HANSER

Errata

Javid Jamae, Peter Johnson

JBoss im Einsatz

Den JBoss Application Server konfigurieren

Übersetzt aus dem Englischen von Dorothea Heymann Reder

ISBN: 978-3-446-41574-4

Weitere Informationen oder Bestellungen unter

http://www.hanser.de/978-3-446-41574-4

sowie im Buchhandel.

JBoss in Action – Errata

In Anhang B werden einige Änderungen aufgeführt, die zwischen den Releases CR2 und GA an JBoss AS 5.0.0 vorgenommen wurden. Diese Änderungen werden in den vorliegenden Errata nicht noch einmal wiederholt. In B.1.3 wird zum Beispiel das Verzeichnis *common/lib* erwähnt, in dem die JAR-Dateien liegen, die früher in *server/xxx/lib* untergebracht waren. Das Buch nimmt an vielen Stellen auf das Verzeichnis *server/xxx/lib* Bezug; diese Stellen werden aber in diesen Errata nicht noch einmal aufgeführt, weil sie ja bereits in B.1.3 genannt wurden. Im Mittelpunkt der nachfolgenden Errata stehen die Änderungen an JBoss AS 5.0.0.GA, die im Buch oder im Anhang B noch unerwähnt geblieben sind.

Änderungen, die für den Release JBoss AS 5.1.0.GA spezifisch sind, sind als solche gekennzeichnet.

Kapitel 1

Seite 7, Tabelle 1.1

Änderung: „Red Hat Developer Studio“ soll heißen „JBoss Developer Studio“

Grund: Umbenennung aus Marketing-Gründen.

Seite 19, Tabelle 1.4

Änderung: Die Beschreibung für *standardjboss.xml* soll heißen: „Konfiguriert die verschiedenen EJB 2.x-Container.“

Grund: Das Buch konzentriert sich auf die EJB3-Konfiguration, aber per Zufall sind in Kapitel 7 einige Verweise auf EJB 2.x eingeflossen.

Seite 20, Das lib-Verzeichnis, letzter Satz

Änderung: Der letzte Satz dieses Abschnitts wird gestrichen.

Grund: Datenbanktreiber haben Probleme, wenn sie in isolierten oder beschränkten (Scoped) Installationen eingesetzt werden, auch wenn es dafür Workarounds gibt. Siehe auch

dieser Artikel. Selbst wenn Sie im Moment noch keine beschränkte Installation verwenden, kann sich das in Zukunft ändern. Daher ist es am einfachsten, dies von vorneherein zu vermeiden.

Kapitel 2

Seite 35, Die Beans-Konfigurationsdatei

Die Datei *profile.xml* liegt jetzt im *server/xxx/conf/bootstrap*-Verzeichnis. Die meisten Bean-Konfigurationsdateien wurden aus dem *server/xxx/conf*-Verzeichnis in das *server/xxx/conf/bootstrap*-Verzeichnis verlagert.

Seite 39, Tabelle 2.1

JBoss AS 5.1.0: Die Datei *jbossjta-properties.xml* wurde in *jbossts-properties.xml* umbenannt.

Seite 39, Absatz unter der Tabelle 2.1

Die Datei *standardjboss.xml* ist spezifisch für EJB 2.x.

Seite 38, letzter Aufzählungspunkt (Diverse MBeans für den Remoting-Service)

Der Remoting-Service wird jetzt mit POJOs gesteuert, die in der Datei *server/xxx/deploy/remoting-jboss-beans.xml* definiert sind.

Seite 39, Tabelle 2.1

Die Dateien *boostrap.xml* und *bootstrap-norepo.xml* werden in B.1.5 behandelt.

Die *jboss-minimal.xml*-Datei existiert nicht mehr.

Jetzt gibt es eine *java.policy*-Datei, in der die Sicherheitsrichtlinie für den Anwendungsserver definiert ist (oder eben nicht).

Die Beschreibung zur Datei *jacorb.properties* soll lauten: „Wird zum Konfigurieren des Java Object Request Broker (JacORB)-Service verwendet.“

Seiten 39 ff., Abschnitt 2.2.1

JBoss AS 5.1.0: Standardmäßig werden im *server.log* nur Meldungen protokolliert, die mindestens auf INFO-Ebene liegen. Dies können Sie ändern, indem Sie den `JAVA_OPTS` in den Run-Skripten die Systemeigenschaft `jboss.server.log.threshold` hinzufügen. Wenn Sie zum Beispiel “`-Djboss.server.log.threshold=DEBUG`” hinzufügen, werden auch Meldungen der `DEBUG`-Ebene in der *server.log*-Datei protokolliert. Oder Sie ändern den Standardwert, indem Sie in der *server/xxx/conf/jboss-service.xml*-Datei das Attribut `DefaultJBossServerLogThreshold`

der MBean `jboss.system:type=Log4jService,service=Logging`
bearbeiten.

Seiten 44-45, Abschnitt 2.2.3

In der Datei *server/xxx/log/boot.log* können Sie sehen, welche Systemeigenschaften eingestellt wurden.

Kapitel 3

Seite 59, Tabelle 3.2

Für das Suffix *.zip* soll der URL in der Spalte Anwendungstyp lauten:
http://localhost:8080/someapp.zip

Seiten 61-64, Abschnitt 3.1.5

Änderungen an diesem Abschnitt und Tabelle 3.3 werden in B.1.5 behandelt.

