Table of Contents

Head First EJB™ By Bert Bates, Kathy Sierra ISBN: 0596005717 Publisher: O'Reilly
Print Publication Date: 2003/10/01

12 a bean's environment

The Joy of Deployment

You worked hard on that bean. You coded, you compiled, you tested. About a hundred zillion times. The *last* thing you want to touch is already-tested source code, just because something simple changed in the deployment configuration. Maybe the name of the database is different. Maybe you hard-coded a tax-rate (remember—bean's can't access property files). And what if you don't even *have* the source code? Maybe you got it from Beans-r-Us, and they won't sell you the source (not on *your* budget, anyway). EJB supports bean reuse through the Deployment Descriptor and a bean's special *environment.*

***exam** objectives*

A Bean's Environment

Official:	*What it really means:*
13.1 Identify correct and incorrect statements or examples about an enterprise bean's JNDI naming.	You must know the ways in which a bean can be customized at deployment time, without changing source code. When a bean does a JNDI lookup using the bean's own special environment, you must know how the code relates to what's in the deployment descriptor, and a bunch of finicky details that mean the difference between a successful deployment or one that won't deploy. Or much worse... one that deploys but blows up sometime later, while its running.
13.2 Identify correct and incorrect statements about the purpose and/or use of the deployment descriptor elements for environment entries, EJB references, resource manager connection factory references, including whether a given code listing is appropriate and correct with respect to a particular DD element.	You need to know what *is* and is *not* guaranteed by the EJB 2.0 specification. For example, an EJB 2.0 container is *not* required to support JXTA, JMX,Servlets or JSPs, but it *is* required to support JNDI, JMS, and JAXP. You need to know what EJB really gives you.
13.3 Given a list of responsibilities, identify which belong to the Deployer, Bean Provider, App Assembler, Container, Sys Admin, or any combination.	You also need to know what you can and cannot do in EJB, if you want to make a bean that is portable across all EJB 2.0-compliant containers. For example, you can't *listen* on a ServerSocket, but you *can* create a client Socket. You *can't* access the server's local file system, or make your own threads. But you *can* make your bean class extend another.
1.2 Given a list of technology specifications, identify which are requirements for an EJB 2.0 container.	You need to know about the EJB roles (Bean Provider, Application Assembler, etc.) and who does what during development and deployment. Although you won't have many questions about this, it's an area the exam beta testers struggled with, so if we were you, we'd study this part carefully.
1.3 Identify correct and incorrect statements or examples about EJB programming restrictions.	
1.4 Match EJB roles with the corresponding description of the role's responsibilities, where the description may include deployment descriptor information.	
1.5 Given a list, identify which are requirements for an ejb-jar file.	Finally, you must know what *is* and is *not* supposed to be in an ejb-jar file. For example, the home and component interface ***must*** be there (unless you're using a message-driven bean), but the stubs must ***not*** be there.

Head First EJB™ By Bert Bates, Kathy Sierra ISBN: 0596005717 Publisher: O'Reilly
Print Publication Date: 2003/10/01

Raise your hand if you like changing your source code every time you need to change the way your bean behaves.

Of course you don't like going back to your source code just because, say, somebody changed the name of the database. You already know that you can change transaction attributes and security access just by tweaking the deployment descriptor. But wait... *there's more!*

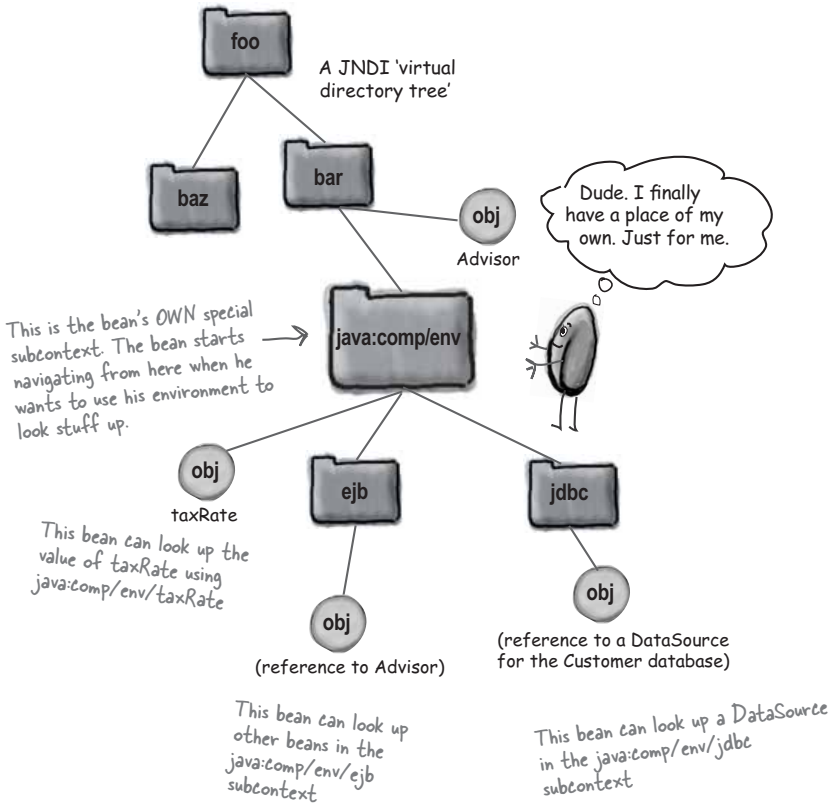
Every EJB container gives your bean its own special environment, that your bean can use to look up four different things:

- **Environment Entries**
  These are deploy-time values (so you don't have to code in values like the current discount percentage or tax rate).
- **Resource Manager Connection Factories**
  You'll use these to get a connection to a database.
- **Enterprise Bean References**
  When one bean wants to look up another bean. You'll do this a lot! Most designs involve at least some level of bean-to-bean communication.
- **Resource Environment References**
  You'll use this to get a reference to something called "an administered object" from the server, like a JMS destination.

Head First EJB™ By Bert Bates, Kathy Sierra ISBN: 0596005717 Publisher: O'Reilly
Print Publication Date: 2003/10/01

A bean's special place- java:comp/env

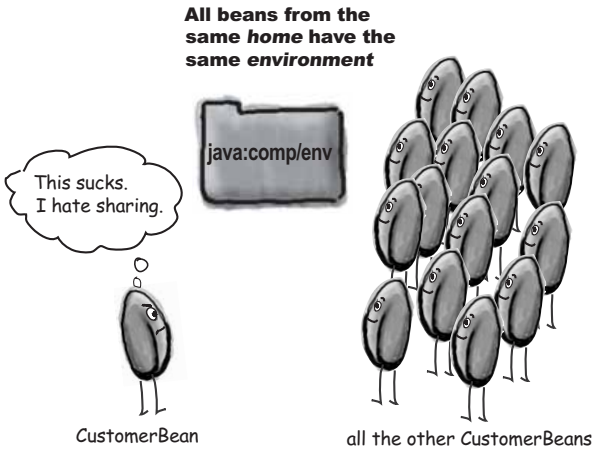
Your bean is entitled to a JNDI context all its own. A special place that's just for your bean, where your bean can look things up. It all starts with an InitialContext, but the java:comp/env subcontext is where the bean begins navigating when he wants to look something up.



Head First EJB™ By Bert Bates, Kathy Sierra ISBN: 0596005717 Publisher: O'Reilly
Print Publication Date: 2003/10/01

But it's not per bean instance... It's per bean home

The scope of a bean's environment is for all beans from the same home. If you deployed a particular bean type into your server three times, each of the three deployments would have it's own unique home, so there'd be three different environments.




```
InitialContext ic = new InitialContext();
Double discount =  (Double) ic.lookup("java:comp/env/custDiscount");
```

This value will be the same no matter which instance of CustomerBean runs this code.

