ARS

Member of

TIMETOACT GROUP
SOFTWARE & CONSULTING

Java Naming and Directory Interface (JNDI)

Java EE Grundlagen

ARS

Inhalte dieses Kapitels

Einführung

Programmierung

JNDI und Java EE

Zusammenfassung

Einführung [1|8]

Was ist ein Namensdienst?

- Assoziation von Namen mit Objekten
  - Kopien oder Referenzen auf externe Ressourcen
  - Suche nach Namen
- Analogien: Telefonbuch, Domain Name System (DNS)

Was ist ein Verzeichnisdienst?

- Namensdienst mit
  - Attributen für Objekte
  - Attributmanipulation
  - Suche nach Attributen
- Beispiel: Lightweight Directory Access Protocol (LDAP)

ARS

Einführung [2|8]

Für was steht JNDI ?

- Java Naming and Directory Interface
- Standardisierte Java-Schnittstelle zum Zugriff auf verschiedene Namens- und Verzeichnisdienste

Motivation für JNDI

- Einheitliche Zugriffe auf die unterschiedlichsten Verzeichnisdienste
- JNDI wird benutzt um ganze Java-Objekte zu lesen und zu schreiben
- JNDI wird benutzt um verschiedene Verzeichnisdienste, wie etwa LDAP und NDS, als ein großes Verzeichnis erscheinen zu lassen

ARS

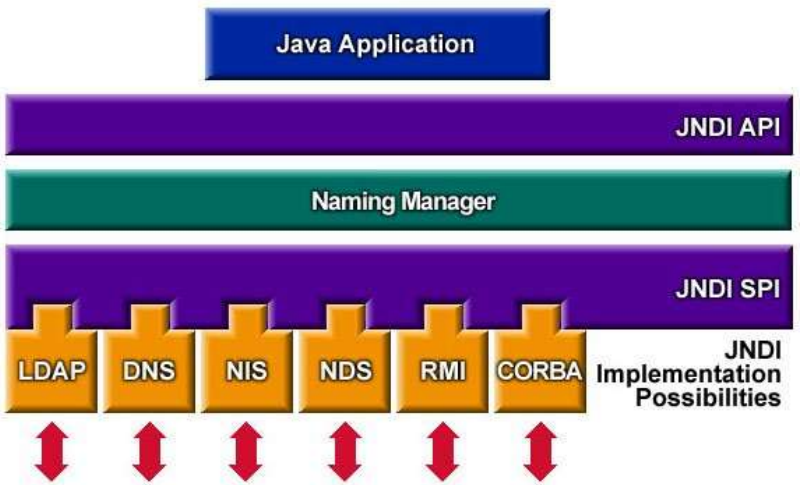
Einführung [3|8]

Verwendung von JNDI in Java EE

- Bereitstellung von Ressourcen
  - Am Server konfigurierte Datenbankverbindungen, URLs, …
  - EJBs
- Abfrage in den Komponenten der Anwendung
  - Nutzen der JNDI API (programmatisch)
  - Resource Injection

JNDI Architektur

- Besteht aus zwei Teilen
  - Client API (JNDI API)
  - Service Provider Interface (JNDI SPI)