Seite 65, Tipp

Wegen diverser Änderungen an Klassenladern sind die Eigenschaften `useJBossWebLoader` und `java2ClassLoadingCompliance` nicht mehr im Gebrauch. Um anderen Anwendungen Zugriff auf Klassen in einer WAR-Datei zu geben, müssen Sie die Bean `WarClassLoaderDeployer` in der *server/xxx/deployers/jbossweb.deploy/META-INF/war-deployers-jboss-beans.xml*-Datei entfernen oder auskommentieren. Optional können Sie eine *WEB-INF/jboss-classloader.xml*-Datei bereitstellen und genau konfigurieren, welche Anwendungen für die Klassen in der WAR-Datei sichtbar sind.

Siehe auch *http://www.jboss.org/community/wiki/useJBossWebClassLoaderinJBoss5*.

Seite 74, Tabelle 3.5

Der zweite Satz der Beschreibung von Tag `<min-pool-size>` sollte lauten: „Beachten Sie, dass sich der Anwendungsserver nur, wenn die Option `<prefill>` angegeben wurde, …“

Dann muss noch ein neuer Tabelleneintrag für das Tag `<prefill>` mit folgender Beschreibung hinzugefügt werden: „Wenn dies auf `true` gesetzt wird, richtet der Anwendungsserver die Anzahl der in `<min-pool-size>` angegebenen Verbindungen schon zum Zeitpunkt der Bereitstellung der Datenquelle ein. Ist es `false` (der Standardwert), dann richtet der Anwendungsserver erst bei der ersten Anwendungsanforderung Verbindungen ein.“

Seite 76, Tabelle 3.6

In der Beschreibung der MBean `jboss.jca:name=XXX, service=LocalTxCM` wird folgender Satz gestrichen: „Mit dieser MBean verwalten Sie diverse Aspekte von verteilten Transaktionen, z.B. den Timeout-Wert für Transaktionen auf der lokalen XA-Ressource.“

Seite 76, Abschnitt „Eine Datenquelle in eine EAR-Datei verpacken“

Im dritten Absatz wird gesagt, dass Sie die JAR-Datei für den JDBC-Treiber im EAR-Archiv unterbringen können. Beachten Sie, dass dies nicht empfohlen wird. Wenn Sie Klassen-Scoping verwenden, indem Sie ein Klassenlader-Repository angeben, wird die Neubereitstellung des EAR ein Speicherleck im Permgen auslösen, weil die JDBC-Treiberklassen nicht freigegeben werden.

Seite 80, Tabelle 3.6 soll heißen Tabelle 3.7

Kapitel 4

Seite 94, Abschnitt 4.1.6

Änderung: Beide `class="org.jboss.logging.XLevel"`-Attribute werden gestrichen.

Grund: Mit der Aktualisierung von Log4J auf die Version 1.2.14 wurde dieses Attribut überflüssig. Wenn Sie das Attribut angeben, wird `TRACE`-Logging tatsächlich nicht angezeigt.

Kapitel 5

Seite 129, Tabelle 5.1

Änderung: Die Beschreibung für `replication-config` soll beginnen mit: „Gibt an, wann **und wie** ...“

Abschnitt 5.2.1 „Wichtige Verzeichnisse finden“: Seite 132, zweiter Absatz, vorletzter Satz

Änderung: Das Deployer-Verzeichnis soll heißen: *server/xxx/deployers/jbossweb.deployer*.

Abschnitt 5.2.1 „Wichtige Verzeichnisse finden“: Seite 132, erster Aufzählungspunkt

Änderung: Das Verzeichnis soll heißen: *deploy/jbossweb.sar/server.xml*.

Seite 132, vierter Aufzählungspunkt

Änderung: Das Verzeichnis soll heißen. *deployers/jbossweb.deployer/web.xml*.

Kapitel 6

Abschnitt 6.1.1 „Sicherheit konfigurieren in *web.xml*“: Seite 156, erster Absatz, zweiter Satz

Änderung: `web-resource-collection` soll heißen `auth-constraint`

Kapitel 7

Seite 187, Listing 7.1

Die Archivdateinamen sind nicht richtig, das Listing sollte folgendermaßen lauten:

```
<application>
  <display-name>Some Enterprise Archive</display-name>
  <module>
    <web>
      <web-uri>SomeWebArchive.war</web-uri>
      <context-root>/myapp</context-root>
    </web>
  </module>
  <module>
    <ejb>SomeEjbJarArchive.jar</ejb>
  </module>
</application>
```

Seite 193, Abschnitt 7.3, zweiter Absatz, erster Satz

Änderung: „...: mit der Konfiguration der einzelnen *EJB*-Anwendungen und mit der Konfiguration von ...“

Seite 193, Abschnitt 7.3, Ergänzung nach dem zweiten Absatz

Änderung: Folgender Absatz wird eingefügt: „Ein Großteil der JBoss AS-spezifischen EJB-Konfiguration wird den EJBs durch Annotations hinzugefügt. Die Standard-Annotations werden mittels Aspect Oriented Programming (AOP) auf EJBs angewendet. Sie können diese Standard-Annotations in den einzelnen Beans überschreiben oder die Standardwerte durch Konfiguration der Aspekte ändern.“

Abschnitt 7.3.1 „Was gehört wohin?“: Seite 194, erster Absatz

Änderung: Der Absatz, der mit „Das *conf*-Verzeichnis enthält …“ beginnt, wird entfernt.