Head First EJB™ By Bert Bates, Kathy Sierra ISBN: 0596005717 Publisher: O'Reilly
Print Publication Date: 2003/10/01

***deploy-time** customization*

Environment subcontexts

java:comp/env/jdbc/CustomerDB

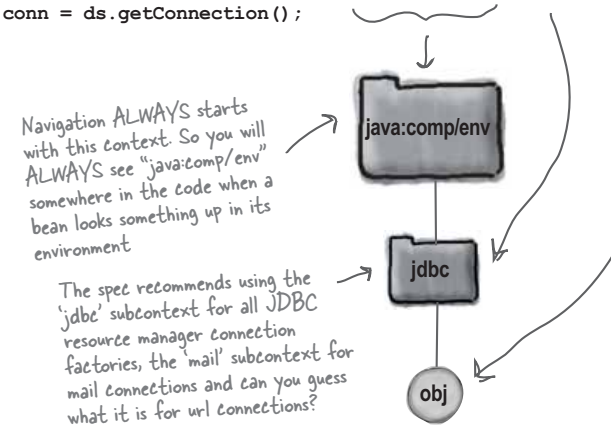
the private environment for this bean type

A subcontext we named 'jdbc' by convention (although 'jdbc' is not required, and we could have said java:comp/env/CustomerDB

The name of the 'thing' you're looking up (in this case, resource connection factory to the 'CustomerDB' database)

in a bean's business method:

```
InitialContext ic = new InitialContext();
DataSource ds =  (DataSource) ic.lookup("java:comp/env/jdbc/CustomerDB");
Connection conn = ds.getConnection();
```



Navigation ALWAYS starts with this context. So you will ALWAYS see "java:comp/env" somewhere in the code when a bean looks something up in its environment

The spec recommends using the 'jdbc' subcontext for all JDBC resource manager connection factories, the 'mail' subcontext for mail connections and can you guess what it is for url connections?

a reference to the connection factory for the customer database named "CustomerDB"... or is it?

how the heck does the programmer know the name of the database??

How can we deal with this? How does the programmer hard-code a lookup String like "java:comp/env/jdbc/CustomerDB" without knowing how the database was configured into the server?

(hint: how did we deal with it earlier when the Bean Provider used programmatic security with isCallerInRole("someRole")?)

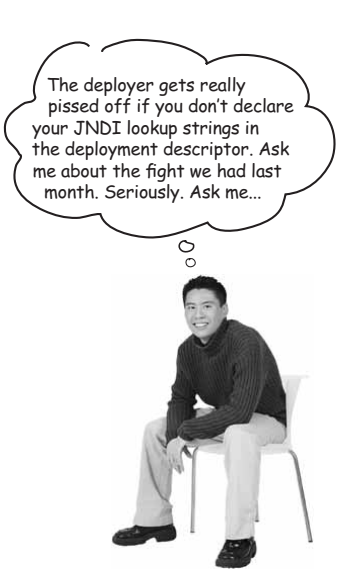
Head First EJB™ By Bert Bates, Kathy Sierra ISBN: 0596005717 Publisher: O'Reilly
Print Publication Date: 2003/10/01

It's simple...

if the programmer puts a made-up JNDI name in code, he has to announce that to the deployer in the DD.

The Deployer has *no* clue what the Bean Provider put in code, unless the Bean Provider:

A. Tells him.
B. Writes the names on a sticky note and sticks it on the Deployer's monitor
or
C. Declares references in the Deployment Descriptor, complete with helpful descriptions that make it very clear what the Deployer is supposed to map to the programmer's made-up names.



Declares the made-up JNDI names in the Deployment Descriptor, for the Deployer.

Maps the programmer's made-up/ fake names to the REAL JNDI names under which the resources were deployed or configured into the server.

Head First EJB™ By Bert Bates, Kathy Sierra ISBN: 0596005717 Publisher: O'Reilly
Print Publication Date: 2003/10/01

Environment entries: deploy-time constants

Imagine you're writing a simple checkout bean for an online shopping cart system. You don't know where that bean might end up. Even if you do, you know that things like tax rates and discount policies can change, even within the same company.

Environment entries let you write your code using a variable that you'll fill in at runtime using a JNDI lookup. Remember, the Bean Provider chooses the name, and it's up to the Deployer to fill in the value.

this is the part of the lookup that must go in the DD

in a bean's business method:

```
InitialContext ic = new InitialContext();
Double dbl =  (Double) ic.lookup("java:comp/env/smartCustomerDiscount");
customerDiscount = dbl.doubleValue();
// use the primitive double value to calculate the discount
```

You do NOT put the "java:comp/env/" part in the deployment descriptor. Ever! In the DD, you put ONLY the part that comes after the "java:comp/env/"

in the Deployment Descriptor

```
<entity>
    ...
    <env-entry>
      <description>discount for smart customers</description>
      <env-entry-name>smartCustomerDiscount</env-entry-name>
      <env-entry-type>java.lang.Double</env-entry-type>
      <env-entry-value>0.05</env-entry-value>
    </env-entry>
    ...
</entity>
```

if you have any compassion at all, you'll put in a description for the poor Deployer.

The programmer made up this name

the type must be either a String or a wrapper class

The value is optional for the Bean Provider, but he can use this element to supply a default. But the Deployer MUST ensure that there's a valid value before the bean is deployed.

Environment entries are different from the other customizations in that the Deployer doesn't map from the Bean Provider's name to some other real name. Environment entries don't exist until the Bean Provider says they do, by putting in the <env-entry>. As long as the <env-entry> has a value when its deployed, the environment entry will be created.

You can't change a value dynamically!

Once a bean has been deployed, there is NO WAY to change the value of the environment variable! The only way to update the value a bean sees is to redeploy the bean with the new DD.

Head First EJB™ By Bert Bates, Kathy Sierra ISBN: 0596005717 Publisher: O'Reilly
Print Publication Date: 2003/10/01

there are no Dumb Questions

Q: Why can't you just use Java property files instead of environment entries?

A: You're not allowed to use properties because you can't access the file system to read them in, and you don't have any control over system properties (for example, you have no way to set a property as a JVM command-line argument.)

Q: Why can't I have an environment entry that's shared among multiple beans in the same app? Or at least within the same ejb-jar?

A: Because... because you can't. There's simply no mechanism for sharing the bean's environment because that would defeat the whole purpose of the bean having his own private space. Having a bean's environment prevents what would be an extremely likely disaster: naming collisions between different beans! In other words, different beans deployed with environment entries (or other references) that use the same name.

If you have a shared resource or environment entry, you MUST configure it with each bean you deploy. This is NOT per Deployment Descriptor, but per every individual bean, regardless of where the bean lives.

Environment entries must be one of these types:

- String
- Byte
- Short
- Integer
- Long
- Float
- Double
- Character
- Boolean

Only Strings and wrappers are allowed for environment entries

Watch it!

You MUST know the scope of an environment entry!

You're expected to know that environment entries are private and unique to each home. If you deploy an environment entry 'foo' with a value of 10, for the Customer bean, all instances of CustomerBean will get that same value when they do a lookup using java:comp/env/foo from a business method.

But your environment entry 'foo' does NOT cause a naming collision with any other BEAN type that's been deployed with the same environment entry name. *Customer bean's 'foo' is not the same as Advice bean's 'foo', even though the lookup strings are identical: ic.lookup("java:comp/env/foo") and even if both beans are in the same ejb-jar or enterprise archive (.ear).*

But we're not done yet... a single bean type must NOT be deployed with more than one 'thing' of the same name. So you can't call an environment entry 'foo' and a resource manager connection factory 'foo' in the Customer bean. As long as you use subcontexts, though, you're OK. Because 'jdbc/foo' is different from just 'foo' alone. So you CAN have both:
java:comp/env/foo
and
java:comp/env/jdbc/foo
but NOT two things named:

java:comp/env/foo, even if those two things are different resource types.

Head First EJB™ By Bert Bates, Kathy Sierra ISBN: 0596005717 Publisher: O'Reilly
Print Publication Date: 2003/10/01

It's subcontexts all the way down

Remember, your java:comp/env space is just a subcontext. A special one, sure, but still a context. So you can save that in a Context variable, and save yourself from having to retype "java:comp.env" every frickin' time you want to look something up.

