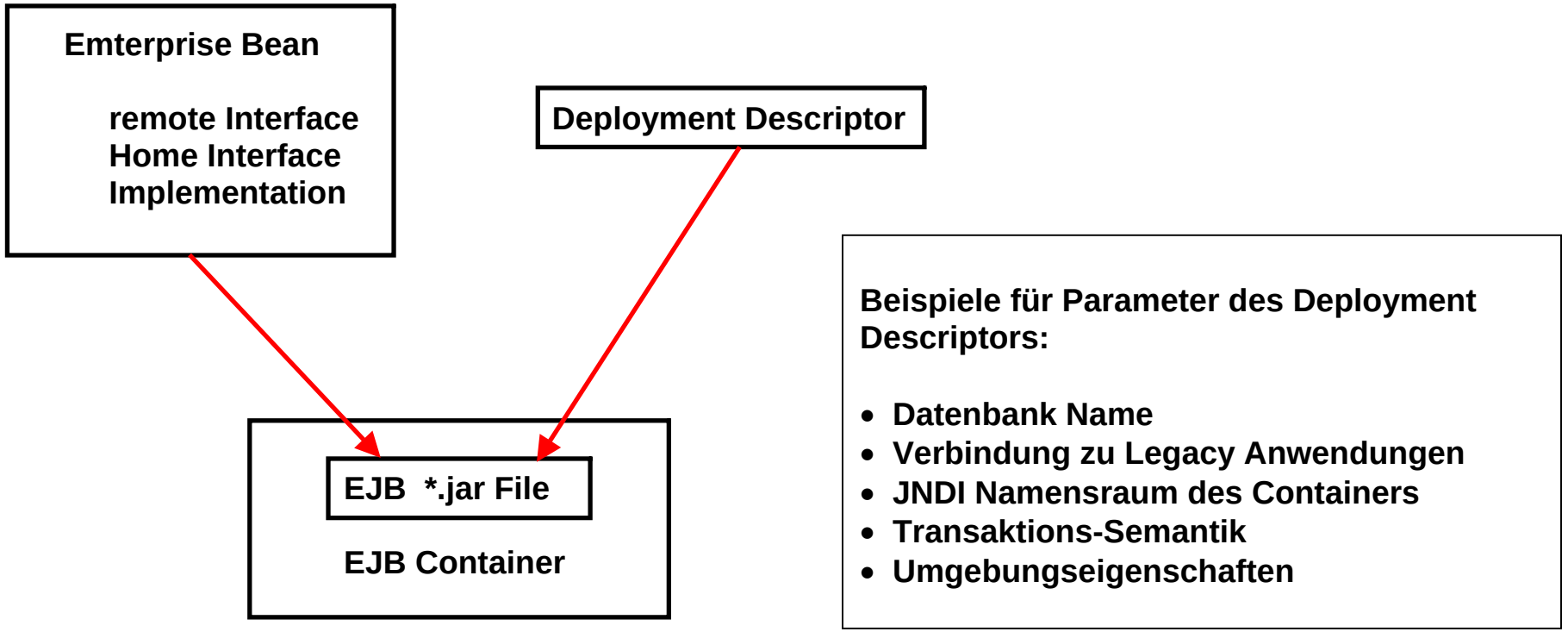
- **EJB Cntainer, in denen EJBs laufen**
- **Servlet Container, in denen Servlets und EJBs ablaufen.**

Angenommen das Erstellen einer neuen Anwendung für einen Web Application Server:

Beide Container Arten stellen Services für die Software Elemente zur Verfügung, die in ihnen laufen. Beispiele für solche Service Funktionen sind z.B. Sicherheit und Namensdienste. Der Entwickler müsste derartige Funktionen für seine Anwendung selbst entwickeln, wenn der Container sie nicht zur Verfügung stellen würde.

Servlet Container werden häufig auch als Web Container bezeichnet. Sie enthalten statische Web Seiten (zusätzlich zu Servlets und JSPs), die spezifisch für diese Anwendung entwickelt wurden und damit Bestandteil der Anwendung sind.

EJBs, die eventuell teilweise bereits existieren, müssen für eine neue Anwendung und Installation in einem EJB Container konfiguriert werden. Dies geschieht mit Hilfe eines „Deployment Descriptors“.



Deployment Descriptor

Der Deployment Descriptor legt die Laufzeit Parameter einer EJB fest.

Der Deployment Descriptor wird in XML codiert und beim Aufruf der EJB ausgewertet. Parameter können statisch zur Assembly Zeit oder dynamisch zur Laufzeit festgelegt werden.

Der Deployment Descriptor wird typischerweise durch den EJB Entwickler angelegt, kann aber durch einen Administrator mit Hilfe eines Tools abgeändert werden.