Grund: Dieser Absatz beschreibt die *standardjboss.xml*-Datei, die zum Konfigurieren von EJB2-Containern und nicht von EJB3-Containern verwendet wird.

Abschnitt 7.3.1 „Was gehört wohin?“: Seite 194, dritter Absatz

Änderung: Der letzte Satz wird ersetzt durch: „Die Datei *deploy/ejb3-interceptors-aop.xml* ist die Hauptdatei, die Sie ändern müssen, um globale Standardeinstellungen für EJBs anzugeben, wie zum Beispiel die EJB-Poolgrößen, Pool-Timeouts und Cache-Konfigurationen.“

Seite 198, Tabelle 7.1, Beschreibung von *deploy/ejb3-interceptors-aop.xml*

Änderung: Am Ende der Beschreibung wird angefügt: „..., und Standardgrößen für EJB-Pools.“

Seite 199, erster Satz nach Listing 7.10

Änderung: Dieser Satz wird durch folgende zwei Sätze ersetzt: „... wird sie unter dem Namen *MessagePrinterBean/remote* oder *MessagePrinterBean/local* gebunden, je nachdem, ob sie eine Remote-Schnittstelle oder eine lokale Schnittstelle erweitert. Clients können dann die Bean unter diesem Namen nachschlagen.“

Grund: Die JNDI-Namensbestandteile */remote* und */local* beschreiben wir auf der nächsten Seite, aber dieses Beispiel war ohne Erwähnung dieser Endungen technisch nicht richtig.

Seite 201-204, Abschnitt 7.4.3

Grund: Der gesamte Abschnitt beschreibt fälschlich, wie EJB3-Container mit der Datei *standardjboss.xml*-Datei konfiguriert werden. Diese Datei wird aber nur zur Konfiguration von EJB2-Containern und nicht für EJB3-Container eingesetzt. Die Änderungen in diesem Abschnitt sind zu stark gestreut, um sie hier einzeln zu dokumentieren, daher geben wir hier den gesamten aktualisierten Text für 7.4.3 wieder.

Änderung:

7.4.3 EJB-Container konfigurieren

In JBoss AS gibt es für jeden remote zugänglichen EJB-Typ eine Containerdefinition. Es ist nützlich zu wissen, wie Container und die für den Containeraufruf erstellten dynamischen Proxies konfiguriert werden, um z.B. solche Features wie die Poolgröße für Session Beans und SFSB-Passivierung selbst bestimmen zu können. In diesem Abschnitt betrachten wir überblicksartig, wie die verschiedenen EJB-Container konfiguriert werden, damit Sie mit den spezifischeren Konfigurationen im Rest dieses Kapitels keine Probleme haben.

Hinweis

Sie können in EJB3 auf einen Entity Manager (die Schnittstelle zum Lesen und Schreiben von Entity-Objekten in einer Datenbank) nicht remote zugreifen, und Sie können ihn auch

nicht in einem Server-Container verwalten. Der Entity Manager greift aus einem persistenten Kontext auf Entity-Objekte zu, also aus einem speicherresidenten Cache von Entity-Objekten, die mit der Datenbank synchronisiert sind. Ein Persistenzkontext wird in der *META-INF/persistence.xml*-Datei einer Anwendung konfiguriert. Da Entities nicht von einem Server-Container verwaltet werden, sehen Sie im Zusammenhang mit der Containerkonfiguration nichts, das sich auf EJB3-Entities bezieht.

Die Standardwerte für den dynamischen Proxy und die Container-Definition für jeden EJB-Typ sind beide in der *server/xxx/deploy/ejb3-interceptors-aop.xml*-Datei definiert. Diese ist eine JBoss AOP-Konfigurationsdatei, die automatisch durch Pointcuts Verhalten auf Beans anwendet. Viele der konfigurierbaren Teile der Beans werden durch Annotations definiert, sodass Sie sie in der einzelnen Bean überschreiben können. Diese Datei hat zwei Hauptblöcke: einen, der dynamische Proxies definiert, und einen, der den Container definiert. Listing 7.12 zeigt die allgemeine Struktur der Datei.

Listing 7.12 Allgemeine Struktur von *ejb3-interceptors-aop.xml*

```
<aop>
 <stack>...</stack>
  <stack>...</stack>
  <domain>...</domain>
  <domain>...</domain>
</aop>
```

Jeder Toplevel-Stack-Block definiert den Interceptor-Stack für einen dynamischen Proxy. Die `domain`-Blöcke sind Aspektdomänen, die (in dieser Datei) verwendet werden, um den Server-Container für jeden EJB-Typ zu konfigurieren. Listing 7.13 zeigt als Beispiel die SFSB-Konfiguration.