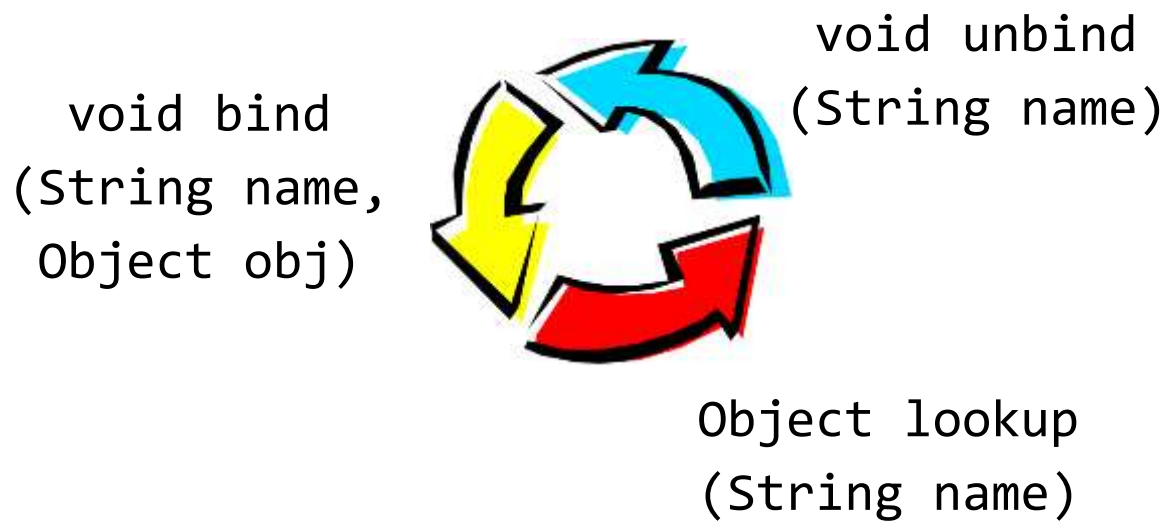


ARS

Einführung [4|8]

Typische Benutzung

- Registrierung eines Objekts beim Namensdienst mittels der Methode `bind()` bzw. `rebind()`
- Ermitteln einer Objektreferenz mittels `lookup()`
- Entfernen der Assoziation mittels `unbind()`

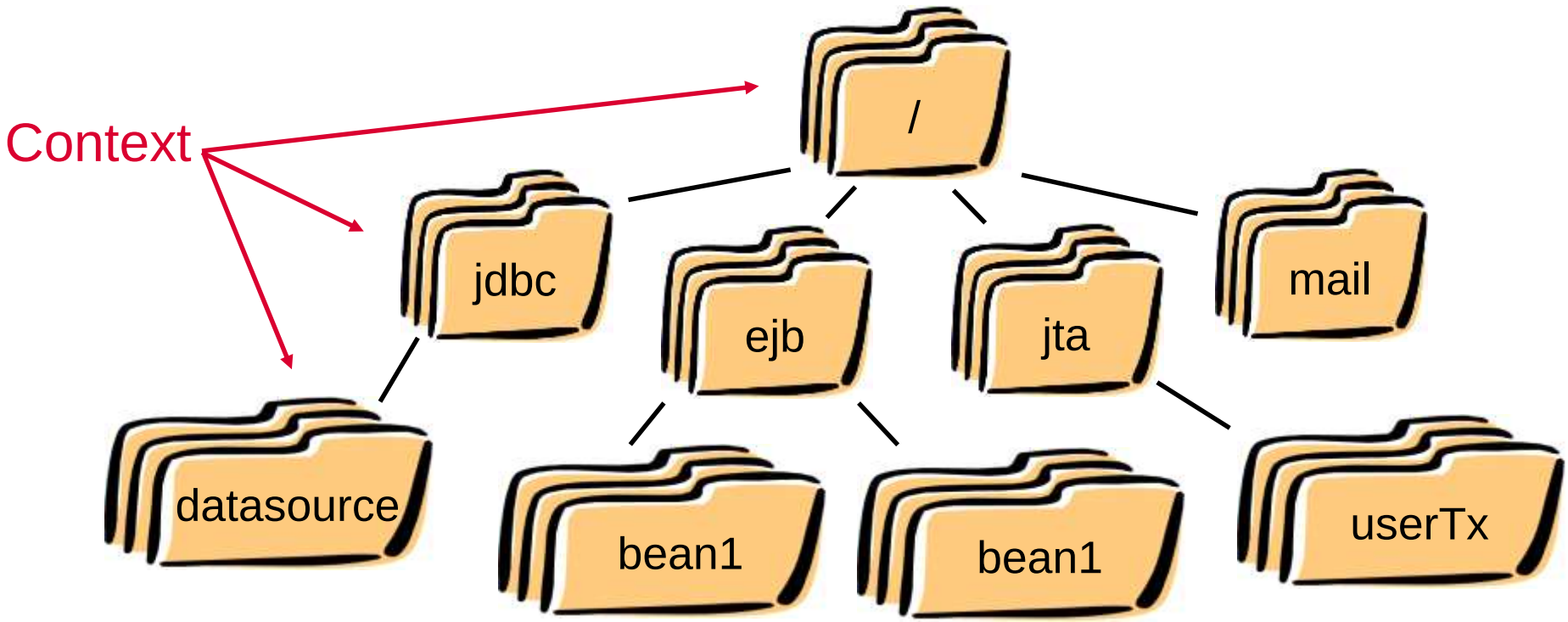


ARS

Einführung [5|8]