Saving your special environment context:

```
InitialContext ic = new InitialContext();
Context mySpecialPlace = (Context) ic.lookup("java:comp/env");
// now look something up on mySpecialPlace
Double dbl =  (Double) mySpecialPlace.lookup("smartCustomerDiscount");
```

Using a subcontext

taxInfo is a subcontext, of the bean's environment.

```
InitialContext ic = new InitialContext();
Context myTaxInfoCtx = (Context) ic.lookup("java:comp/env/taxInfo");
// now look something up on myTaxInfoCtx subcontext
Double dbl =  (Double) myTaxInfoCtx.lookup("taxRate");
```

Note: you don't need to narrow anything coming out of JNDI lookups EXCEPT for stubs (Remote home interface references), so all you need is a plain old cast.

taxRate is an object within the taxInfo subcontext

Creating a subcontext

In the DD for the environment entry name, anything followed by a slash automatically becomes a subcontext!

A subcontext exists simply because you say it does. If you type "java:comp/env/foo/bar" into your lookup code, you've said, "There is a subcontext in my environment named "foo", and it contains the object "bar". As long as you use the same subcontext in the deployment descriptor when you deploy the bean, the subcontext will magically exist, just because you said it does. In other words, you don't have to go through a process of somehow creating a new JNDI context and naming it. Act like it's there, and it's there. Don't you wish everything worked that way?

```
<env-entry-name>taxInfo/taxRate</env-entry-name>
```

putting "taxInfo/" before "taxRate" automatically creates the taxRate subcontext when this bean is deployed

Resource manager connection factories (think: *database*)

Any bean can use a database. In fact, even an entity bean with container-managed persistence can get a connection to a database, as long as its using that database for something other than managing its own persistence. The code is simple: look up a DataSource, and ask it for a Connection. Once you have a Connection, you can send JDBC statements to do your SQL. Although javax.sql.DataSource is by far the most commonly-used resource manager connection factory, you can have others including a mail or URL connection.

this is the part of the lookup that must go in the DD

in a bean's business method:

```
InitialContext ic = new InitialContext();
DataSource ds =  (DataSource)ic.lookup("java:comp/env/jdbc/CustomerDB");
Connection conn = ds.getConnection();
// use the connection to do JDBC
```

in the Deployment Descriptor

Be SURE to leave off the java: comp/env/ part, and begin with the first subcontext (jdbc)

```
<entity>
    ...
    <resource-ref>
        <res-ref-name>jdbc/CustomberDB</res-ref-name>
        <res-type>javax.sql.DataSource</res-type>
        <res-auth>Container</res-auth>
        <res-sharing-scope>Shareable</res-sharing-scope>
    </resource-ref>
    ...
</entity>
```

Programmer's made-up name. NOT the read DB name!.

the resource manager connection factory type

Who checks to see if the user has the right security access for the database? Either the Bean or the Container

<res-sharing-scope> is optional, and defaults to "Shareable". It means that other beans in the same app, using the same resource, in the SAME transaction, can use the same connection. To disable sharing, the value should be "Unshareable"

Head First EJB™ By Bert Bates, Kathy Sierra ISBN: 0596005717 Publisher: O'Reilly
Print Publication Date: 2003/10/01

Resource manager connection factory types

You can't put arbitrary types into the DD. For the four standard resource manager connection factories (five, if you count JMS topics and queues as two different types), you must use one of the following:

1. `<res-type>javax.sql.DataSource</res-type>`

2. `<res-type>javax.jms.QueueConnectionFactory</res-type>`
   `<res-type>javax.jms.TopicConnectionFactory</res-type>`

3. `<res-type>javax.mail.Session</res-type>`

http://www.headfirst.com

4. `<res-type>javax.net.URL</res-type>`

Although these four are the only types standard to EJB 2.0, you can use the Connector architecture if you need access to other resources, like legacy systems. Connectors are out of scope for this book (and the exam), so you can relax. But you do need to know it's out there, if you need it.

Resource authorization

Authentication to the EJB server itself is one thing, but chances are, the database has its own log-in scheme. A user might need a log-in name and password that is different from the one he uses to authenticate to the EJB server.
As a Bean Provider, you can choose between two ways to give the resource manager (such as a database) the user's log-in data.

1. `<res-auth>Container</res-auth>`

   Container authorization means the Deployer must configure the sign-on information for the resource manager. It's completely vendor and resource-specific, and might mean that the deployer has to map between the principals and roles used in EJB security to whatever the resource manager needs. At the simplest level, the Deployer might specify a name and password that'll let anybody in.

2. `<res-auth>Bean</res-auth>`

   Bean authorization means the programmer uses the overloaded version of getConnection() that takes a name and a password:

```
Connection conn = ds.getConnection(userName, password);
```


Head First EJB™ By Bert Bates, Kathy Sierra ISBN: 0596005717 Publisher: O'Reilly
Print Publication Date: 2003/10/01

The complete resource mapping picture

fake JNDI name:

<resource-ref>**CustDB**</resource-ref>

actual JNDI name: **CustDatabase**

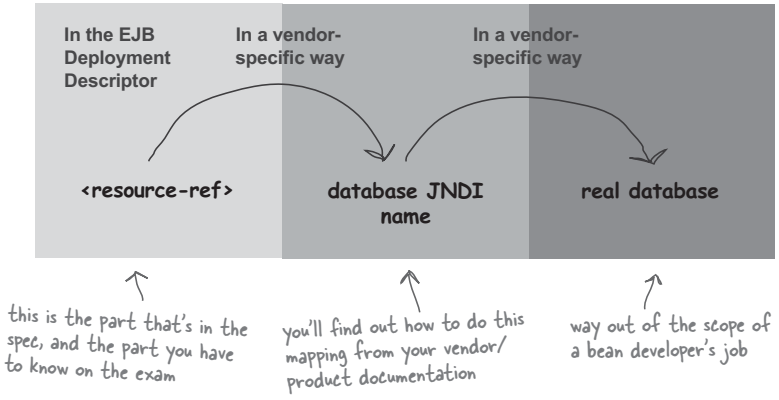
REAL database name: **CustomerData**

In code, the Bean Provider does a JNDI lookup on a DataSource (which he uses to get a database connection). He doesn't know the real JNDI name of the database (and he DEFINITELY doesn't know actual database name), so he makes one up. But he tells the deployer what he's done, by declaring a <resource-ref> element in the DD.

Deployer maps the Bean Provider's made-up <resource-ref> name to the actual JNDI name under which the DataSource is registered.

The Sys-Admin (or someone in the operational environment configures the database into the server, and gives it a JNDI name.

Where and how the mapping happens



EJB references (when a bean wants another bean)

Beans who need other beans are the luckiest beans in the server. But remember, beans have to go through the home interface just like everybody else. If Bean A wants to do a JNDI lookup on Bean B's home, we've got the same problem as always—what's Bean B's JNDI name? As a Bean Provider, you're just making one up. At deploy time, the Deployer will map your fake coded name to the real JNDI name matching a bean of the type you specified in the DD.

in a bean's business method:

"ejb/AdviceGiver" must go in the DD (without the quotes). Remember, when coding you have no idea what the REAL name will be...

```
InitialContext ic = new InitialContext();
Object o =  ic.lookup("java:comp/env/ejb/AdviceGiver");
AdviceHome home = (AdviceHome) PortableRemoteObject.narrow(o, AdviceHome.class);
Advice advisor = home.create();
// call methods on Advisor
```

in the Deployment Descriptor

Be SURE to leave off the java: comp/env/ part, and begin with the first subcontext (ejb)

Programmer's made-up name. NOT the real JNDI name of the bean.

```
<entity>
    ...
    <ejb-ref>
        <ejb-ref-name>ejb/AdviceGiver</ejb-ref-name>
        <ejb-ref-type>Session</ejb-ref-type>
        <home>headfirst.AdviceHome</home>
        <remote>headfirst.Advice</remote>
    </ejb-ref>
    ...
</entity>
```

this must be either Session or Entity

The fully-qualified name of the home interface

The fully-qualified name of the component interface

Head First EJB™ By Bert Bates, Kathy Sierra ISBN: 0596005717 Publisher: O'Reilly
Print Publication Date: 2003/10/01

deploy-time customization

EJB local references

Use the <ejb-local-ref> tag if you want to look up the bean's *local* home.

```
<ejb-local-ref>
    <ejb-ref-name>ejb/AdviceGiver</ejb-ref-name>
    <ejb-ref-type>Session</ejb-ref-type>
    <local-home>headfirst.AdviceHomeLocal</local-home>
    <local>headfirst.AdviceLocal</local>
</ejb-local-ref>
```

The sub-elements are different for the local interfaces: <local-home> instead of <home> and <local> instead of <remote>

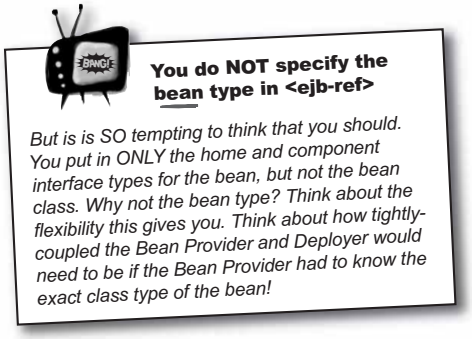
there are no Dumb Questions

Q: I understand the element names for <home> and <local-home>. Makes sense. But what's up with <remote> and <local>? Shouldn't it be <component> and <local-component>?

