Table of Contents (summary)

	Intro	xix
1	Welcome to EJB: *an introduction*	1
2	EJB Architecture: *architectural overview*	61
3	Exposing Yourself: *the client view*	111
4	Being a Session Bean: *session bean lifecycle*	173
5	Entities are Persistent: *entity bean intro*	259
6	Being an Entity Bean: *bean/entity synchronization*	295
7	When Beans Relate: *entity bean relationships*	373
8	Getting the Message: *message-driven beans*	437
9	The Atomic Age: *EJB transactions*	469
10	When Beans Go Bad: *exceptions in EJB*	525
11	Protect Your Secrets: *security in EJB*	569
12	The Joy of Deployment: *a bean's environment*	599
A	Appendix A: *Final Mock Exam*	637

Table of Contents (the real thing)

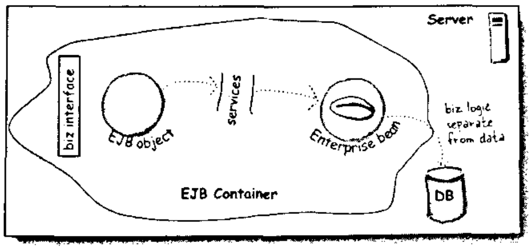
i Intro

Your brain on EJB. Here *you* are trying to *learn* something, while here your *brain* is doing you a favor by making sure the learning doesn't *stick*. Your brain's thinking, "Better leave room for more important things, like which wild animals to avoid and whether naked snowboarding is a bad idea." So how *do* you trick your brain into thinking that your life depends on knowing EJB?

Who is this book for?	xviii
We know what your brain is thinking	xix
Metacognition	xxi
Bend your brain into submission	xxiii
What you need for this book	xxiv
Passing the certification exam	xxvi
Technical reviewers	xxviii
Acknowledgements	xxix

1 Welcome to EJB

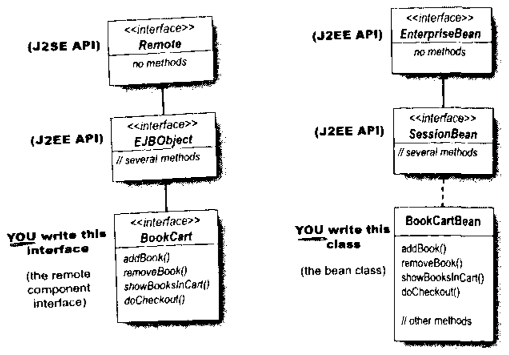
Enterprise JavaBeans are easy. Well, at least when you compare EJB to what you'd have to do to write your own scalable, transactional, secure, persistent, concurrent enterprise component server. In this chapter, we'll develop, deploy, and run an EJB application, and then dive into the details. Before we're done, we'll look at the use, benefits, and characteristics of EJB, and we'll look (briefly) at how EJB containers work.



Exam objectives 2
What is EJB all about? 3
No more vendor lock-in! 5
How does it all work? 7
Behind the scenes... 8
Beans come in three flavors 11
The Advice Guy bean 15
Five things you do to build a bean 16
EJB Roles and Responsibilities 26
Tutorial 28
Coffee Cram 59

2 EJB Architecture

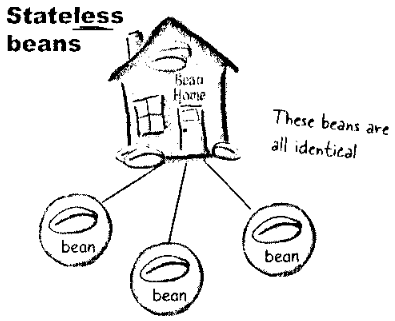
EJB is about infrastructure. Your components are the building blocks. With EJB, you can build big applications. The kind of applications that could run *everything* from the Victoria's Secret back-end to document-handling systems at CERN. But an architecture with this much flexibility, power, and scalability isn't simple. It all begins with a distributed programming model...



Exam objectives 62
Making a Remote method call 64
What about arguments and return values? 67
The client calls business methods through the Remote interface 79
EJB uses RMI 81
The Remote object is not the bean, it's the bean's bodyguard 82
Architectural overview: Session beans 98
Architectural overview: Entity beans 99
Architectural overview: Creating a Stateful Session bean 100
Architectural overview: Creating a Stateless Session bean 101
Architectural overview: Message-driven beans 106
Organize your beans 108

3 Exposing Yourself

You can't keep your bean private. Clients need to see what you've got. (Except for message-driven beans, which don't *have* a client view). The Advice Bean exposes the getAdvice() method in its Component interface—the place where you declare business methods. But that's not *all* the client sees. Remember, the Advice interface extends EJBObject, an interface with methods of its *own*. Methods the client can see. Methods the client can *call*. And it works the same way with the Home interface.