Namenssysteme, Namensräume und Kontexte

- Strukturierung in Baumhierarchie
  - Namenssystem ist Teilmenge (Teilbaum) des Datenbestands
  - Namensraum (Namespace) umfasst alle eingetragenen Namen eines Namenssystems



ARS

Einführung [6|8]

Initial Context

- Startpunkt für alle Operationen (z.B. Suche) auf einem Namensraum
- Ermittlung über …

Initial Context Factory

- Implementierung des verwendeten JNDI-Treibers
- Beispiel: `com.ibm.websphere.naming.WsnInitialContextFactory`
- Notwendige Informationen bei der Instanziierung
  - IP Adresse bzw. Name des JNDI-Servers
  - Portnummer des JNDI-Servers
  - Evtl. Benutzername und Passwort zur Authentifizierung
  - Startpunkt innerhalb des JNDI-Baums
- Angaben können im Java-EE-Container weggelassen werden
  - Standardkonstruktor

ARS

Einführung [7|8]

Erzeugung des Initial Contexts (Unmanaged Client)

```
public InitialContext getInitialContext() {
    // Set necessary properties
    Properties properties = new Properties();
    properties.put(Context.PROVIDER_URL, "iiop://localhost:2809/");
    properties.put(Context.INITIAL_CONTEXT_FACTORY,
      "com.ibm.websphere.naming.WsnInitialContextFactory");
    try {
      // Retrieve initial context
      InitialContext initialContext = new InitialContext(properties);
    } catch (NamingException namingException) {
      // exception handling
    }
    return initialContext;
}
```

ARS

Einführung [8|8]

Erzeugung des Initial Contexts (Managed Client)

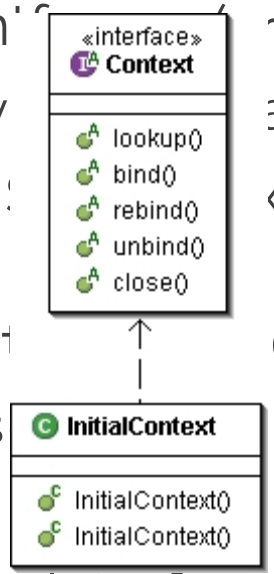
```
public InitialContext getInitialContext() {
  try {
    // Retrieve initial context
    InitialContext initialContext = new InitialContext();
  } catch (NamingException e) {
    // exception handling
  }
  return initialContext;
}
```

ARS

Programmierung [1|8]

Package `javax.naming`

- Klassen und Interfaces zum Zugriff auf einen Namensdienst
  - Interface `Context`
    - Registrieren (`bind()`) und En[illegible] (`bind()`)
    - Suchen und Umbenennen v[illegible]assoziation (`lookup()`, `rename()`)
    - Erzeugen und Löschen von [illegible]kten (`createSubcontext()`, `destroySubcontext()`)
    - Auflisten von Einträgen (`lis[illegible]()`)
    - Schließen des Kontext (`clos[illegible]`)
  - Klasse `InitialContext`
    - Implementierungsklasse des Interface Context
    - Spezifiziert eine Wurzel im Datenmodell
  - Klasse `NamingException`



ARS

Programmierung [2|8]

Verwendung eines Namensdienstes

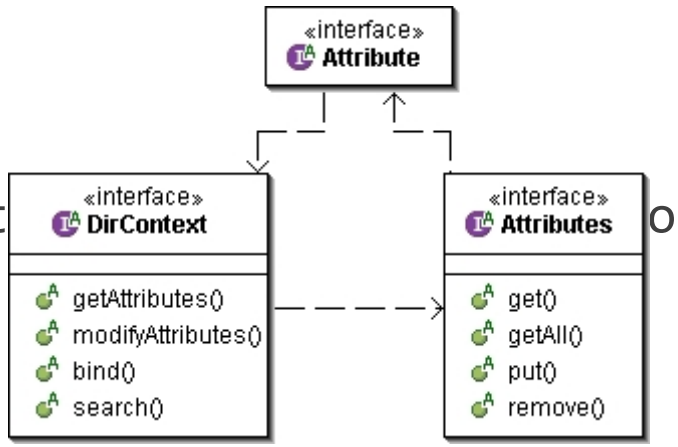
```
try {
  // Perform the bind
  initialContext.bind("favorite", new Fruit("orange"));
  initialContext.rebind("favorite", new Fruit("lemon"));

  // Rename the object
  initialContext.rename("favorite", "myFavorite");

  // Perform lookup on the object
  Fruit favorite = (Fruit)initialContext.lookup("myFavorite");

  // Remove the binding
  initialContext.unbind("myFavorite");
  initialContext.close();
} catch (NamingException e) {
  // exception handling
}
```