Deployment Deskriptor

Der Deployment Deskriptor ist eine Datei im XML – Format, die eine oder mehrere EJBs, deren Zusammenwirken und die Art, wie der EJB – Container sie zur Laufzeit behandeln soll, beschreibt. Er enthält hauptsächlich deklarative Informationen, welche nicht im Bean – Code zu finden sind. Dies sind vor allem Informationen über die Struktur der Bean und ihrer Abhängigkeiten zu anderen Beans oder Ressourcen wie z. B. einer Datenbankverbindung.

Außerdem können im Deployment Deskriptor Umgebungsvariablen gesetzt werden, die von der Bean ausgelesen werden können und somit ihr Verhalten beeinflussen. Dies führt zu einer höheren Flexibilität, da dieselbe Bean in verschiedenen Umgebungen eingesetzt werden kann und nur der Deployment Deskriptor angepasst werden muss.

Bei der Installation einer CICS Anwendung (in einer beliebigen Sprache) benutzen wir das „Define“ Kommando zur Definition der Eigenschaften der neuen Anwendung. Für EJBs hat der Deployment Descriptor eine vergleichbare Funktion. Es besteht allerdings ein wesentlicher Unterschied: Die Parameter des Define Kommandos werden bei der Installation der CICS Anwendung fest eingebunden. Der Deployment Descriptor kann beim Aufruf einer EJB ausgewertet und abgeändert werden, was die Flexibilität (auf Kosten der Performance) erhöht.

Ab EJB Version 3.0 können viele Angaben im Deployment Descriptor auch durch Annotationen direkt im Java Code implementiert werden.

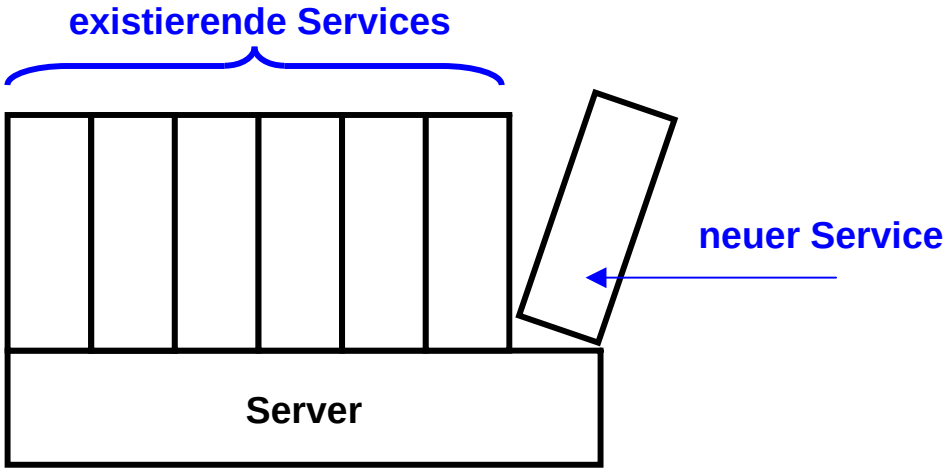
Annotationen

Als Annotation wird im Zusammenhang mit der Programmiersprache Java ein Sprachelement bezeichnet, das die Einbindung von Metadaten in den Quelltext erlaubt. Dieses Element wurde mit der Version Java 5.0 eingeführt.

Annotationen beginnen mit einem @-Zeichen. Daran schließt sich ihr Name an. Optional kann eine kommagetrennte Parameterliste folgen, die in runden Klammern eingefasst wird. Beispielsweise markiert die Annotation im folgenden Quelltextausschnitt die Klasse A als überholt (deprecated):

```
@Deprecated
public class A {}
```

Eingesetzt werden Annotationen unter anderem im Java-EE-Umfeld, um Klassen um Informationen zu erweitern, die vor Java 5.0 in separaten Dateien hinterlegt werden mussten. Prominente Beispiele sind Home- und Local-Interfaces sowie Deployment-Deskriptoren.



Installation einer neuen Anwendung auf einem Server

Wenn wir eine neue Anwendung entwickeln, und auf einem Server zum Laufen bringen wollen, gehen wir durch die folgenden Schritte:

1. **Wir entwickeln die Anwendung in irgend einer Programmiersprache auf irgendeiner Entwicklungsumgebung, z.B. TSO, JDK, Eclipse, andere ...**
2. **Wir testen die neue Anwendung aus (normalerweise auf einem getrennten Testsystem).**
3. **Wir installieren die neue Anwendung für die Benutzung durch zahlreiche Klienten auf einem Produktionssystem (einem Server), z.B. einem CICS Transaktionsserver oder dem EJB Container eines Web Application Servers.**