A: You can run from your past, but you can't hide. Before EJB 2.0, there was no concept of local interfaces. So the client view was always called Home and Remote. Although even that was an inconsistent name scheme, since both the home and the business method interface were Remote (as in java.rmi.Remote). But once local interfaces came on the scene, things got a bit more complicated. "Let's see... we can name the local home "local-home" and then we'll name the local remote "local-remote". See the problem? So with EJB 2.0, we stopped calling the business interface "the *remote* interface" and started calling it "the *component* interface". And now we call them "local component interfaces" and "remote component interfaces".

Q: Um, you still didn't answer my question. How come the tag still says "remote" for the remote component interface and "local" for the local component interface?

A: Because of the past. Backward compatibility and all that. In EJB 1.1, the <ejb-ref> tags said <home> and <remote>, so they still do. The naming scheme is basically this: "The component interface is either *local* or *remote*. If the tag- doesn't explicitly say "home", then you're talking about the component interface. These quirky little inconsistencies are just part of EJB's *charm*.



Head First EJB™ By Bert Bates, Kathy Sierra ISBN: 0596005717 Publisher: O'Reilly
Print Publication Date: 2003/10/01

Using <ejb-link> with EJB references

If the Application Assembler sees that one bean's EJB reference it to another bean in the same application, she should use the <ejb-link> to link the <ejb-ref> to *another bean specified in the deployment descriptor*. Think of <ejb-link> as a kind of "jump to this label" thing, where the value of the link matches the value of an <ejb-name> element somewhere else.

Somewhere in the DD

```
<entity>
    ...
    <ejb-ref>
        <ejb-ref-name>ejb/AdviceGiver</ejb-ref-name>
        <ejb-ref-type>Session</ejb-ref-type>
        <home>headfirst.AdviceHome</home>
        <remote>headfirst.Advice</remote>
        <ejb-link>AdviceEJB</ejb-link>
      </ejb-ref>
    ...
</entity>
```

The <ejb-link> MUST match the value of an <ejb-name> for some other bean in this DD (or another DD in the same J2EE app).

Somewhere *else* in the DD

```
<session>
    <ejb-name>AdviceEJB</ejb-name>
    ...
</session>
```

Remember, <ejb-name> is just a label in the DD, for other parts of the DD to refer to. It's not the real BEAN class name or anything (unless you happen to make your ejb-name the same as the bean class).

<ejb-name> must be unique within a single DD (which means a single ejb-jar), but not within a single J2EE app.

But this does NOT mean all beans in the same J2EE app must have unique <ejb-name> values. A single .ear can have multiple ejb-jar files, with one DD per ejb-jar, and each of the DD's in the app can have the same <ejb-name>. Ah.... but THEN there's a problem with <ejb-link>, which looks in the entire app, not just the current DD, for a matching <ejb-name>. Not to worry. If you DO have two DD's in the app with identical <ejb-name> entries, use the alternate <ejb-link> syntax to add the path to the ejb-jar where this link's matching <ejb-name> lives. The path (followed by "#") is relative to the current DD's jar.

<ejb-link>../custServices/advice.jar#AdviceEJB</ejb-link>

<ejb-link> values MUST match the value of some OTHER bean's <ejb-name>

And remember: <ejb-name> is nothing more than a label in the DD. It doesn't have to match the class name, interface name, JNDI name, or anything else. It's just the label in the DD for a particular bean.

Nobody but your co-workers will care if you name your beans after, say, your pets.

Head First EJB™ By Bert Bates, Kathy Sierra ISBN: 0596005717 Publisher: O'Reilly
Print Publication Date: 2003/10/01

Resource environment references (think: *JMS destinations*)

As a Bean Provider, you can look up two different kinds of resource-related things in JNDI: a resource manager connection factory reference, and a reference to something known as an *administered object*. The main difference is that a resource environment reference is to a *thing you want*, not the factory that gives you a *connection* to the thing you want. In other words, the administered object *is* the destination, whereas a resource manager connection factory reference is just the first step in getting what you really want (a *connection*).

But today, just do a mental search and replace in your mind so that everywhere you see resource environment reference, you substitute JMS destination. Because that's pretty much all you'll use it for now. Yes, they could have called it "JMS destination reference", but that would be too limiting for the future. Not to mention too clear, unambiguous, and meaningful to have any value whatsoever as a cognitive challenge.

in a bean's business method:

```
InitialContext ic = new InitialContext();
Object o =  ic.lookup("java:comp/env/jms/NewCustomerQueue");
javax.jms.Queue custQ = (javax.jms.Queue) o;
// use the custQ
```

You're now tired of hearing that you need to leave off the java:comp/env/ part, and begin with the first subcontext (jms)

in the Deployment Descriptor

```
<entity>
    ...
    <resource-env-ref>
        <resource-env-ref-name>jms/NewCustomerQueue</resource-env-ref-name>
        <resource-env-ref-type>javax.jms.Queue</resource-env-ref-type>
    </resource-env-ref>
    ...
</entity>
```

Programmer's made-up name. NOT the read JNDI name of the Queue

This could also be javax.jms.Topic (you're not limited to just JMS destinations, but that's the only standard thing we have right now.

Head First EJB™ By Bert Bates, Kathy Sierra ISBN: 0596005717 Publisher: O'Reilly
Print Publication Date: 2003/10/01

Bean Provider and Application Assembler
responsibility for the Deployment Descriptor

Don't worry about memorizing all of these now! It will make more sense as we get farther into the topics. For now, it's OK for you to have just an overall concept of what each is responsible for.

Bill puts in mostly things that are related to the code in the bean classes

Annie puts in mostly things about how two or more beans are related to one another in the application, and sometimes she customizes the bean info for a particular application.

Bean Provider

- bean name
- fully-qualified name of bean class and home and component interfaces
- bean type (session, entity, etc.)
- re-entrancy (for entity beans only)
- state management for session beans (stateless or stateful)
- transaction demarcation type (bean or container)
- entity bean persistence management (bean or container)
- primary key class
- for CMP, abstract schema name, CMP fields, CMR relationships, finder and select queries.
- resource manager connection factory *references*
- environment entry declarations
- EJB references (local and remote)
- security role *references*
- for message-driven beans: destination, message selector, and acknowledgement mode.

Application Assembler

Modifications to Bean Provider information:

- values of environment entries
- description fields (change or create)
- relationship name modifications
- message-driven bean message selector (may restrict, but not replace)

Application Assembly information (all optional):

- binding enterprise bean references (i.e. linking one bean to another in the same ejb-jar or J2EE app)
- security roles (the recommended roles for clients of the beans.)
- method permissions: a relationship between security roles and methods of the home and component interface of the bean
- linking security role *references* to security *roles*
- security identity type: *caller* or *run-as*
- transaction attributes for methods of a CMT bean

Head First EJB™ By Bert Bates, Kathy Sierra ISBN: 0596005717 Publisher: O'Reilly
Print Publication Date: 2003/10/01

Deployer responsibility for the Deployment Descriptor

Deployer

Modifications to Bean Provider information:

- ensure legal values for all environment entries

Other tasks related to the deployment descriptor. All are done in a vendor-specific tool and NOT a part of the ejb-jar deployment descriptor:

SECURITY

- assign of the security domain and principal realm to the app
- assign principals and/or groups to security roles, but NOT the security role *references*.
- principal delegation for inter-component calls (i.e. configuring the run-as principal).