ARS

Programmierung [3|8]

Das Package `javax.naming.directory`

- Erweitert das Package `javax.naming` um Klassen und Interfaces zum Zugriff auf einen Verzeichnisdienst
- Zugriff und Suche nach Objektattributen
- Interface `DirContext`
  - Registrieren (`bind()`)
  - Abfragen (`getAttribut`
  - Suche nach Attributen
- Interface `Attributes`
  - Repräsentiert eine Sammlung von Attributen
  - Abfrage von Attributen (`get(), getAll()`)
  - Setzten von Attributen (`put()`)
  - Löschen von Attributen (`remove()`)



`odifyAttributes())`

Programmierung [4|8]

Der Verzeichnisdienst LDAP / X.500

- Das X.500 Verzeichnis stellt eine Art Objektdatenbank dar
- Zu speichernde Informationen werden in Objektklassen beschrieben
  - Attributnamen, Typen und deren Wertebereich
- Unterschiede eines Verzeichnisdienstes zu einer Datenbank

Verzeichnisdienst	Datenbank
Objekte unabhängig voneinander in Hierarchie	Objekte mit Relationen in beliebigen Strukturen
Typischerweise stark verteilt	Typischerweise zentralisiert
Festes Schema für Basisobjekte	Freies Schema

- Festes Attributschema bei X.500

CN = commonName	LO = localityName
ST = stateOrProvinceName	O = organizationName
OU = organizationalUnitName	C = countryName
UID = userid	

- LDAP dient als Zugriffsprotokoll auf X.500-Verzeichnisse

Programmierung [5|8]

Verwendung eines Verzeichnisdienstes

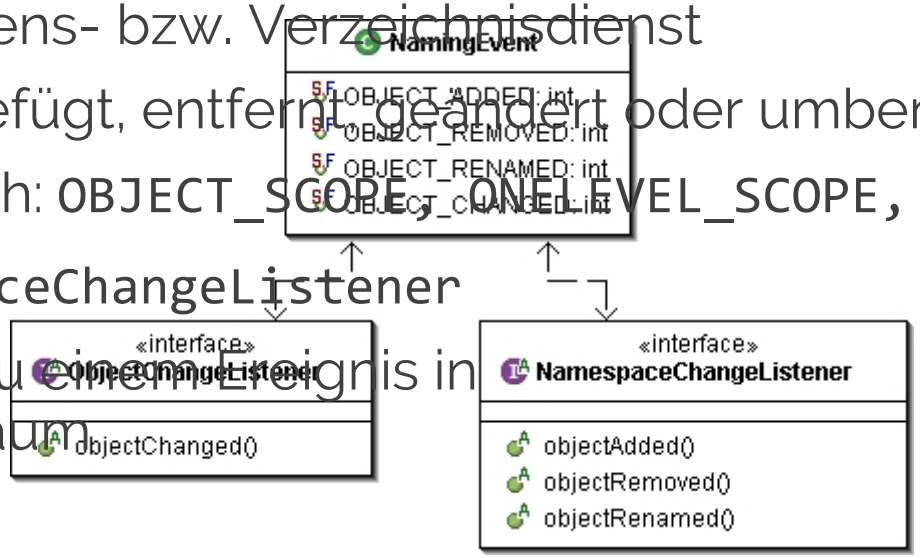
```
try {
  Properties env = new Properties();
  env.put(Context.INITIAL_CONTEXT_FACTORY,
    "com.sun.jndi.ldap.LdapCtxFactory");
  env.put(Context.PROVIDER_URL, "ldap://localhost:389/o=JNDIExample");
  DirContext directoryContext = new InitialDirContext(env);

  // Ask for all attributes of the object
  Attributes attrs = directoryContext.getAttributes(
      "cn=Walter Schilder,ou=People");

  // Find the surname ("sn") and print it
  System.out.println("sn: " + attrs.get("sn").get());
} catch (NamingException e) {
  // exception handling
}
```