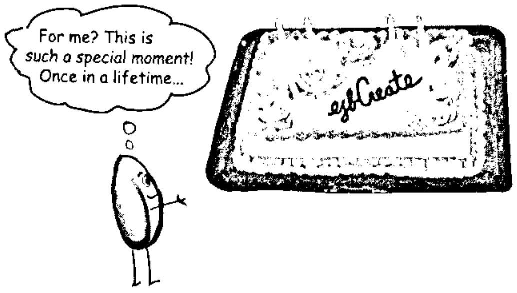


For stateless session beans from the same home, isIdentical() always returns *true*, even for different beans.

Exam objectives	112
What the client really wants	113
What's JNDI?	116
PortableRemoteObject.narrow() (exotic casting)	121
Writing the Remote home interface for a session bean	125
Thankfully, we've got handles (online shopping takes time)	139
Which methods make sense for the local client interfaces?	148
Why so many remove methods?	151
Comparing Remote vs. Local interfaces	154
Arguments to Remote vs. local methods	163
Coffee Cram	168

4 Being a Session Bean

Session beans are created and removed. If you're lucky, you're a state*less* bean. Because the life of a state*ful* bean is tied to the whims of a heartless client. Stateful beans are created at the client's insistence, and live and die *only* to serve that one client. But ahhhh, the life of a stateless bean is ***fabulous!*** Pools, those little umbrella drinks, and no boredom since you get to meet so many different clients.



For me? This is such a special moment! Once in a lifetime...
ejbCreate

Exam objectives	174
Container callbacks, for the special moments in a bean's life	181
Bean Creation	188
Bean things you can do within business methods	199
Passivation: a stateful bean's chance at scalability	200
Bean Removal	208
Writing a Session Bean: your job as Bean Provider	230
SessionContext: you need it more than it needs you	240
Coffee Cram	244

5 Entities are Persistent

Entity beans persist. Entity beans exist. Entity beans *are*. They are object representations of something in an ***underlying persistent store***. (Think: ***database,*** because most entity beans represent something from a relational database.) If you have a Customer entity bean, then one bean might represent the entity Tyler Durden, ID #343, while another is the entity Donny Darko, ID #42. Three beans, representing three real *entities*. An entity bean is simply a *realization* of something that already exists.

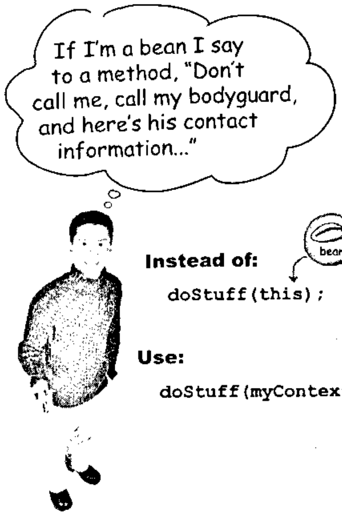
If you've got any last words, you better do it in your ejbRemove()...

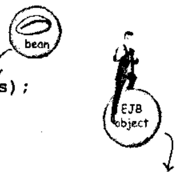
No, Please, No! I'll give you whatever you want, just don't call remove()!

Exam objectives	260
What's an entity bean?	261
Entity beans from the client's point of view	265
A very simple Customer entity bean	268
Entity bean Remote component interface	270
Entity bean Remote home interface	273
What does the client really want from an entity bean home?	274
Home business methods to the rescue	278
Session bean create() vs. entity bean create()	281
Session bean remove() vs. entity bean remove()	282
Entity/bean/instance death	285
Coffee Cram	288

6 Being an Entity Bean

Entity beans are actors. As long as they're alive, they're either in the pool or they're *being* somebody. Somebody from the underlying persistent store (an entity from the database). When a bean is playing a part, the bean and the underlying entity have to stay in sync. Imagine the horror if the bean is pretending to be, say, Audrey Leone, and someone lowers Audrey's credit limit in the database... *but forgets to tell the bean.*


If I'm a bean I say to a method, "Don't call me, call my bodyguard, and here's his contact information..."