The Deployer does things using vendor-specific tools

*You might see a question about the deployer and things related to the deployment descriptor and think: "The deployer isn't supposed to touch things in the DD, so this can't be his job." But... the deployer **does** have a lot of responsibilities **related** to things in the DD, so look carefully at the description of the task.*
*For example, who maps security **roles** to security role **references**? The App Assembler. Who maps **principals** to security **roles**? The Deployer.*

RESOURCE MANAGER CONNECTION FACTORIES

- binding of resource manager connection factory references to an actual resource manager connection factory in the operational environment
- configuration sign-on info for container-authorized resource access.

EJB REFERENCES

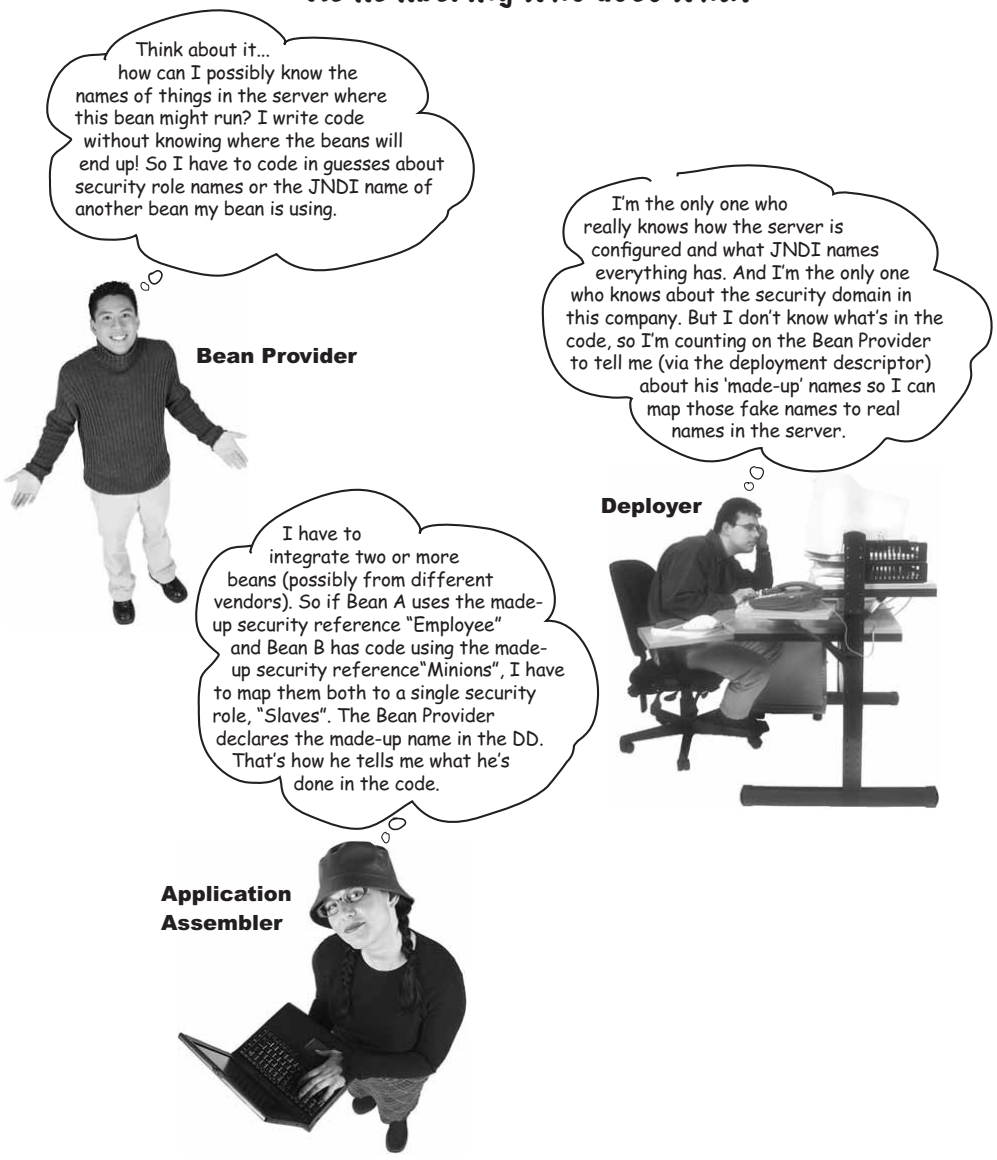
- ensure that all EJB references are bound to the homes of beans that exist in the operational environment
- ensure that the target bean is type-compatible with the types declared in the EJB reference.

RESOURCE ENVIRONMENT REFERENCES (JMS topic or queue)

- ensure that all declared resource references are bound to objects that exist in the operational environment, and ensure that the target object is type-compatible with the declared type

Head First EJB™ By Bert Bates, Kathy Sierra ISBN: 0596005717 Publisher: O'Reilly
Print Publication Date: 2003/10/01

Remembering who does what



Now let's look at the bean's runtime environment

We're almost done with the bean's world. Now that we've seen the bean'sspecial JNDI environment, we still have a few more little details on the bean's runtime environment. Each of these is covered by the exam objectives, so don't fall asleep now. We're almost done!

- **Guaranteed APIs**
  You must which APIs are and are not guaranteed to be part of every EJB 2.0 container. For example, JMS is supported, JXTA is not.

- **Guaranteed services**
  You must know what is and is not guaranteed to be supported by every EJB 2.0 container. For example, transaction support is guaranteed, load-balancing is not.

- **Structure of the ejb-jar**
  Maybe this isn't really a runtime environment thing, but we didn't have another good place to stick it, and you have to know it.
  So here it is, just in case you don't remember what we covered waaaaay back in chapter 1. For example, you must know that the an ejb-jar does not have a manifest, but MUST have a META-INF directory, and that directory MUST hold the deployment descriptor. Which, oh yes, MUST be named "ejb-jar.xml".

- **Programming restrictions**
  If you want your bean to be portable to / compatible with any EJB 2.0-compliant container, you must not do any of the things on the list, even if your vendor allows it (which the vendor may do *unintentionally*).
  And it's not just for portability, but for *safety*. If you try to manage your own threads, for example, you're stepping on the Container's toes, and who knows what kind of mess you'll end up with. You do not want to mess with the Container's job.
  Just because your Container permits it, doesn't make it right. And for the exam, you must know what's restricted in the spec.
  If the exam asks if something is possible, and it's one of the explicitly-restricted things in the spec, you must say NO, even if your vendor lets you do it. As far as the exam is concerned, if you can't do it and remain portable, *you can't do it.*

Which APIs does EJB 2.0 guarantee?

Supported APIs

- Java 2 Standard Edition, v 1.3 (**J2SE**)
- **JNDI** API 1.2
- **JTA** 1.01 extension (the **`UserTransaction`** interface only)
- **JDBC** 2.0 extension
- **JMS** 1.02 extension
- **JavaMail** 1.1, sending mail only
- **JAXP** 1.0

You don't have to memorize all of the exact version numbers

The exam isn't going to trick you on something as trivial as remember whether JavaMail is 1.1 or 1.2, or whether the JTA extension is 1.01 or 1.03. The only numbers you really need to know (besides the 2.0 in EJB 2.0!) are the JDBC 2.0 extension, and that only version 1.3 (not necessarily 1.4) of J2SE is guaranteed.

J2SE support is Java 1.3, NOT 1.4!

Just because an API is in the J2SE standard library for version 1.4, does not mean that API is guaranteed for EJB! The EJB spec requires only J2SE version 1.3, so if you write a bean that relies on something in J2SE 1.4, your bean is not guaranteed to be portable across any EJB 2.0 container.

That means, for example, that the Java Cryptography Extension (JCE) APIs are not guaranteed to be supported by EJB 2.0, even though they are now part of the J2SE platform as of version 1.4.

Without looking it up, write down what each of these APIs do. If you don't know, take a good guess based on what you know about EJB.

We'll start you off by giving you the most difficult one.