Listing 7.13 Die Containerkonfiguration für SFSB in *ejb3-interceptors-aop.xml*

```
<domain name="Base Stateful Bean"
     extends="Intercepted Bean" inheritBindings="true">                ❶
  ...
  <annotation expr="!class(@org.jboss.ejb3.annotation.Pool)">          ❷
    @org.jboss.ejb3.annotation.Pool (value="ThreadlocalPool",
                                     maxSize=30, timeout=10000)        ❷
  </annotation>                                                        ❷
</domain>

<domain name="Stateful Bean"
     extends="Base Stateful Bean" inheritBindings="true">              ❸
  <annotation expr="!class(@org.jboss.ejb3.annotation.Cache) ">        ❹
    @org.jboss.ejb3.annotation.Cache ("SimpleStatefulCache")           ❹
  </annotation>                                                        ❹
  <annotation expr=
    "!class(@org.jboss.ejb3.annotation.PersistenceManager) ">          ❺
    @org.jboss.ejb3.annotation.PersistenceManager
              ("StatefulSessionFilePersistenceManager")                ❺
  </annotation>                                                        ❺
  <annotation expr="!class(@org.jboss.ejb3.annotation.CacheConfig) "> ❻
    @org.jboss.ejb3.annotation.CacheConfig (maxSize=100000,            ❻
                  idleTimeoutSeconds=300,   removalTimeoutSeconds=0)   ❻
  </annotation>                                                        ❻
  ...
</domain>
```

Dass der zweite `domain`-Block die Konfiguration für den SFSB-Container ist, erkennt man an dem Attributnamen ❸. Der zweite `domain`-Block erbt durch das `extends`-Attribut von dem ersten Block (❶). Durch Erweiterung des `Base Stateful Bean`-Blocks erbt der `Stateful Bean`-Block alle in diesem deklarierten Annotations und Bindungen.

Die Containerkonfiguration setzt sich vor allem aus den Interceptors zusammen, die in dem `Base Stateful Bean`-Block definiert sind, der in diesem Listing nicht wiedergegeben wird. Der Container lässt eingehende Anforderungen eine Kette von Interceptors durchlaufen, bevor die EJB erreicht wird. Jeder Interceptor kümmert sich um andere nichtgeschäftliche Aspekte der Anforderung, z.B. Sicherheitsprüfungen und die Initialisierung der Transaktionsverwaltung. Sie können zudem eigene Interceptors anlegen und die Reihenfolge der vorhandenen ändern; eine Möglichkeit, von der in der Praxis allerdings wenig Gebrauch gemacht wird. Wenn Sie die Interceptors ändern oder erweitern müssen, erfahren Sie im JBoss AS-Konfigurationsguide (siehe Quellen am Ende dieses Kapitels), wie das funktioniert.

Manche der anderen Elemente in diesen Konfigurationsdateien werden weitaus häufiger geändert. Schauen wir uns einige dieser Verwendungsmöglichkeiten an.

Die Größen der Bean-Pools konfigurieren

Sie können die Standard-Poolgrößen für jede Container-verwaltete Bean (zum Beispiel SFSB und SLSB) konfigurieren, indem Sie die Werte der Attribute der Annotation `org.jboss.ejb3.annotation.Pool` einstellen ❷. Das `annotation`-XML-Element gibt an, welcher Pointcut erreicht werden muss, bevor die Standard-Annotation angewendet werden darf. In den meisten Fällen ist der Pointcut so konfiguriert, dass die Annotation angewendet wird, wenn sie auf der Klasse noch nicht vorhanden ist. Das wird durch das Ausrufezeichen vor dem Schlüsselwort `class` im `expr`-Attribut veranlasst.

Die Annotation, die angewendet wird, steht im XML-`annotation`-Element. Um die Standardwerte für den Pool zu konfigurieren, ändern Sie die Attribute `maxSize` und `timeout` attributes. Die `maxSize` ist keine strenge Obergrenze, sondern teilt dem Pool nur mit, wie viele Instanzen er maximal am Leben halten soll. Wenn mehr Anforderungen eingehen, als in diesem Maximum vorgesehen, erzeugt der Server mehr Instanzen. Wenn Sie die Anzahl der gleichzeitigen Server-Requests streng begrenzen möchten, müssen Sie im Wertattribut `strictMaxPool` und `strictTimeout` wie folgt verwenden:

```
<annotation expr="!class(@org.jboss.ejb3.annotation.Pool)">
  @org.jboss.ejb3.annotation.Pool (value="StrictMaxPool", maxSize=30,
timeout=10000)
</annotation>
```

Das Attribut `strictMaximumSize` lässt den Container niemals mehr nebenläufige Anforderungen tolerieren, als im `maxSize`-Attribut definiert sind (hier 30). Gehen darüber hinaus noch weitere Requests ein, werden sie blockiert, bis sie nach der im `strictTimeout`-Element festgelegten Anzahl Millisekunden ablaufen. Danach wird eine

`java.rmi.ServerException` ausgelöst. Ist der Timeout-Wert 0, findet das Timeout sofort statt. Der Maximalwert (und zugleich der Standardwert) ist der größtmögliche `Long`-Wert, d.h. 9.223.372.036.854.775.807 Millisekunden oder ca. 292.471.208 Jahre.

Sie können das `@Pool`-Attribut auch direkt auf eine Bean anwenden, wenn Sie die Standardwerte überschreiben möchten. Die Klasse `org.jboss.ejb3.annotation.defaults.PoolDefaults` enthält Konstanten, um die Annotation-Argumente anzugeben. Wenn Sie das Wertargument spezifizieren, können Sie zum Beispiel `PoolDefaults.POOL_IMPLEMENTATION_THREADLOCAL` für den lokalen Standard-Threadpool oder `PoolDefaults.POOL_IMPLEMENTATION_STRICTMAX` für eine strenge Größenbeschränkung des Pools verwenden.