Für den letzten Schritt gehen wir durch die folgenden Teilschritte:

a. **Wir definieren dem Server gegenüber, dass es jetzt einen neuen Service (unsere neue Anwendung) gibt. Dies geschieht, damit beim Aufruf des neuen Services der Server weis, wo er den entsprechenden Code findet, unter welchen Umständen der Zugriff erfolgt, was der Server beim Aufruf beachten muss, welche Ressourcen er bereitstellen muss, und vieles mehr. Dies erfolgt bei einer CICS Anwendung mittels mehrerer „define“ Kommandos, bei einer EJB Anwendung mittels des „Deployment Descriptors“ .**

b. **Wir installieren den Code der neuen Anwendung auf dem Server**

Vergleich CICS – Enterprise Java Beans

	CICS	EJB
Definition	Define Command	Deployment Descriptor
Installation	Install Command	Deploy Command
Package	Group	JAR, WAR, EAR

Unter CICS wird ein neues Anwendungsprogramm, ein Mapset usw. mit Hilfe des Define Kommandos gegenüber dem CICS Transaktionsserver definiert. Bei einer EJB übernimmt der Deployment Descriptor die gleiche Funktion.

Für die Installation einer neuen Anwendung werden das Business Logik Programm, der Mapset und die TRID zu einer „Group“ zusammengefasst und das „Install“ Kommande ausgeführt. Bei einer JEE Anwendung werden die Kompnenten zu einer Java Archive (jar) File zusammengefasst und mittels des „Deploy“ Kommandos installiert.

Die CICS Group entspricht hierbei grob dem Java Archive.

ejb – jar – Datei

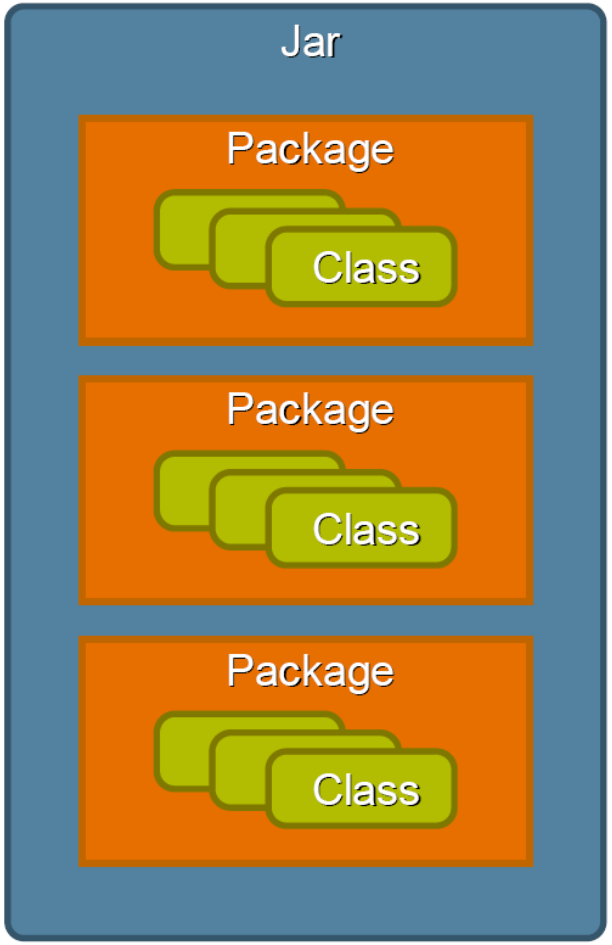
Files mit einer .jar extension (JAR files), sind im Prinzip einfach eine Gruppe von Files

Eine *ejb – jar* – Datei ist das standardmäßige Verpackungsformat für Enterprise JavaBeans. Dabei handelt es sich um eine normale Java – ARchivdatei (JAR), die mit Hilfe des Hilfsprogramms *jar* erzeugt werden kann. Sie enthält aber spezielle Dateien, die alle jene Informationen zur Verfügung stellen, die ein EJB – Container zur Inbetriebnahme der in der JAR – Datei enthaltenen Beans benötigt.

In einer *ejb – jar* – Datei gibt es zwei Arten von Inhalten:

- **Die Klassendateien aller EJBs einschließlich ihrer Interfaces sowie der Bean – Implementationen (der Bean Code). Darüber hinaus können auch containergenerierte Klassen enthalten sein**
- **die Deployment Descriptor – Dateien. Als Minimum muss eine standardmäßige *ejb – jar.xml* – Datei enthalten sein.**

Die kompilierten Java – Klassen, aus denen die EJBs bestehen, befinden sich in packagespezifischen Verzeichnissen innerhalb der *ejb – jar* – Datei, genau wie es bei normalen JAR – Dateien der Fall ist. Der Deployment Deskriptor, mit dem die EJBs beschrieben und konfiguriert werden, muss in der *ejb – jar* – Archivdatei unter *META-INF/ejb-jar.xml* zu finden sein.



Enterprise Applications sind eine Kollektion von JARs. Sie

- **enthalten Packages**
- **enthalten Classes**
- **encapsulate Daten und Logik**

Zur Laufzeit (runtime) ist eine Jar nichts anderes als eine Sammlung von Classes auf einem Classpath

Die Jar Specifikation definiert ein Class-Path Attribute, welches den Classpath extended.