ARS

Programmierung [6|8]

Package `javax.naming.event`

- Erweitert JNDI um Klassen und Interfaces um Benachrichtigungen
- Klasse `NamingEvent`
  - Ereignis im Namens- bzw. Verzeichnisdienst
    - Objekt hinzugefügt, entfernt, geändert oder umbenannt
  - Gültigkeitsbereich: `OBJECT_SCOPE, ONELEVEL_SCOPE, SUBTREE_SCOPE`
- Interface `NamespaceChangeListener`
  - Korrespondiert zu einem Ereignis in einem Namensraum
  - `objectAdded()`
  - `objectRemoved()`
  - `objectRenamed()`
- Interface `ObjectChangeListener`
  - `objectChanged()`



ARS

Programmierung [7|8]

Erstellung eines Event Listeners

```
public class NCListener implements NamespaceChangeListener {
  private String id;
  public NCListener(String id) {
    this.id = id;
  }
  public void objectAdded(NamingEvent evt) {
    System.out.println(id + ">>> added: " +
      evt.getNewBinding());
  }
  public void namingExceptionThrown(NamingExceptionEvent evt) {
    System.err.println(id + ">>> NCL got an exception");
    evt.getException().printStackTrace(System.err);
  }
}
```

ARS

Programmierung [8|8]

Verwendung von Naming Events

```
try {
  // Get event context for registering listener
  EventContext eCxt =
    (EventContext)initialContext.lookup("ou=People");
  // Create listener
  NamingListener listener = new NCListener("nc1");
  // Register listener for namespace change events
  eCxt.addNamingListener(
    "ou=Objects,cn=Walter Schilder",
    EventContext.ONELEVEL_SCOPE,
    listener);
  // Not strictly necessary if context get's closed
  eCxt.removeNamingListener(listener);
} catch (NamingException e) {
  // exception handling
}
```

ARS

JNDI und Java EE [1|9]

Globaler Namensraum (direct lookup)

- *Direct Lookup*
- Einträge sind für alle mit dem Namensdienst verbundenen Applikationen sichtbar

```
ref = initialContext.lookup("ejb/SessionHome");
```

Lokaler Namensraum (non direct lookup)

- *Non-direct Lookup*
- Suchraum für einzelnes Modul bzw. Enterprise-Anwendung

```
java:global[/<app-name>]/<module-name>/<bean-name>!<fully-qualifiedbean interface-name>
java:global[/<app-name>]/<module-name>/<bean-name>
java:app/<module-name>/<bean-name>!<fully-qualified-bean-interface-name>
java:app/<module-name>/<bean-name>
java:module/<bean-name>!<fully-qualified-bean-interface-name>
java:module/<bean-name>
```

ARS

JNDI und Java EE [2|9]

Beispiel: Name der Anwendung bzw. des Moduls

- Abfrage über JNDI möglich (s.u.)
- Initialwert in EAR/Module Deployment Descriptor
  - Festlegung am Application Server

Resource Injection

- Kein programmatischer Zugriff auf JNDI
- Dependency Injection → Unabhängigkeit von JNDI

```
@Resource(lookup="java:module/ModuleName")
private String moduleName;

@Resource(lookup="java:app/AppName")
private String appName;
```

ARS

JNDI und Java EE [3|9]

Namespace Binding

- Kein Direktzugriff auf JNDI-Ressourcen über deren Bezeichnung
  - Synchronisation aller Laufzeitumgebungen einer Anwendung notwendig
  - Namenskonflikte zwischen Anwendungen
  - Umkonfiguration einer Anwendung erfordert Umkonfiguration der Ressource
- **Besser:**
  - Ressourcenkonfiguration am Server mit beliebigem JNDI-Namen
  - Anwendung definiert benötigte Ressourcen (Art, JNDI-Name)
  - Bei Installation der Anwendung
    - Zuweisung der Server-Ressourcen zu den anwendungsspezifischen Ressourcen