Die SFSB-Passivierung konfigurieren

Die Passivierung zustandsbehafteter Beans wird mit drei Annotations konfiguriert: `Cache` (❹), `PersistenceManager` (❺) und `CacheConfig` (❻). Die Annotation `PersistenceManager` definiert, welcher Persistenzmanager für die Passivierung verwendet werden soll. Der Container nutzt den Persistenzmanager, um passivierte Objekte in den Sekundärspeicher zu schreiben oder daraus zu lesen. Der Standardpersistenzmanager von SFSB ist `StatefulSessionFilePersistenceManager`; er schreibt den Session Bean-Zustand in eine Datei.

Die Passivierungsregeln sind in der Annotation `CacheConfig` definiert. In dieser Annotation besagt das Attribut `maxSize`, wie viele Beans maximal im Cache gespeichert werden können, bevor dieser anfängt, die Beans zu passivieren. Die Passivierung geschieht nach einem Least Recently Used (LRU)-Algorithmus. Das Attribut `idleTimeoutSeconds` gibt an, wie viele Sekunden vor der Passivierung einer SFSB-Instanz abgewartet wird, unabhängig davon, ob der Cache bereits seine maximale Größe erreicht hat oder nicht. Das Attribut `removalTimeoutSeconds` gibt an, wie viele Sekunden abgewartet wird, bevor der Cache die Bean komplett entfernt.

Beachten Sie, dass das Attribut `removalTimeoutSeconds` auf null Sekunden voreingestellt ist. Damit wird diese Löschfunktion effektiv ausgeschaltet, sodass die SFSB nach dem Zeitraum von `idleTimeoutSeconds` (standardmäßig 300 Sekunden) automatisch passiviert wird. Wenn Sie das Attribut `removalTimeoutSeconds` einstellen, dann sollten Sie einen Wert größer `idleTimeoutSeconds` wählen, wenn Sie möchten, dass die Passivierung funktioniert. Wenn `removalTimeoutSeconds` auf einen Wert eingestellt wird, der ungleich null, aber kleiner als `idleTimeoutSeconds` ist, wird die SFSB schon vor der Passivierung aus dem Cache entfernt. Das ist ein möglicher Weg, um die SFSB-Passivierung zu deaktivieren.

Die Passivierung kann auch deaktiviert werden, indem die `Cache`-Annotation (❹) von `SimpleStatefulCache` auf `NoPassivationCache` umgestellt wird, wie hier gezeigt.

```
<annotation expr="!class(@org.jboss.ejb3.annotation.Cache) ">
    @org.jboss.ejb3.annotation.Cache ("NoPassivationCache")

/annotation>
```

Wenn Sie für Ihre SFSB-Passivierung eine Datenbank oder einen verteilten Cache verwenden möchten, verwenden Sie am besten die *all*-Konfiguration mit einem CacheLoader. Dies werden wir im Zusammenhang mit Clustering in den Kapiteln 12 und 13 genauer erklären.

Nun können Sie Session Beans und ihre Container konfigurieren, aber noch nicht die Entity-Persistenz.

Seite 206, Abschnitt 7.6, nach dem letzten Satz des zweiten Absatzes

Änderung: Da nur eine einzige Instanz vorhanden ist, muss der Service so programmiert werden, dass er Thread-sicher ist.

Abschnitt 7.8.1 „EJB-Sicherheit durch Anmerkungen“
Seite 215, Tipp (unter Tabelle 7.2)

Änderung: Der Tipp soll lauten: Achten Sie darauf, die richtige Anmerkungsklasse zu verwenden. So gibt es z.B. die Anmerkungsklasse `org.jboss.ejb3.annotation.security.SecurityDomain` und die Anmerkungsklasse `org.jboss.aspects.security.SecurityDomain`. Da beide denselben Namen haben und nach Anzahl und Typ dieselben Argumente nehmen, kann es leicht passieren, dass Sie die falsche Klasse importieren (oder `org.jboss.aspects.security.SecurityDomain`), wenn Sie einer EJB Sicherheit durch Anmerkungenhinzufügen.

Grund: Die Annotation wurde seit der JBoss 5.0 Beta umbenannt, aber der Tipp wurde nicht aktualisiert.

Kapitel 8

Seite 232, erster Absatz, vorletzter Satz

Die Datei *example-destinations-service.xml* mit den im Beispiel verwendeten Destinationen liegt im Verzeichnis *docs/examples/jms*.

Seite 240, Absatz unter Tabelle 8.4

Dieser Absatz gilt nur für EJB 2.x MDBs.

Seite 247, Abschnitt 8.5.1, Absatz vor Listing 8.8

Die Datei **-ds.xml* aus dem Beispiel liegt in Wirklichkeit in *docs/examples/jca/postgres-ds.xml*.

