Lightweight POJO Frameworks in JBoss

Michael Yuan, PhD
JBoss, a division of RedHat

© JBoss Inc. 2006

What is "lightweight"?

Lightweight development model

Heavy duty and rich runtime services

Flexible deployment footprint

The lightweight runtime?

- Some consultants / authors claim
  - ✓ Tomcat + Hibernate + Spring is all you need
  - ✓ Small runtime footprint is key
- Then, why not use PHP? It is even more "lightweight"
  - ✓ Actually many of the same people have already switched to Ruby since it is "lighter" than Java
- In fact, most of their apps use full blown app servers to provide key services
  - ✓ There is nothing "lightweight" about the runtime for an enterprise application!
  - ✓ Flexibility in runtime footprint is key!!!

Our definition

A lightweight application framework makes extensive use of POJOs (Plain Old Java Objects) to assemble complex applications. It is a development model optimized for developer productivity and architectural flexibility.

About me

- 5 books on Java EE and ME
- 50+ articles in leading magazines
- Worked on many products inside JBoss
- Upcoming books from Prentice Hall
  - ✓ Lightweight Java web app development
  - ✓ JBoss Seam: Simplicity and Power Beyond Java EE 5

Lightweight Java Web Application Development

Leveraging EJB 3.0, JSF, POJO, and Seam

Michael Juntao Yuan
Norman Richards

PRENTICE HALL JBOSS SERIES

Agenda

- Lightweight Principals
- The JBoss MicroContainer
- JBoss AOP
- Hibernate
- EJB3
- Seam

Extensive use of POJOs

- Self-contained object
  - ✓ Takes care of its own business logic
  - ✓ Loose coupling via interfaces
- Dependency resolution is key
  - ✓ Not all POJOs have external dependency
  - ✓ If they do, it is resolved outside of the object
  - ✓ Dependency injection as opposed to dependency lookup
- Easy to work with
  - ✓ Little boilerplate code
  - ✓ Easy to unit test
  - ✓ Little constraint on external frameworks
  - ✓ Achieve good OO design

Reduce repetitive artifacts

- Framework required interfaces
  - ✓ Big complaint against EJB 1.x/2.x
  - ✓ Business interfaces are *good*
- The XML hell
  - ✓ Repeat Java code in XML
  - ✓ Verbose, hard to read and hard to understand
  - ✓ But it does separate POJO code from external context
- Generate as much stuff as possible
  - ✓ Proxies, configurations, procedural code (i.e., SQL)

Do not Repeat Yourself (DRY)

Two types of POJOs

- Business objects
  - ✓ Objects with external dependency
  - ✓ Can use external services or provide services to other POJOs
  - ✓ Dependency injection is the key here
- Persistence objects
  - ✓ Portable OO model for SQL
  - ✓ No external dependency
  - ✓ Managed in a persistence context

What about the UI?

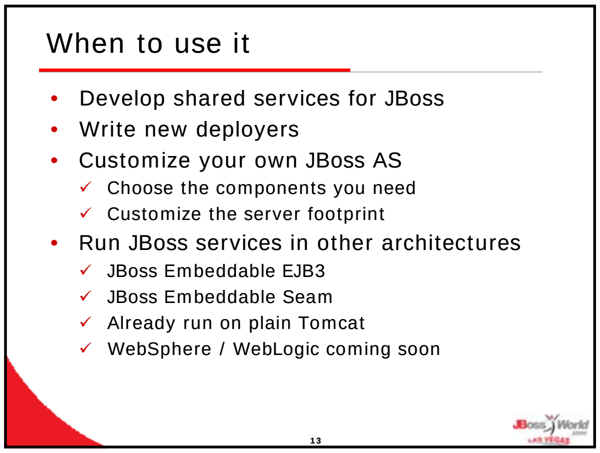
- JavaServer Faces
  - ✓ Componentized UI for generating HTML / JavaScript / CSS
  - ✓ Integration with visual designers
- Facelet
  - ✓ Template engine for JSF
  - ✓ Dependency injection for the UI

Agenda

- Lightweight Principals
- The JBoss MicroContainer
- JBoss AOP
- Hibernate
- EJB3
- Seam