Java Archiv

Web Application

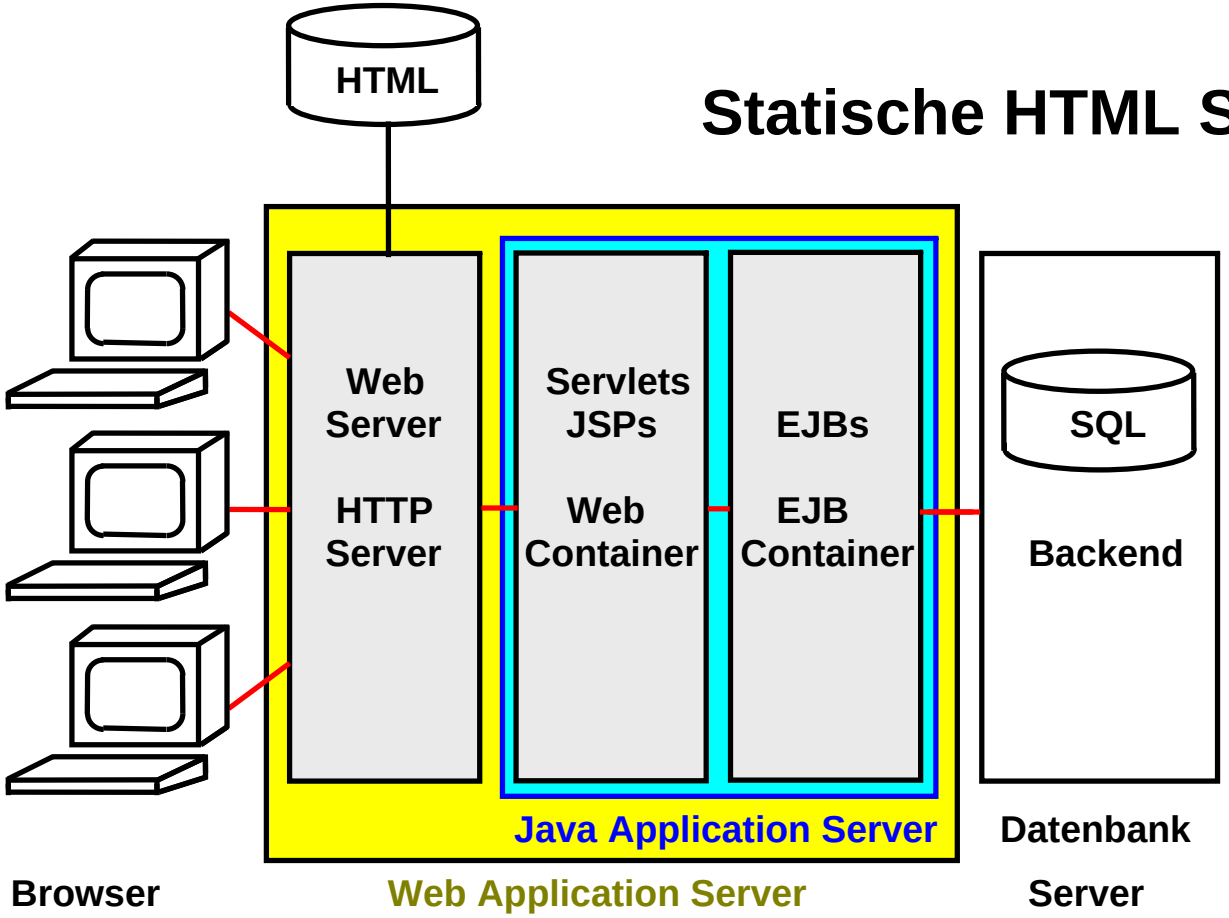
Eine "Web-Applikation" besteht aus einem oder mehreren Servlets, weiteren Java-Klassen, die als Hilfsklassen zur Unterstützung der Servlets dienen, statische Dateien wie HTML-Seiten und GIF/JPG-Bilder, sowie JSPs (Java Server Pages), die eine dynamische Ausgabe formatieren. All diese Elemente sind wichtig für die Ausführung. Zusammen bilden sie eine "Web-Application".

Der gesamte Web-Interaktion Prozess beginnt mit einer einzigen Seite im Browser. Der Benutzer klickt auf eine Schaltfläche oder einen Link auf der Seite. Dies verursacht eine Anfrage (Request) an die Web-Anwendung im Web-Container des Web-Application-Servers, zum Beispiel unter Verwendung von HTML-Forms. Die Anfrage wird von der Web-Anwendung verarbeitet. Je nach den Umständen kann (oder kann nicht) eine EJB oder eine andere Business Logik Komponente aufgerufen werden. Eine neue Seite wird zurück an den Browser gesendet. Diese enthält die Ergebnisse der Anfrage und präsentiert Schaltflächen oder Links für die nächste Anfrage. So besteht die Web-Anwendung aus einem Satz von Verarbeitungsschritten oder Interaktionen. Jede von ihnen erhält eine Anfrage, die von einer Seite erzeugt wird. Eine Antwort wird in Form von einer Seite erstellt, die als Eingabe für eine nachfolgende Interaktion dienen kann.