Seite 250, Listing 8.11

Die Listing-Überschrift sollte heißen: *messaging-jboss-beans.xml* (Auszug)

Kapitel 9

Seite 260, zweiter Absatz

Die JAX-RPC-Dokumentation befindet sich jetzt unter *http://jbossws.jboss.org/mediawiki/index.php?title=JAX-RPC_User_Guide.*

Seite 274, Tabelle 9.3, zweiter Punkt

Die `configFile`-Standardeinstellung ist jetzt *server/xxx/deployers/jbossws.deployer/META-INF/standard-jaxws-endpoint-config.xml*.

Alle *standard-*-config.xml*-Dateien wurden vom *server/xxx/deploy/jbossws.sar/META-INF*-Verzeichnis in das *server/xxx/deployers/jbossws.deployer/META-INF*-Verzeichnis verlagert.

Seite 281, letzter Absatz (nach Listing 9.15)

Wie schon gesagt, liegt die *standard-jaxws-endpoint-config.xml*-Datei jetzt unter *server/xxx/deployers/jbossws.deployer/META-INF*.

Seite 284, Absatz vor Listing 9.18

Wie schon gesagt, liegt die *standard-jaxws-endpoint-config.xml*-Datei jetzt unter *server/xxx/deployers/jbossws.deployer/META-INF*.

Kapitel 10

JBoss Portal 2.7

Siehe auch B.1.2 wegen Informationen über JBoss Portal 2.7.0.

Das Kapitel erwähnt mehrere DTD-Dateien. In der Version 2.7 lagen diese Dateien im Quellcode unter *core/src/resources/portal-core-sar/dtd* und die Versionsnummer wurde auf 2_6 umgestellt. So gilt zum Beispiel der Text über die *jboss-app_2_0.dtd*-Datei auf Seite 310, vorletzter Absatz, für Portal 2.6, aber in Portal 2.7 befindet sich die Datei in *core/src/resources/portal-core-sar/dtd/jboss-app_2_6.dtd*.

Kapitel 11

Seite 330, Abschnitt 11.4.1, letzter Absatz

Die *overview.xhtml*-Datei für Portal 2.7.0 liegt unter *jboss-portal.sar/portal-identity.sar/portal-identity.war/jsf/register*.

Kapitel 12

Seite 365, zweiter Absatz unter Abbildung 12.6, letzter Satz (vor Überschrift “Entity-Daten im Cache entwerten”)

Änderung: Alles hinter dem Wort “speichern” entfernen.

Grund: Im GA-Release ist die Buddy-Replikation für HTTP-Session-Replikation deaktiviert.

Seite 371, Abschnitt 12.2.2, erster Absatz, zweiter Satz

Änderung: Der Satz soll beginnen mit: „Da der Server geclustert ist und der Client auf eine SLSB zugreift, ruft der...“

Grund: Damit wird klargestellt, dass Round-Robin-Lastverteilung verwendet wird, weil wir auf eine SLSB zugreifen. Für SFSBs wird das Round-Robin-Verfahren nicht verwendet.

Seite 371, Abschnitt 12.2.3, erster Absatz, letzter Satz

Änderung: Der letzte Satz über den Farming-Service von JBoss wird gestrichen.

Grund: Dieser Service steht in JBoss AS 5 nicht mehr zur Verfügung.

Seite 373, Tabelle 12.2

Änderung: Die Tabelle sollte wie folgt lauten:

Node 1: Konsolenausgabe	Node 2: Konsolenausgabe
INFO [STDOUT] 0	INFO [STDOUT] 1
INFO [STDOUT] 2	INFO [STDOUT] 3
INFO [STDOUT] 4	INFO [STDOUT] 5
INFO [STDOUT] 6	INFO [STDOUT] 7
INFO [STDOUT] 8	INFO [STDOUT] 9
INFO [STDOUT] 10	...

Grund: Die Standardstrategie für Lastverteilung ist Round-Robin, aber die Tabelle im Buch zeigt Random-Robin-Lastverteilung an.

Seite 376, Abschnitt 12.3.2, erster Absatz

Änderung: Die Datei *server/all/deploy/cluster/cluster-jboss-beans.xml* soll heißen *server/all/deploy/cluster/hapartition-jboss-beans.xml*

Seite 376, Absatz nach der Tabelle

Änderung: Die Datei *cluster-jboss-beans.xml* soll heißen *hapartition-jboss-beans.xml*

Seite 378, erster Absatz, letzter Satz

Änderung: Soll geändert werden in: „Alle Services in dieser Datei sind so vorkonfiguriert, dass sie den UDP-Protokollstapel nutzen“.

Grund: Der JBoss Messaging Post Office-Service ist in dieser Datei nicht definiert und verwendet nicht den UDP-Protokollstack.

Seite 378, Absatz vor Abbildung 12.12

Änderung: Die Datei *cluster-jboss-beans.xml* soll heißen *hapartition-jboss-beans.xml*

Seite 379, Abschnitt 12.3.3, Absatz vor Listing 12.4, erster Satz

Änderung: Es wird angefügt „..., (mit Ausnahme des JBoss Messaging Post Office-Service, der den `jbm-data`-Stapel verwendet).“

Seite 380, erster Absatz, zweiter Satz

Änderung: Die Behauptung, dass TCP zuverlässiger sei als UDP, wird entfernt. Der Satz sollte beginnen mit “TCP wird von den meisten Computern unterstützt...”