Instead of:

```
doStuff(this);
```

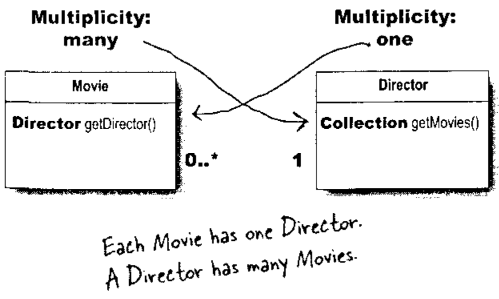
Use:

```
doStuff(myContext.getEJBObject());
```

Exam objectives	296
The real power of entity beans is synchronization	298
Container-managed vs. bean-managed persistence	303
The EntityBean interface adds new container callbacks	310
Writing a CMP entity bean	313
Object identity: the primary key	332
Finder methods	339
Home business methods	345
Coffee Cram	362

7 When Beans Relate

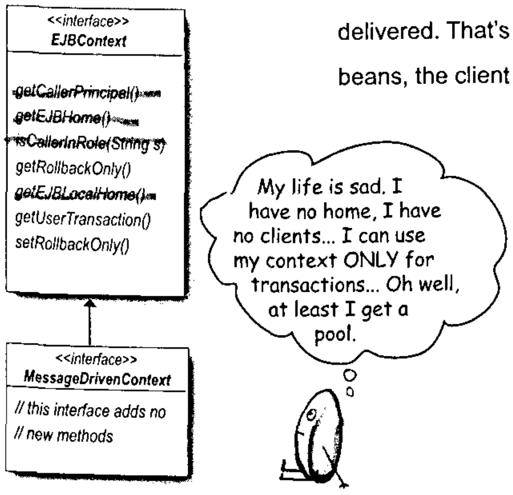
Entity beans need relationships. An Order needs a Customer. A LineItem needs an Order. An Order needs LineItems. Entity beans can have container-managed relationships (CMR) and the Container takes care of virtually *everything*. Make a new LineItem that's related to an Order? If you ask the Customer to show you his Orders, you'll see the new LineItem. Best of all, you can use EJB-QL to write *portable* queries.



Exam objectives	374
Relationships	378
Multiplicity	380
CMP and CMR fields	383
Cascade delete can propagate	393
EJB-QL for the MovieBean	402
SELECT and FROM are the mandatory!	409
The WHERE clause	411
Collections don't bark()!	414
The BETWEEN, IN, IS EMPTY, and LIKE expression	416
Relationship assignments	421
Coffee cram	425

8 Getting the Message

It's fun to receive messages. Not as much fun as, say, getting that eBay package with the genuine Smurf™ lamp, but fun and *efficient* nonetheless. Imagine if you sent your order to eBay, and you couldn't leave your house until the package was delivered. That's what it's like with Session and Entity beans. But with message-driven beans, the client can send a message and walk away.

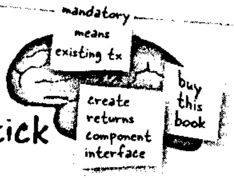

<<interface>> EJBContext
~~getCallerPrincipal()~~
~~getEJBHome()~~
~~isCallerInRole(String s)~~
getRollbackOnly()
~~getEJBLocalHome()~~
getUserTransaction()
setRollbackOnly()
<<interface>> MessageDrivenContext
// this interface adds no
// new methods
My life is sad. I have no home, I have no clients... I can use my context ONLY for transactions... Oh well, at least I get a pool.

Exam objectives	438
Writing a message-driven bean: your job as Bean Provider	447
Complete DD for a message-driven bean	449
Topics and Queues	450
MessageDrivenContext	455
Message acknowledgement	458
Coffee Cram	463

9 The Atomic Age

Transactions protect you. With transactions, you can try something knowing that if anything goes wrong along the way, you can just pretend the whole thing didn't happen. Everything goes back to the way it was *before*. Transactions in EJB are a thing of beauty—you can deploy a bean with customized transaction behavior *without* touching the bean's source code! But you *can* write transaction code, if you need to.

Make it Stick


mandatory means existing tx
create returns component interface
buy this book

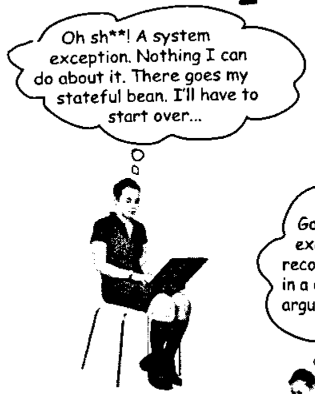
CMT beans run
transactions unknown,
while BMT beans
use only their own.