Es ist wichtig, ein "Web-Applikation" von einer Enterprise Java Bean (EJB) zu unterscheiden. Nach der JEE-Spezifikation sind dies verschiedene Dinge. Sie werden in verschiedenen Arten von Containern ausgeführt, beispielsweise in einem WebSphere für z/OS Anwendungsserver (WAS).

Servlets, die als Teil einer Web-Anwendung ausgeführt werden, können auf EJBs zugreifen. Sie verhalten sich wie ein "Ersatz" Anwendungs-Client für eine EJB. Das Servlet steuert die HTTP-Sitzung mit dem Browser, das Format der Eingabeparameter und das Format der Ausgabe die an den Browser des echten Klienten zurückgegeben wird. Die EJB konzentriert sich auf die Geschäftslogik der Anwendung und greift auf Unternehmensressourcen und Daten zu.

Statische HTML Seiten



Eine neue Anwendung besteht in der Regel aus 2 Teilen, der Business Logik und der Präsentationslogik.

Die Businesslogik wird als eine Menge von EJBs erstellt, und in der Form eines .jar Verzeichnisses für die Installation in dem Web Application Server bereit gestellt.

Wir brauchen etwas vergleichbares für die Präsentation Logik.

In einem Unternehmen sind die meisten statischen Web Seiten nur im Zusammenhang mit einer spezifischen Anwendung relevant

Die Präsentationslogik besteht typischerweise wiederum aus 2 Teilen:

a. einer Reihe von Servlets und Java Server Pages, und
b. einer Reihe von statischen HTML Seiten.

Beispiele solcher statischen HTML Seiten sind z.B. ein Welcome Screen oder ein Verabschiedungsscreen („Thank you for visiting our order status page“).

Es ist sinnvoll, diese statischen HTML Seiten, zusammen mit den Servlets und Java Server Pages der Präsentatinslogik in ein gemeinsames „Web Archive“, einer WAR File, zu verpacken.

Web Archive (WAR)

Eine WAR – Datei (Web ARchiv) dient standardmäßig dazu, Web – Anwendungen zu verpacken. In einer WAR – Datei gibt es drei Arten von Inhalten:

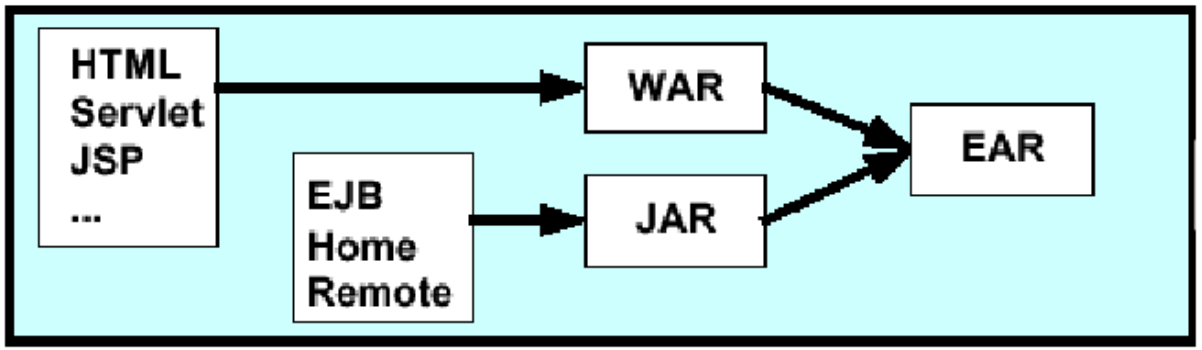
- **Die Klassendateien aller Servlets und JSPs, die Bestandteil der Webanwendung sind. Diese sind unter dem Verzeichnis WEB-INF/classes in einer package – spezifischen Verzeichnisstruktur zu finden.**

- **Die statischen HTML – Seiten, welche für das Ausführen der Webanwendung nötig sind. Es ist aber auch möglich, JSP – Seiten, als dynamische Generierung von Response – Seiten zu verwenden. Diese Dokumente befinden sich innerhalb der WAR – Datei in einer Verzeichnisstruktur. Zum Beispiel könnten alle Bilder im Verzeichnis *images* abgelegt sein, während die *index.html* im context – root des Archivs liegt.**

- **Die Deployment Deskriptor – Dateien. Der Deployment Deskriptor eines Web – Archives ist (wie auch schon der ejb – jar – Deskriptor), eine XML – Datei, welche eine Beschreibung der Anwendung, die Namen und die Parameter der Servlets zur Laufzeit, sowie sessionrelevante Einstellungen enthält. Als Minimum muss eine standardmäßige web.xml – Datei enthalten sein, welche im Verzeichnis WEB-INF zu finden ist. Wie bei den Java Archiven ist es auch hier möglich, das noch weitere Deskriptoren, je nach Art des Produktes und des Herstellers, in dem Web – Archiv Verwendung finden.**

Außerdem werden auch Sicherheitsaspekte, wie zum Beispiel der Loginmechanismus einer Anwendung definiert.