Grund: Wir sagten, dass TCP im allgemeinen zuverlässiger ist als UDP, was auch meistens stimmt, aber bei JGroups verfügt UDP über die Protokolle NAKACK und UNICAST, die es genauso zuverlässig machen wie TCP.

Seite 383, Tabelle 12.5, HA Partition-Cache

Änderung: Die Konfigurationsdatei soll geändert werden in:
deploy/cluster/hapartition-jboss-beans.xml

Seite 395, Abschnitt 13.2.1, dritter Absatz

Änderung: Die Datei soll heißen: *server/all/deploy/cluster/jboss-cache-manager.sar/META-INF/jboss-cache-manager-jboss-beans.xml*

Seite 395, Abschnitt 13.2.1, vierter Absatz

Änderung: `CacheMode` soll heißen `cacheMode` und `SyncReplTimeout` soll heißen `syncReplTimeout`.

Seite 395, Abschnitt 13.2.1, vierter Absatz, letzter Satz

Änderung: Soll heißen: „Die Cache-Buddy-Replikation ist deaktiviert; wenn Sie sie aktivieren, müssen Sie beachten, dass sie für die Replikation mit nur einem einzigen Knoten konfiguriert ist. Das genügt für die meisten Einsätze, aber wenn es nötig ist, kann die Zahl der Knoten auch erhöht werden.“

Kapitel 13

Seite 401, Abschnitt 13.3.2, erste Zeile im Codestück

Änderung: `@Zustandsbehaftet` soll heißen `@Stateful`.

Seite 405, Abschnitt „Entities für die Cache-Nutzung konfigurieren“, Absatz unter dem Code, letzter Satz

Änderung: Soll heißen: „Die von Hibernate unterstützten Optionen sind in Tabelle 13.6 aufgeführt.“

Grund: Hibernate und nicht JBoss Cache ermöglicht diese Strategien.

Seite 407, letzter Absatz, vorletzter Satz

Änderung: Der Satz “Die Suche endet, sobald ein Lookup Erfolg hatte” wird gestrichen.

Grund: Dieses Feature steht in JBoss AS 5.0 nicht zur Verfügung, soll aber mit JBoss AS 5.1 CR1 wieder eingefügt werden. Aktuell wird jeder Knoten durchgesehen, bevor eine Antwort zurückgegeben wird. Siehe auch: *https://jira.jboss.org/jira/browse/JBAS-5703*

Seite 408, Abschnitt 13.5.2, erster Absatz

Änderung: Die Datei *server/all/deploy/cluster/cluster-jboss-beans.xml* soll heißen *server/all/deploy/cluster/hapartition-jboss-beans.xml*

Seite 408, Tabelle 13.7, bindAddress

Änderung: „Bleibt der Wert leer (die Standardeinstellung)“ wird ersetzt durch „Wenn die Option `-b` nicht angegeben wird“

Seite 409, Abschnitt „Automatische Discovery von Knoten“, erster Absatz, zweiter Satz

Änderung: Das Wort „JGroups“ wird aus dem Satz gestrichen.

Grund: HA-JNDI verwendet Multicasting, baut aber nicht auf JGroups auf.

Seite 410, Satz vor der Überschrift „Auf HA-JNDI mit dem Server zugreifen”

Änderung: Dem Satz wird hinzugefügt „..., und wird nicht für Umgebungen empfohlen, in denen Clients und Server durch Firewalls getrennt sind.“

Kapitel 14

Seiten 425 und 426, Abschnitte 14.5 und 14.5.1

JBoss AS 5.1.0: Für Windows werden die `JAVA_OPTS` jetzt in der *bin/run.conf.bat*-Datei eingestellt.

Seite 427, Tabelle 14.2

Beachten Sie, dass dies für `MaxPermSize` in *run.bat* und *run.conf* auf 256MB festgelegt ist, was für die meisten Zwecke ausreicht. Die Standardeinstellung des JDK beträgt normalerweise 64MB, was für die Ausführung eines Anwendungsservers nicht genug ist.

Seite 429, Abschnitt „Die Collector-Typen verstehen“

Der letzte Satz des ersten Absatzes stimmt nicht: Seit JDK 1.5.0_06 ist das neue Argument `-XX:+UseParallelOldGC` verfügbar, mit dem man einen Parallel-GC für eine größere Collection einrichten kann.

Seite 430, Der CMS-Collector

Wir empfehlen in einer Cluster-Umgebung den CMS-Collector, denn wenn eine größere Collection zu lange dauert, können bei JGroups fälschlich „Fehler“ entdeckt werden, die keine sind.

Seite 443, Abschnitt 14.6.2, erster Absatz

Die *server.xml*-Datei befindet sich in *server/xxx/deploy/jbossweb.sar*.

Seite 443, Tabelle 14.9

Die neueste Version von JBoss Web Server kennt die `maxSpareThreads`-Option nicht. Stattdessen müssen Sie einen Executor definieren, wie in *http://tomcat.apache.org/tomcat-6.0-doc/config/executor.html* beschrieben. Der folgende Executor reduziert beispielsweise die Zahl der Threads auf 50, wenn das System 60 Sekunden lang nicht mehr als 50 Threads benutzt:

```
<Executor name="executor1" namePrefix="EXEC-http-"
  maxThreads="200" minSpareThreads="50" maxIdleTime="60000" />
```

Diesen Executor können Sie dann im HTTP-Connector referenzieren:

```
<Connector protocol="HTTP/1.1" executor="executor1" ... />
```

Wir danken *sra78* für den Hinweis in den JBoss-Foren.