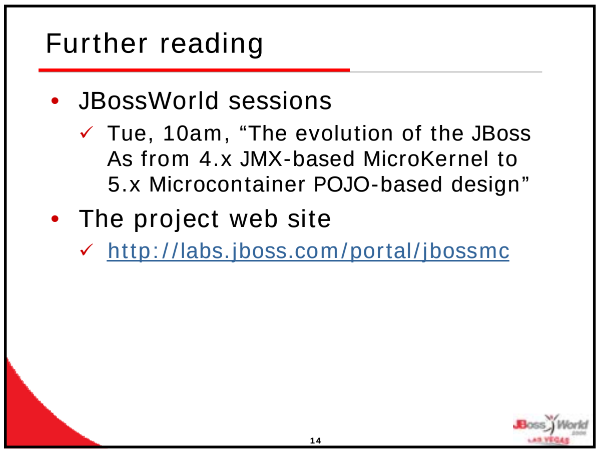
JBoss MicroContainer

- A generic dependency injection container
  - ✓ Use XML to wire together objects
  - ✓ Works in any Java SE environment
  - ✓ Drop-in app deployment in JBoss (.beans packages)
  - ✓ The DI part is similar to other XML-based DI frameworks out there
- Core of JBoss AS 5.0+
  - ✓ Lifecycle callback hooks for POJOs
  - ✓ Deployer support
  - ✓ JBoss AOP integration
  - ✓ All features in the current JMX microkernel plus more management features



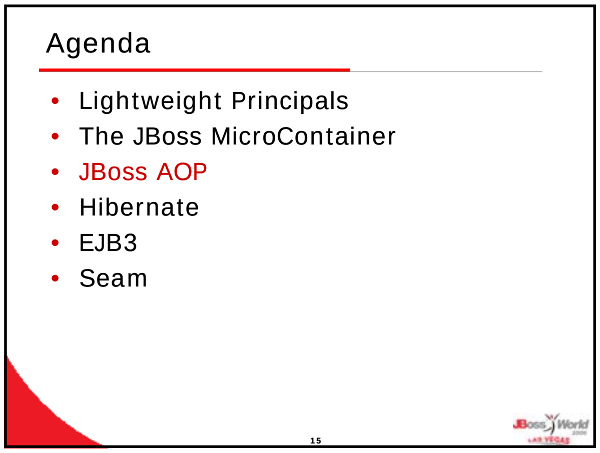
When to use it

- Develop shared services for JBoss
- Write new deployers
- Customize your own JBoss AS
  - ✓ Choose the components you need
  - ✓ Customize the server footprint
- Run JBoss services in other architectures
  - ✓ JBoss Embeddable EJB3
  - ✓ JBoss Embeddable Seam
  - ✓ Already run on plain Tomcat
  - ✓ WebSphere / WebLogic coming soon



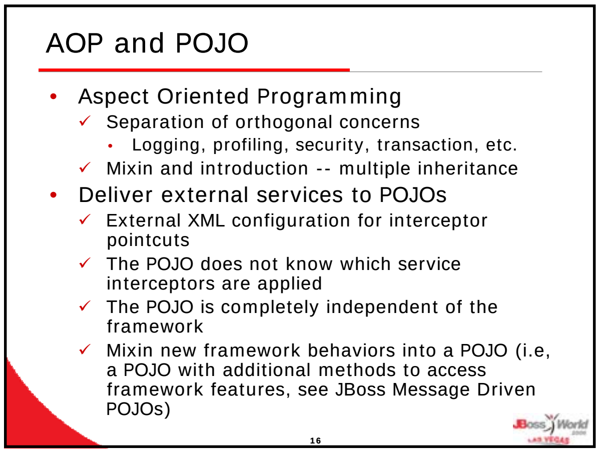
Further reading

- JBossWorld sessions
  - ✓ Tue, 10am, “The evolution of the JBoss As from 4.x JMX-based MicroKernel to 5.x Microcontainer POJO-based design”
- The project web site
  - ✓ http://labs.jboss.com/portal/jbossmc