Der Servlet Container eines Web Archives wird analog häufig als Web Container bezeichnet. Um eine Verwechselung mit dem Web Server (z.b. Apache) zu vermeiden, bezeichnet man letzteren dann als http Server. Um Missverständnisse zu vermeiden, empfehlen wir dringend, den Begriff Web Server zu vermeiden, und statt dessen die Begriffe http Server und Web Container (oder Servlet Container) zu benutzen.



Zusammenspiel von WAR, JAR und EAR

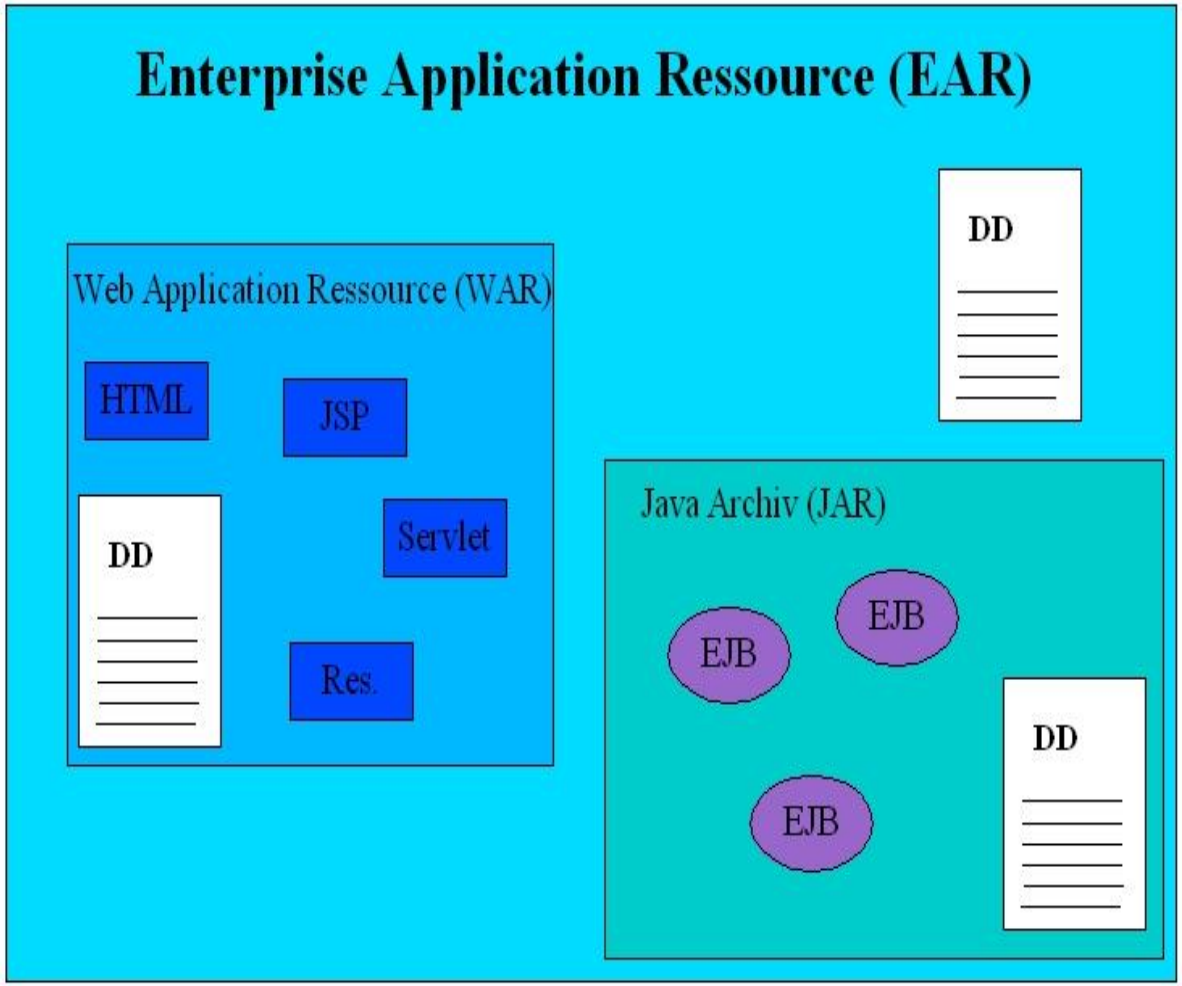
Eine J2EE Anwendung besteht aus Präsentationslogik und aus Business Logik, aus Web – und EJB – Komponenten. Beide Arten von Komponenten sind in Archiven verpackt, Verhalten und Eigenschaften sind in Deployment Deskriptoren (DD) definiert.