J2SE _____

JNDI _____

JTA _____

JDBC _____

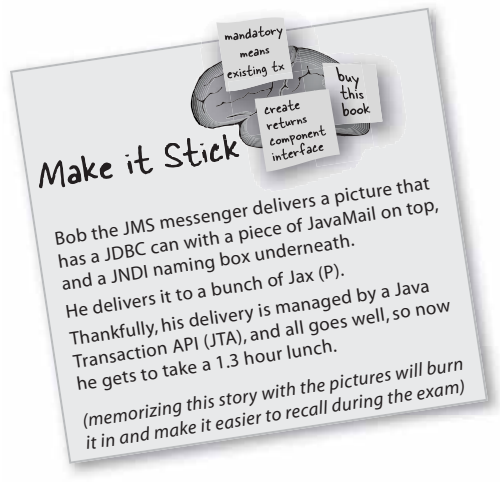
JMS _____

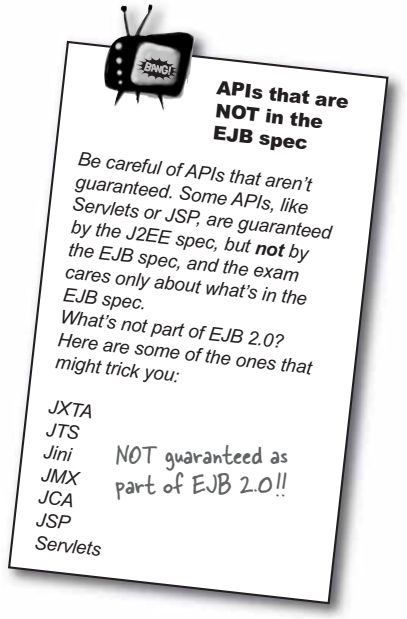
JavaMail sending mail

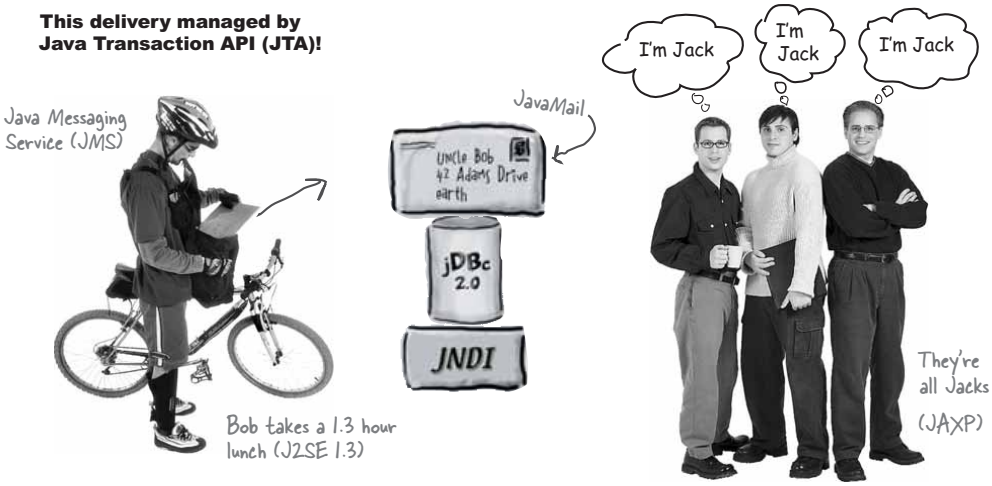
JAXP _____

Head First EJB™ By Bert Bates, Kathy Sierra ISBN: 0596005717 Publisher: O'Reilly
Print Publication Date: 2003/10/01

memorizing the APIs in EJB 2.0







Head First EJB™ By Bert Bates, Kathy Sierra ISBN: 0596005717 Publisher: O'Reilly
Print Publication Date: 2003/10/01

Some of the key, guaranteed services and behavior:

- Distributed **transactions**
- **Thread-safety**
- Container-managed **persistence** for entity beans
- A **security domain** and one principal realm (multiple realms is not guaranteed)
- **Enforce client access** security policies specified by the deployment descriptor and other deployment tools
- Implementation of the **java:comp/env** environment naming context provided to the bean
- **Generation of classes** that implement the home and component interfaces, and stub classes for remote objects.
- **Implementation of the** resource manager connection factory classes for resources configured with the server.

Some things sound good but aren't guaranteed!

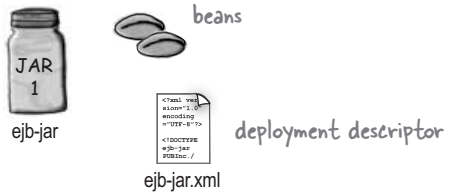
Everybody talks about their clustered, load-balanced, fault-tolerant system. Oh yeah, with fail-over, lazy-loading of entity data, and in-memory data caching.

Although most J2EE vendors provide one or more of these capabilities, they aren't guaranteed in the spec!!

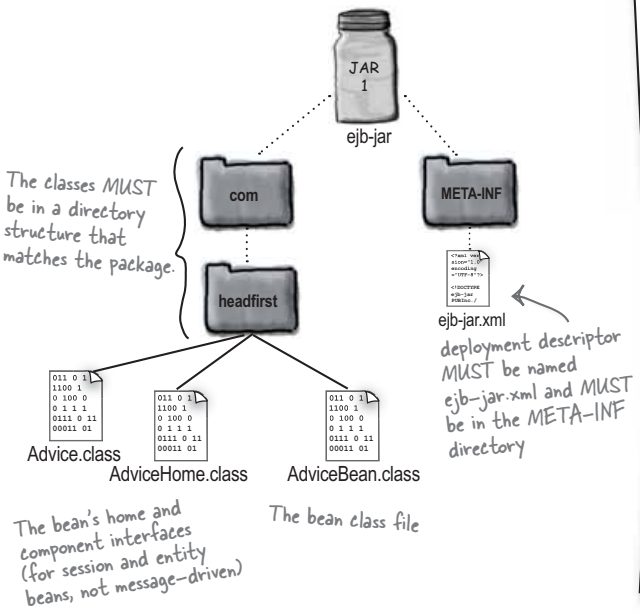
Be sure you know the difference between what is guaranteed and what is not. Look in the spec, under the sections titled "Container Provider responsibility"

Head First EJB™ By Bert Bates, Kathy Sierra ISBN: 0596005717 Publisher: O'Reilly
Print Publication Date: 2003/10/01

What MUST be in an ejb-jar?



Structure of an ejb-jar



You have to know what is NOT in the ejb-jar.

You're expected to know what's NOT in the ejb-jar file. Notice what's missing? The thing you expect to find in a JAR? Here's a hint...it's the thing that usually goes in the META-INF directory.

Figure it out? The manifest! The manifest file is not required as of EJB 2.0. You CAN put one in, but it's optional and usually not needed.

You also must know what MUST not be in the ejb-jar: classes generated by the container! Remember, the container implements the home and component interface, and makes stubs if the interfaces are remote. This doesn't happen until deployment... so they aren't part of what the Bean Provider or Application Assembler put into their deliverable—the ejb-jar file.

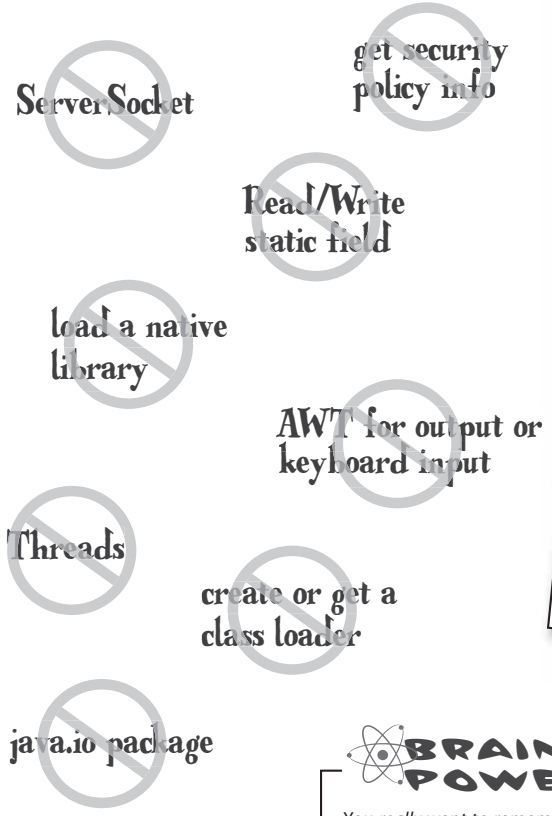
The naming convention is not required.