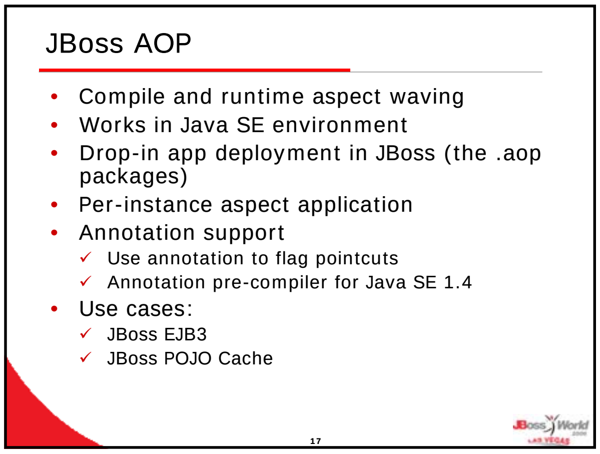
Agenda

- Lightweight Principals
- The JBoss MicroContainer
- JBoss AOP
- Hibernate
- EJB3
- Seam



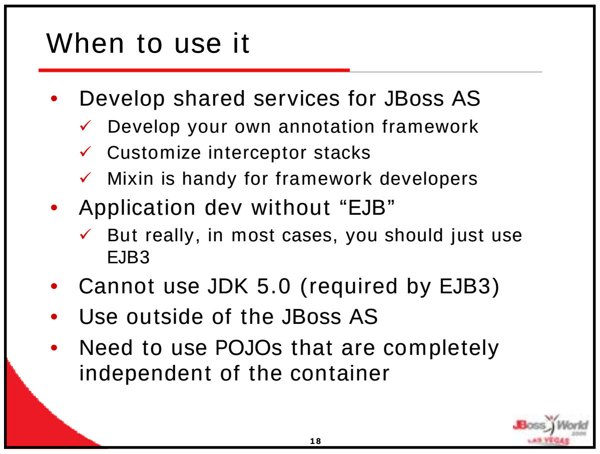
AOP and POJO

- Aspect Oriented Programming
  - ✓ Separation of orthogonal concerns
    - Logging, profiling, security, transaction, etc.
  - ✓ Mixin and introduction -- multiple inheritance
- Deliver external services to POJOs
  - ✓ External XML configuration for interceptor pointcuts
  - ✓ The POJO does not know which service interceptors are applied
  - ✓ The POJO is completely independent of the framework
  - ✓ Mixin new framework behaviors into a POJO (i.e, a POJO with additional methods to access framework features, see JBoss Message Driven POJOs)



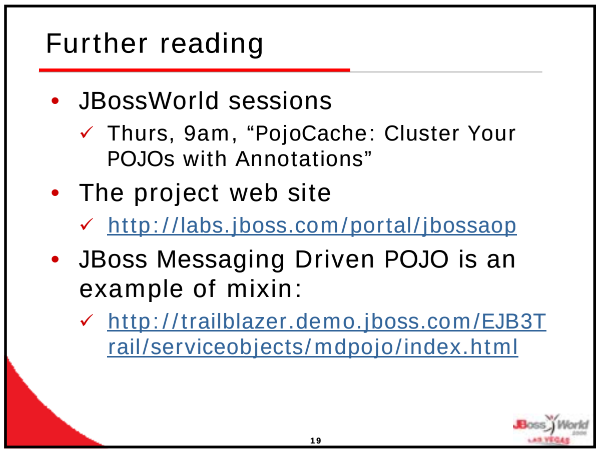
JBoss AOP

- Compile and runtime aspect waving
- Works in Java SE environment
- Drop-in app deployment in JBoss (the .aop packages)
- Per-instance aspect application
- Annotation support
  - ✓ Use annotation to flag pointcuts
  - ✓ Annotation pre-compiler for Java SE 1.4
- Use cases:
  - ✓ JBoss EJB3
  - ✓ JBoss POJO Cache