ARS

JNDI und Java EE [4|9]

Namespace Binding

- Mapping vom lokalen auf globalen Namensraum
  - Festlegung beim Deployment der Applikation
  - Speicherung in Konfiguration des Applikation Servers, z.B.

```
<resRefBindings jndiName="jdbc/DS">
  <bindingResourceRef href="WEB-INF/web.xml#ResourceRef_1"/>
</resRefBindings>
```

- Lokale JNDI Namen

  `java:comp/env/<ResourceType>/<ResourceName>`

  - `<ResourceType>: jdbc, jms, url, mail, ejb oder jca`

ARS

JNDI und Java EE [5|9]

Namespace Binding

- Einträge im lok. Namensraum über den Deployment Deskriptor
  - Eintrag als Umgebungsvariablen
    - Tag `<env-entry>`
  - Referenz auf eine externe Ressource (typischerweise eine Object Factory)
    - Tag `<resource-ref>`
  - Referenz auf ein externe Ressource mit ungeschütztem Zugriff
    - Tag `<resource-env-ref>`
  - Referenz auf eine EJB Home Interface
    - Tag `<ejb-ref>`
  - Referenz auf ein lokales EJB Home Interface
    - Tag `<ejb-local-ref>`

ARS

JNDI und Java EE [6|9]

Beispiel: Umgebungsvariable

- Deklaration

```
<env-entry>
  <description>Maximum Value</description>
  <env-entry-name>maxVal</env-entry-name>
  <env-entry-type>java.lang.Integer</env-entry-type>
  <env-entry-value>10</env-entry-value>
</env-entry>
```

- Zugriff

```
Integer max = (Integer)
  initialContext.lookup("java:comp/env/maxVal");
```

ARS

JNDI und Java EE [7|9]

Beispiel: Datenbankverbindung

- Deklaration

```
<resource-ref id="ResourceRef_1">
  <res-ref-name>jdbc/ApplicationDB</res-ref-name>
  <res-type>javax.sql.DataSource</res-type>
  <res-auth>Container</res-auth>
  <res-sharing-scope>Shareable</res-sharing-scope>
</resource-ref>
```

- Zugriff

```
DataSource ds = (DataSource)
  initialContext.lookup("java:comp/env/jdbc/ApplicationDB");
```

ARS

JNDI und Java EE [8|9]

Beispiel: JMS-Queue

- Deklaration

```
<resource-env-ref id="ResourceEnvRef_1">
   <resource-env-ref-name>
     jms/StockQueue
   </resource-env-ref-name>
   <resource-env-ref-type>
     javax.jms.Queue
   </resource-env-ref-type>
 </resource-env-ref>
```

- Zugriff

```
Queue queue = (Queue)
  initialContext.lookup("java:comp/env/jms/StockQueue");
```

ARS

JNDI und Java EE [9|9]

Umgang mit Referenzen auf EJBs

- Eintrag `ejb-ref`
  - Name der EJB Referenz: `ejb-ref-name`
  - Typ der EJB: `ejb-ref-type`
  - Vollständiger Name des Home Interface: `home`
  - Vollständiger Name des Remote Interface: `remote`
  - Name der EJB im umgebenden JEE Application Package: `ejb-link`
- Deklaration

```
<ejb-ref id="EjbRef_1">
   <ejb-ref-name>ejb/productsmail</ejb-ref-name>
   <ejb-ref-type>Session</ejb-ref-type>
   <home>de.ars.comm.ProductsMailHome</home>
   <remote>de.ars.comm.ProductsMail</remote>
   <ejb-link>ejb/ProductsMail</ejb-link>
</ejb-ref>
```

ARS

Best Practices

Caching

- Ermittlung des `InitialContext` inperformant
- Objekte aus der JNDI → Dynamik beachten

→ Keine Unterstützung für Caching durch JNDI

→ Empfohlen: Resource Injection (Verwaltung durch Container)

ARS

Zusammenfassung

Namens und Verzeichnisdienst

Programmierung

- Initial Context
- bind, unbind, lookup

Bedeutung in der Java EE

- Globaler/lokaler Namensraum
- Referenzen

Best Practice