Seite 448, Tabelle 14.12

Der *CompilerThread1* wird vom Just-In-Time (JIT)-Compiler verwendet, um den Java-Bytecode in maschinenspezifische Opcodes umzusetzen, aber nicht, um JSPs zu kompilieren.

Kapitel 15

Seite 459, Abschnitt 15.2.2, zweiter Absatz

Die Datei *bindings.xml* liegt jetzt unter *server/xxx/conf/bootstrap* (siehe auch B.1.5).

JBoss AS 5.1.0: Die Datei *server/xxx/conf/bindingservice.beans/META-INF/bindings-jboss-beans.xml* enthält die Portzuweisungen, die früher, in den Versionen in 5.0.0 und 5.0.1, unter *server/xxx/conf/bootstrap/bindings.xml*-Datei gespeichert waren. Das Verfahren zum Definieren oder Verwenden der Portzuweisungen bleibt gleich.

Unmittelbar vor GA wurde in der Konfigurationsdatei des Bindungsservice einiges geändert. Im Großen und Ganzen gilt der Text in Abschnitt 15.2.2 weiter, aber einige Details haben sich geändert:

- Die Standard-Portbindungen sind in der Bean `StandardBindings` angegeben.
- Die Zuordnungen des Namens zur Systemeigenschaft `jboss.service.binding.set` sind jetzt als Parameter der jeweiligen Bindungs-Bean definiert. So hat die Bean `Ports01Bindings` zum Beispiel als ersten Parameter den Wert `ports-01`.

Der JNDI-Port ist durch die MBean `jboss:service=Naming` definiert, die in *server/xxx/conf/service.xml* deklariert ist, und nicht durch die `RemoteNamingBean`, die in der Datei *server/xxx/deploy/naming-jboss-beans.xml* definiert ist. (Hinweis: Zwischen 5.0.0.CR2 und GA wurde erfolglos versucht, die MBeans in *jboss-service.xml* durch eine Reihe von servicespezifischen **-jboss-beans.xml*-POJO-Konfigurationsdateien zu ersetzen, und *naming-jboss-beans.xml* war eine dieser Dateien. Nach einigen Rückschlägen kehrte man wieder zur Datei *jboss-service.xml* zurück, aber zu spät, um auch den Abschnitt dieses Buches wieder auf den Ursprungstext umstellen zu können.)

Seite 467, Tabelle 15.3

Für den Monitoring Service soll die Datei *server/xxx/lib/Jboss-monitoring.jar* heißen *server/xxx/lib/jboss-monitoring.jar*

Seite 470, Tabelle 15.4

Die Konfigurationsdatei für den Timer Service für EJB 2.x ist *server/xxx/deploy/ejb2-timer-service.xml*.

Seiten 471-472, Abschnitt 15.5.1

Die Datei *ejb3-timer-service.xml*-Datei wurde im Release JBoss AS 5.0.1.GA fallengelassen. Daher sind für diese Version die in diesem Abschnitt skizzierten Schritte nicht notwendig, um die DefaultDS-Datenquelle zu ändern.

Seite 473, Abschnitt 15.6.1, zweiter Absatz

`_JAVA_LOADER_DEBUG` wird ersetzt durch `_JAVA_LAUNCHER_DEBUG`.

Seit JBoss AS 5.0.0 sind die Dateien *service.bat* und *jbosssvc.exe*, die eingesetzt werden, um den Anwendungsserver als Windows-Dienst auszuführen, im Anwendungsserver inbegriffen. Die Datei *jbosssvc.exe*-Datei ist jedoch in den Versionen 5.0.0 und 5.0.1 schadhaft und funktioniert daher nicht. Also müssen Sie die Datei immer noch von JBoss Native beschaffen. Die *jbossvc.exe*-Datei in 5.1.0 ist jedoch einwandfrei und kann durchaus genutzt werden, um den Anwendungsserver als Windows-Dienst auszuführen.

Anhang B

Seite 490, Abschnitt „Embedded Jopr installieren und konfigurieren“

Der Name der WAR-Datei für Embedded Jopr hat sich im 1.1-Release abermals geändert, aber zum Glück existiert eine *jboss-web.xml*-Datei, die den Kontext */admin-console* deklariert. Daher sind die von uns angegebenen URLs jetzt korrekt! :-)

JBoss AS 5.1.0: Embedded Jopr ist jetzt im JBoss AS inbegriffen und muss nicht separat installiert werden. Es ist unter *server/xxx/deploy/admin-console.war* zu finden.

Seite 495 unten

JBoss AS 5.1.0: Auf Windows werden die `JAVA_OPTS` nunmehr in der *bin/run.conf.bat*-Datei eingestellt.

Seite 496, Abschnitt „Die Webkonfiguration”

Mit GA kam noch eine andere Konfiguration auf, nämlich *standard*. Das ist die Konfiguration, die an Java EE 5 JCK übergeben wird. Mehr darüber erfahren Sie in der *readme.html*-Datei.

Seite 499 oben

JBoss AS 5.1.0: Der Standard-Log-Level ist `INFO`.