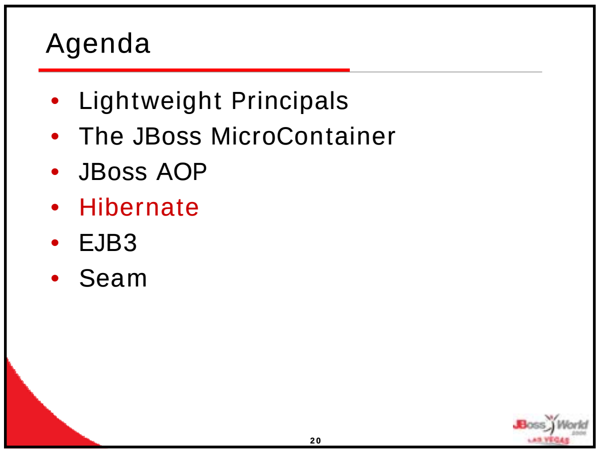
When to use it

- Develop shared services for JBoss AS
  - ✓ Develop your own annotation framework
  - ✓ Customize interceptor stacks
  - ✓ Mixin is handy for framework developers
- Application dev without “EJB”
  - ✓ But really, in most cases, you should just use EJB3
- Cannot use JDK 5.0 (required by EJB3)
- Use outside of the JBoss AS
- Need to use POJOs that are completely independent of the container



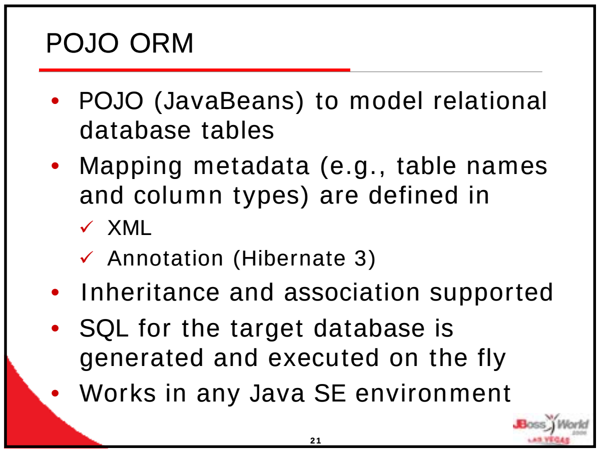
Further reading

- JBossWorld sessions
  - ✓ Thurs, 9am, “PojoCache: Cluster Your POJOs with Annotations”
- The project web site
  - ✓ http://labs.jboss.com/portal/jbossaop
- JBoss Messaging Driven POJO is an example of mixin:
  - ✓ http://trailblazer.demo.jboss.com/EJB3Trail/serviceobjects/mdpojo/index.html



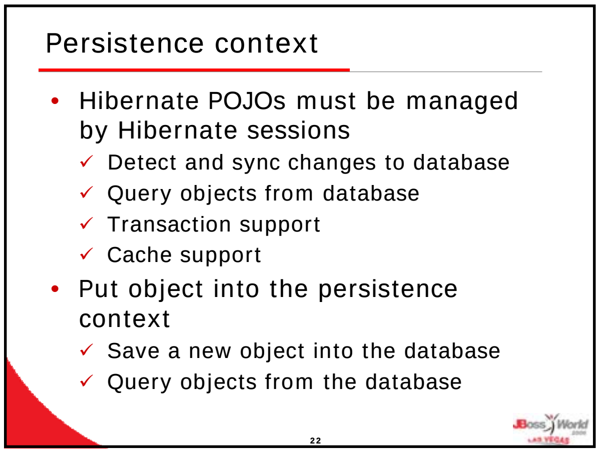
Agenda

- Lightweight Principals
- The JBoss MicroContainer
- JBoss AOP
- Hibernate
- EJB3
- Seam



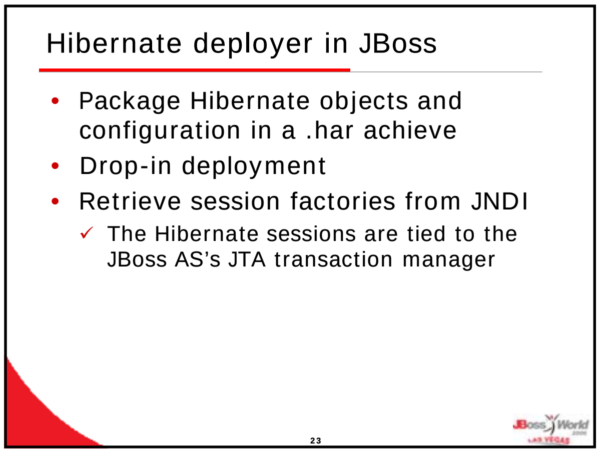
POJO ORM

- POJO (JavaBeans) to model relational database tables
- Mapping metadata (e.g., table names and column types) are defined in
  - ✓ XML
  - ✓ Annotation (Hibernate 3)
- Inheritance and association supported
- SQL for the target database is generated and executed on the fly
- Works in any Java SE environment