The naming convention for a bean is to put the component name as the component interface name, then add 'home' or 'bean' to come up with the other two names. We strongly recommend that you follow it; it makes deployment much easier and let's others read your code. But it's ***not a requirement.***

Head First EJB™ By Bert Bates, Kathy Sierra ISBN: 0596005717 Publisher: O'Reilly
Print Publication Date: 2003/10/01

Programming restrictions

What to avoid in EJB if you want to guarantee your bean can be deployed on ANY EJB 2.0-compliant server



Don't be fooled by things that only SEEM bad, but aren't

You're expected to know what you can and can't do in EJB 2.0. If you think about it too much, you can psych yourself into thinking (on the exam) that anything could be bad.

Here are a few things that you CAN do in EJB, that might trip you up on the exam:

***java.net.Socket** -- plain old client Sockets are fine. You just can't listen on a ServerSocket.*

***Extend another class** from your bean class -- normal java inheritance is OK for a bean class.*

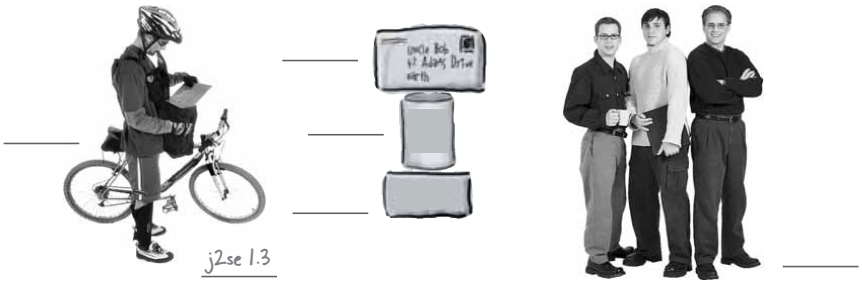
***Mark a bean field static and final** -- in fact, that's the only way you're supposed to mark a static field. (see the restriction about no read/write static fields.)*

You *really* want to remember these programming restrictions. The best way to make that happen? Stop right now and think about each of these restricted things, and come up with one or more reasons for why the restriction exists. When you're done, turn to page 494 in the EJB 2.0 spec, where these restrictions (and others) are described.

Head First EJB™ By Bert Bates, Kathy Sierra ISBN: 0596005717 Publisher: O'Reilly
Print Publication Date: 2003/10/01

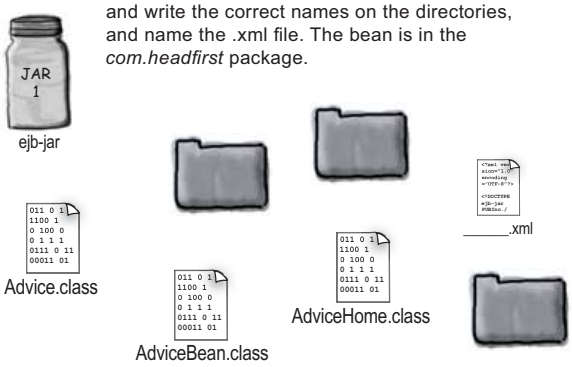
Memorize THIS

1. Looking at the picture below, see if you can tell the story, putting in the API's where they belong. We did one for you.

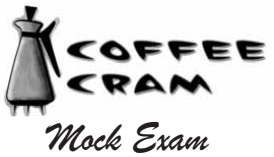


2. Using the pieces below (and ONLY the pieces below) reassemble them into their correct configuration (drawing lines as needed). Draw your finished structure in the space at the right, and write the correct names on the directories, and name the .xml file. The bean is in the *com.headfirst* package.

Draw the structure of the JAR file here:



Head First EJB™ By Bert Bates, Kathy Sierra ISBN: 0596005717 Publisher: O'Reilly
Print Publication Date: 2003/10/01



1 Which APIs are guaranteed to be supported by EJB 2.0 containers? (Choose all that apply.)

- ❑ A. JAXP
- ❑ B. JNDI
- ❑ C. JXTA
- ❑ D. JDBC
- ❑ E. JMS

2 What's true about an enterprise bean's environment? (Choose all that apply.)

- ❑ A. Environment entries can be unique for instances of the same enterprise bean type.
- ❑ B. Within a single EJB 2.0 container, an EJB can have multiple sets of environment entries.
- ❑ C. An EJB's environment entry's values can be modified by the EJB at runtime.
- ❑ D. Environment entry values may be primitives or wrapper types.

3 Which APIs are guaranteed to be supported by EJB 2.0 containers? (Choose all that apply.)

- ❑ A. J2SE 1.3
- ❑ B. JAXB 1.0
- ❑ C. JAXR 1.0
- ❑ D. JAXP 1.0

Head First EJB™ By Bert Bates, Kathy Sierra ISBN: 0596005717 Publisher: O'Reilly
Print Publication Date: 2003/10/01

4 Given a bean named 'Customer', and an environment entry named 'lastName', which code fragment(s) inside of the bean class would return the value of the environment entry? (Choose all that apply.)

❑ A.
```
Context c = new SessionContext();
Context e = (Context) c.lookup("java:comp/env");
String name = (String) e.lookup("lastName");
```

❑ B.
```
Context c = new InitialContext();
Context e = (Context) c.lookup("java:comp/env/
Customer");
String name = (String) e.lookup("lastName");
```

❑ C.
```
Context e = new Lookup("java:comp/env");
Context c = new InitialContext(e);
String name = (String) c.lookup("lastName");
```

❑ D.
```
Context c = new InitialContext("Customer");
Context e = (Context) c.lookup("java:comp/env");
String name = (String) e.lookup("lastName");
```

❑ E.
```
Context c = new InitialContext();
Context e = (Context) c.lookup("java:comp/env");
String name = (String) e.lookup("lastName");
```

5 When programming a session bean class which technique(s) should always be avoided to ensure bean portability across all EJB 2.0 containers? (Choose all that apply.)

❑ A. Using the `java.net.Socket` class.

❑ B. Using inner classes.

❑ C. Using the 'final' modifier for fields.

❑ D. Passing 'this' as an argument.

Head First EJB™ By Bert Bates, Kathy Sierra ISBN: 0596005717 Publisher: O'Reilly
Print Publication Date: 2003/10/01

6 Which of the following are valid data types in a **<env-entry-type>** element in a bean's deployment descriptor? (Choose all that apply.)

- ❑ A. **byte**
- ❑ B. **short**
- ❑ C. **ArrayList**
- ❑ D. **java.lang.Boolean**
- ❑ E. **java.lang.Character**

7 When programming EJBs which declaration(s) should be avoided to ensure bean portability across all EJB 2.0 containers? (Choose all that apply.)

- ❑ A. **final int x;**
- ❑ B. **static int x;**
- ❑ C. **final static int x;**
- ❑ D. **final transient int x;**

8 Who is typically responsible for specifying finder and select queries in the bean's deployment descriptor?

- ❑ A. The bean provider.
- ❑ B. The application assembler.
- ❑ C. The deployer.
- ❑ D. The system administrator.
- ❑ E. The server provider.

9 Which deployment descriptor element(s) would be used when obtaining a JDBC connection? (Choose all that apply.)

- ❑ A. **<ejb-ref>**
- ❑ B. **<ejb-link>**
- ❑ C. **<role-name>**
- ❑ D. **<env-entry>**
- ❑ E. **<resource-ref>**

Head First EJB™ By Bert Bates, Kathy Sierra ISBN: 0596005717 Publisher: O'Reilly
Print Publication Date: 2003/10/01

10 Who will typically merge multiple ejb-jar files into a single ejb-jar file.

- ❑ A. The bean provider.
- ❑ B. The application assembler.
- ❑ C. The deployer.
- ❑ D. The system administrator.
- ❑ E. The server provider.

11 Which deployment descriptor element(s) would be used by a bean provider to locate the home interfaces of other EJBs? (Choose all that apply.)

- ❑ A. **`<ejb-ref>`**
- ❑ B. **`<res-type>`**
- ❑ C. **`<env-entry>`**
- ❑ D. **`<role-name>`**
- ❑ E. **`<resource-ref>`**

12 Which are bean provider responsibilities concerning resource manager connection factory references? (Choose all that apply.)