Alle Komponenten (Business Logik und Präsentationslogik) einer Anwendung sind in einem EAR (Enterprise Application Ressource) zusammengestellt. Eine vollständige Anwendung kann durch Installation eines EAR – Files auf einen anderen Web Application Server verteilt werden und sollte sich dort wie erforderlich konfigurieren lassen.

Hierbei wird die heute übliche Konfiguration unterstellt, bei der der http Server eine vom Web Application Server getrennte Einheit ist.

Die Web Application und die EJB Application werden häufig getrennt und von unterschiedlichen Entwicklern erstellt. Die Web Application wird in ein WAR (Web Archive) gepackt; die EJB Application in ein JAR (Java Archive). Beide werden in ein EAR (Enterprise Archive) kombiniert. Eine neue Anwendung wird in einen JEE Application Server (wie Weblogic oder WebSphere) geladen, indem man mittels dessen “Deployment Tools“ die entsprechende Application EAR spezifiziert.

Anmerkung: Nach dem neuen EJB 3.1 Standard müssen EJBs nicht mehr in einer separaten EJB-JAR Datei deployed werden, sondern können direkt in der WAR Datei mit paketiert werden.



DD

Deployment Descriptor

Die EAR– Datei (application.xml) enthält nur den Namen der Enterprise – Anwendung und deren Bestandteile .