OK, not our best work, we know. So why don't ***you*** try it. Memory devices can help, but they work much better when you create them yourself.

Exam objectives	470
The ACID test	472
How it works in EJB	474
Transaction propagation	475
How do I make (or get) a transaction?	476
setRollbackOnly() lives in TWO interfaces	487
BMT can be a really BAD idea. BMT hurts bean reuse	490
Container-managed transactions	491
How attributes work	492
Methods you MUST mark with an attribute (for a CMT bean)	498
"Unspecified Transaction Context"	499
DD example for CMT	503
SessionSynchronization "special moments"	512
Coffee cram	516

10 When Beans Go Bad

Expect the unexpected. Despite your best efforts, things can go wrong. Terribly, *tragically*, wrong. You need to protect yourself. You can't let your entire program collapse, just because one bean in the family throws an exception. ***The application must go on.*** You can't *prevent* tragedy, but you can *prepare* for it. You need to know what is and is *not* recoverable, and *who* is responsible when a problem occurs.

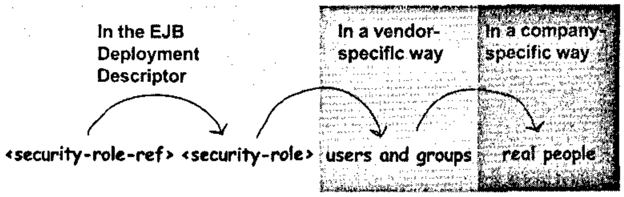

Oh sh**! A system exception. Nothing I can do about it. There goes my stateful bean. I'll have to start over...

Gotta love application exceptions... I can recover from this if I put in a different value for the argument to the create() method...

Exam objectives	526
In EJB, exceptions come in two flavors	532
With an Application Exception, the Container will...	533
With a System Exception, the Container will...	534
RemoteException vs. EJBException	539
Bean Provider's responsibilities	541
The five standard EJB application exceptions	548
Common system exceptions	551
Coffee cram	563

11 Protect Your Secrets

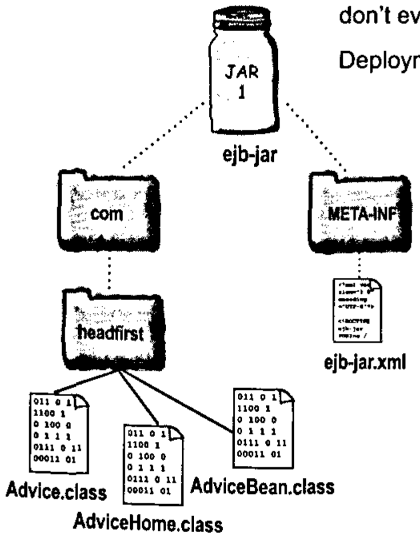
Keep your secrets. Security is about **authentication** and **authorization.** First, you have to prove your identity, and then we'll tell you what you're allowed to do. Security is easy in EJB, because you're only dealing with *authorization*. You decide *who* gets to call which *methods* on your beans. Except one problem... if you're a Bean Provider or App Assembler, you probably don't *know* who the users are going to be!



Exam objectives 570
How to do security in EJB 573
The Application Assembler's job: access control 574
Defining the method permissions 578
The Deployer's job: mapping actual humans to abstract roles 583
Class-level vs. instance-level security 586
Using programmatic security to custom-tailor a method 587
Use <run-as> security identity to pretend someone else is calling... 591
Security context propagation with <run-as> 592
Coffee cram 593

12 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. And what if you don't even *have* the source code? EJB supports bean reuse through the customizable Deployment Descriptor and a bean's special *environment.*



Exam objectives 600
A bean's special place- java:comp/env 602
Creating and using a subcontext 609
Bean Provider and Application Assembler responsibility 617
Deployer responsibility 618
Remembering who does what 619
Which APIs does EJB 2.0 guarantee? 621
What MUST be in an ejb-jar? 624
Programming restrictions 625
Coffee cram 627

A COFFEE CRAM

The final Coffee Cram Mock Exam. This is it. 70 questions. The tone, topics, and difficulty level are virtually identical to the *real* exam. *We know.*

Final Mock Exam 637

i **Index** 685