- ❑ A. Configure resource managers in the EJB server.
- ❑ B. Configure sign-on information for the resource manager.
- ❑ C. Assign such a reference to the deployment descriptor.
- ❑ D. Creating a symbolic link to JNDI.

13 The ejb-jar file is considered to be part of the contract between which pairs? (Choose all that apply.)

- ❑ A. bean provider and system administrator
- ❑ B. bean provider and application assembler
- ❑ C. application assembler and deployer
- ❑ D. application assembler and system administrator
- ❑ E. deployer and system administrator

Head First EJB™ By Bert Bates, Kathy Sierra ISBN: 0596005717 Publisher: O'Reilly
Print Publication Date: 2003/10/01

14 Which class files must be included, either directly or by reference, in every ejb-jar file? (Choose all that apply.)

- ❑ A. The enterprise bean class.
- ❑ B. The stub class for the EJBObject interface.
- ❑ C. The enterprise bean's super classes.
- ❑ D. Any J2SE classes used as arguments or return types.

15 Which role is typically responsible for declaring the resource connection factory references in the deployment descriptor?

- ❑ A. bean provider
- ❑ B. application assembler
- ❑ C. deployer
- ❑ D. container provider
- ❑ E. system administrator

16 What's true about a legal ejb-jar file? (Choose all that apply.)

- ❑ A. It must contain both a home interface and a component interface.
- ❑ B. The deployment descriptor is optional.
- ❑ C. It must contain any J2EE classes used by the bean.
- ❑ D. The enterprise bean class is optional.

17 Which role would typically set up resource manager sign-on information?

- ❑ A. bean provider
- ❑ B. application assembler
- ❑ C. deployer
- ❑ D. container provider
- ❑ E. system administrator

Head First EJB™ By Bert Bates, Kathy Sierra ISBN: 0596005717 Publisher: O'Reilly
Print Publication Date: 2003/10/01

mock exam answers

Mock Exam Answers

1 Which APIs are guaranteed to be supported by EJB 2.0 containers? (Choose all that apply.) (spec: 493)

- [x] A. JAXP
- [x] B. JNDI
- [ ] C. JXTA
- [x] D. JDBC
- [x] E. JMS

2 What's true about an enterprise bean's environment? (Choose all that apply.) (spec: 410–412)

- [ ] A. Environment entries can be unique for instances of the same enterprise bean type.
- [x] B. Within a single EJB 2.0 container, an EJB can have multiple sets of environment entries.
- [ ] C. An EJB's environment entry's values can be modified by the EJB at runtime.
- [ ] D. Environment entry values may be primitives or wrapper types. – Only Strings and Wrappers

3 Which APIs are guaranteed to be supported by EJB 2.0 containers? (Choose all that apply.) (spec: 496–497)

- [x] A. J2SE 1.3
- [ ] B. JAXB 1.0
- [ ] C. JAXR 1.0
- [x] D. JAXP 1.0

Head First EJB™ By Bert Bates, Kathy Sierra ISBN: 0596005717 Publisher: O'Reilly
Print Publication Date: 2003/10/01

4 Given a bean named 'Customer', and an environment entry named 'lastName', which code fragment(s) inside of the bean class would return the value of the environment entry? (Choose all that apply.) (spec: 411–412)

❑ A.
```
Context c = new SessionContext();
Context e = (Context) c.lookup("java:comp/env");
String name = (String) e.lookup("lastName");
```
– SessionContext is not the same as JNDI Context

❑ B.
```
Context c = new InitialContext();
Context e = (Context) c.lookup("java:comp/env/
Customer");
String name = (String) e.lookup("lastName");
```

❑ C.
```
Context e = new Lookup("java:comp/env");
Context c = new InitialContext(e);
String name = (String) c.lookup("lastName");
```

❑ D.
```
Context c = new InitialContext("Customer");
Context e = (Context) c.lookup("java:comp/env");
String name = (String) e.lookup("lastName");
```
– no argument here

☑ E.
```
Context c = new InitialContext();
Context e = (Context) c.lookup("java:comp/env");
String name = (String) e.lookup("lastName");
```

5 When programming a session bean class which technique(s) should always be avoided to ensure bean portability across all EJB 2.0 containers? (Choose all that apply.) (spec: 494–495)

❑ A. Using the **`java.net.Socket`** class.

❑ B. Using inner classes.

❑ C. Using the 'final' modifier for fields.

☑ D. Passing 'this' as an argument.

– client Sockets are OK, just not a ServerSocket

Head First EJB™ By Bert Bates, Kathy Sierra ISBN: 0596005717 Publisher: O'Reilly
Print Publication Date: 2003/10/01

mock exam answers

6 Which of the following are valid data types in a **<env-entry-type>** element in a bean's deployment descriptor? (Choose all that apply.) (spec: 413)

- ❑ A. **byte**
- ❑ B. **short**
- ❑ C. **ArrayList**
- ☑ D. **java.lang.Boolean**
- ☑ E. **java.lang.Character**

– only Wrappers and Strings are supported

7 When programming EJBs which declaration(s) should be avoided to ensure bean portability across all EJB 2.0 containers? (Choose all that apply.) (spec: 494)

- ❑ A. **final int x;**
- ☑ B. **static int x;** – statics should also be final
- ❑ C. **final static int x;**
- ❑ D. **final transient int x;**

8 Who is typically responsible for specifying finder and select queries in the bean's deployment descriptor? (spec: 456–457)

- ☑ A. The bean provider.
- ❑ B. The application assembler.
- ❑ C. The deployer.
- ❑ D. The system administrator.
- ❑ E. The server provider.

9 Which deployment descriptor element(s) would be used when obtaining a JDBC connection? (Choose all that apply.) (spec: 424)

- ❑ A. **<ejb-ref>**
- ❑ B. **<ejb-link>**
- ❑ C. **<role-name>**
- ❑ D. **<env-entry>**
- ☑ E. **<resource-ref>**

10 Who will typically merge multiple ejb-jar files into a single ejb-jar file. (spec: 458)

- ❑ A. The bean provider.
- ☑ B. The application assembler.
- ❑ C. The deployer.
- ❑ D. The system administrator.
- ❑ E. The server provider.

11 Which deployment descriptor element(s) would be used by a bean provider to locate the home interfaces of other EJBs? (Choose all that apply.) (spec: 416)

- ☑ A. **<ejb-ref>**
- ❑ B. **<res-type>**
- ❑ C. **<env-entry>**
- ❑ D. **<role-name>**
- ❑ E. **<resource-ref>**

12 Which are bean provider responsibilities concerning resource manager connection factory references? (Choose all that apply.) (spec: 421)

- ❑ A. Configure resource managers in the EJB server.
- ❑ B. Configure sign-on information for the resource manager.
- ☑ C. Assign such a reference to the deployment descriptor.
- ❑ D. Creating a symbolic link to JNDI.

13 The ejb-jar file is considered to be part of the contract between which pairs? (Choose all that apply.) (spec: 487)

- ❑ A. bean provider and system administrator
- ☑ B. bean provider and application assembler
- ☑ C. application assembler and deployer
- ❑ D. application assembler and system administrator
- ❑ E. deployer and system administrator

mock exam answers

14 Which class files must be included, either directly or by reference, in every ejb-jar file? (Choose all that apply.) (spec: 488)

- [x] A. The enterprise bean class.
- [ ] B. The stub class for the EJBObject interface.
- [x] C. The enterprise bean's super classes.
- [ ] D. Any J2SE classes used as arguments or return types. – It'll already be there, baby!

15 Which role is typically responsible for declaring the resource connection factory references in the deployment descriptor? (spec: 423)

- [x] A. bean provider
- [ ] B. application assembler
- [ ] C. deployer
- [ ] D. container provider
- [ ] E. system administrator

16 What's true about a legal ejb-jar file? (Choose all that apply.) (spec: 488)

- [x] A. It must contain both a home interface and a component interface.
- [ ] B. The deployment descriptor is optional.
- [ ] C. It must contain any J2EE classes used by the bean.
- [ ] D. The enterprise bean class is optional.

17 Which role would typically set up resource manager sign-on information? (spec: 422)

- [ ] A. bean provider
- [ ] B. application assembler
- [x] C. deployer
- [ ] D. container provider
- [ ] E. system administrator