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE application PUBLIC "-//Sun Microsystems, Inc.//DTD J2EE Application 1.3//EN"
  "http://java.sun.com/dtd/application_1_3.dtd">
<application>
  <display-name>Hello World Web App</display-name>
  <module>
     <web>
       <web-uri>[b]hello.war[/b]</web-uri>
       <context-root>hello</context-root>
     </web>
  </module>
</application>
```

Ein sehr primitives Beispiel einer EAR Datei

JEE Komponenten

Die folgende Abbildung enthält einen Überblick über die Komponenten eines JEE Web Application Servers.

Die JEE Anwendung besteht aus 2 Komponenten: Business Logik und Presentation Logic.

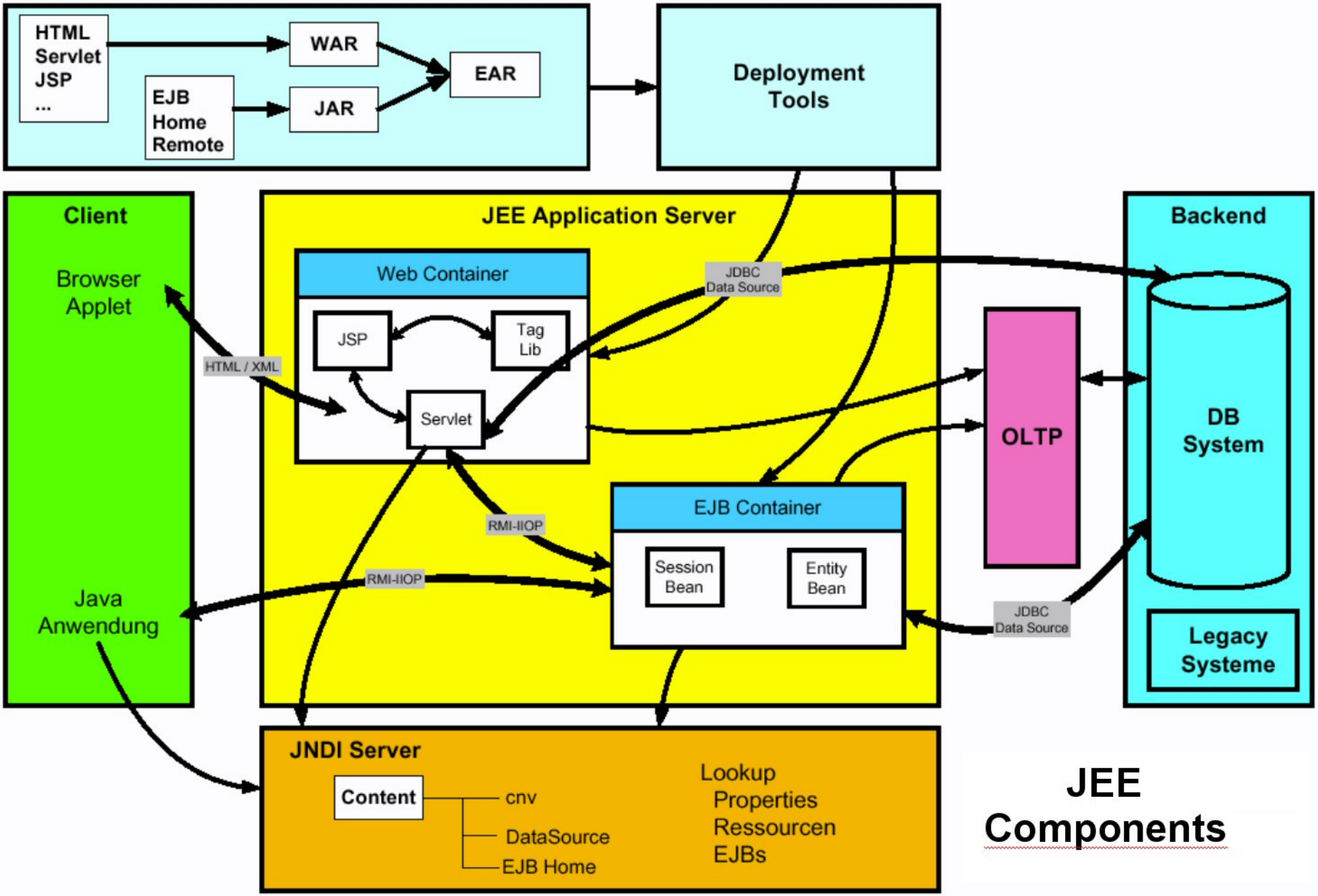
Die Business Logic besteht aus einer EJB Anwendung, die in einem EJB Container Läuft. Die EJBs greifen entweder auf eine Datenbank direkt zu (über JDBC oder SQLJ), oder sie benutzen einen zwischengeschalteten Transaction Monitor (OLTP, On-Line Transaction Processing System), z.B. CICS.

Für die Presentation Logic existieren 2 Alternativen. Ein thin-Client Ansatz benutzt einen Browser, sowie entweder HTTP oder XML für den Aufruf der Server-centric Presentation Logic, die in einem Servlet Container (Web Container) untergebracht ist. In diesem Fall bezeichnet man die Presentation Logic auch als "Web Application". (Nicht gezeigt ist der http Server, der die http Requests von dem Browser entgegennimmt und an den Web Container weiterreicht).

Alternativ wird die Presentation Logic in einem thick Client Ansatz implementiert. Hierbei greift die thick Client Logik auf die EJBs direkt mittels JRMP oder RMI/IIOP zu. z/OS verwendet hier ausschließlich RMI/IIOP.

Wenn der Web Container und der EJB Container als getrennte Prozesse in getrennten JVMs implementiert sind, kommunizieren sie über RMI/IIOP. Wenn sie in der gleichen JVM laufen, ist der RMI/IIOP Overhead nicht erforderlich.

Alle Java Komponenten benutzen JNDI (Java naming and directory service) "to locate each other".


HTML
Servlet
JSP
...
EJB
Home
Remote
WAR
JAR
EAR
Deployment
Tools
Client
Browser
Applet
Java
Anwendung
JEE Application Server
Web Container
JSP
Tag
Lib
Servlet
HTML / XML
JDBC
Data Source
RMI-IIOP
RMI-IIOP
EJB Container
Session
Bean
Entity
Bean
OLTP
JDBC
Data Source
Backend
DB
System
Legacy
Systeme
JNDI Server
Content
cnv
DataSource
EJB Home
Lookup
Properties
Ressourcen
EJBs
JEE
Components

e-business

E-Business-Anwendungen kombinieren Web Application Server, Web-Clients (z. B. Web-Browser) und Standard-Internet-Protokolle, um den Zugriff auf Daten und Anwendungen zwischen mehreren Unternehmen zu erleichtern. Es gibt viele Technologien, die bei der Entwicklung der Client-Seite einer Web-Anwendung in einer E-Business-Anwendung genutzt werden kann. Dazu gehören Java, TCP / IP, HTTP, HTTPS, HTML, DHTML, XML, MIME, SMTP, IIOP und X.509, unter anderem. Der Erfolg des Projekts und künftige Anpassungsfähigkeiten hängen davon ab, welche Technologien einbezogen werden.

IBM ermutigt Entwickler, die Client-Seite von Web-Anwendungen leicht-gewichtig zu machen. Dies wird üblicherweise als Thin-Client bezeichnet. Der Thin Client besteht aus einem Java-fähigen Web-Browser und eine Java Virtual Machine (JVM), eine TCP/IP-Stack, und (möglicherweise) einer Verschlüsselungs (Encryption) Bibliothek. Mit dieser Funktion hat der Client drei wesentliche Vorteile für E-Business-Anwendungen:

- **Jeder Browser hat das Potential, eine Web-Anwendung auszuführen, die auf dem Thin Client-Modell basiert.**
- **Der Client-Teil der Web-Anwendung ist klein und schnell heruntergeladen.**
- **Der Server ist in der Lage, maßgeschneidertes HTML an den Klienten zurück zu geben, indem Client oder Benutzer Attributen ausgewertet werden.**