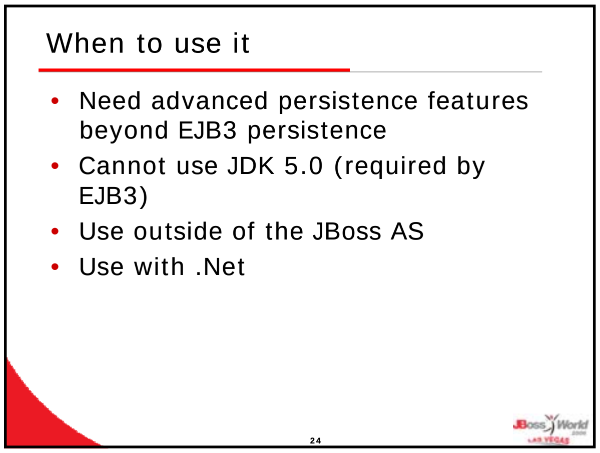
Persistence context

- Hibernate POJOs must be managed by Hibernate sessions
  - ✓ Detect and sync changes to database
  - ✓ Query objects from database
  - ✓ Transaction support
  - ✓ Cache support
- Put object into the persistence context
  - ✓ Save a new object into the database
  - ✓ Query objects from the database



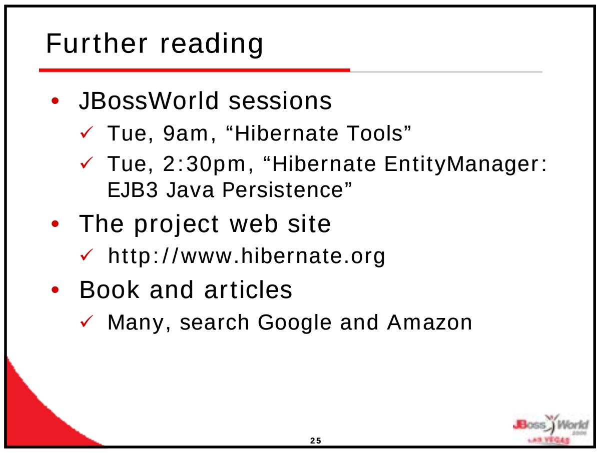
Hibernate deployer in JBoss

- Package Hibernate objects and configuration in a .har achieve
- Drop-in deployment
- Retrieve session factories from JNDI
  - ✓ The Hibernate sessions are tied to the JBoss AS’s JTA transaction manager



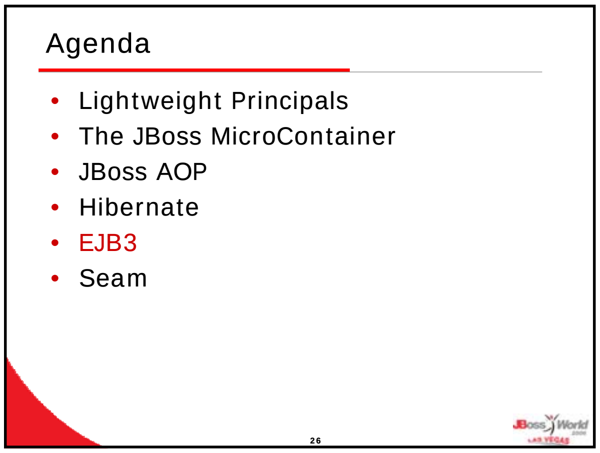
When to use it

- Need advanced persistence features beyond EJB3 persistence
- Cannot use JDK 5.0 (required by EJB3)
- Use outside of the JBoss AS
- Use with .Net



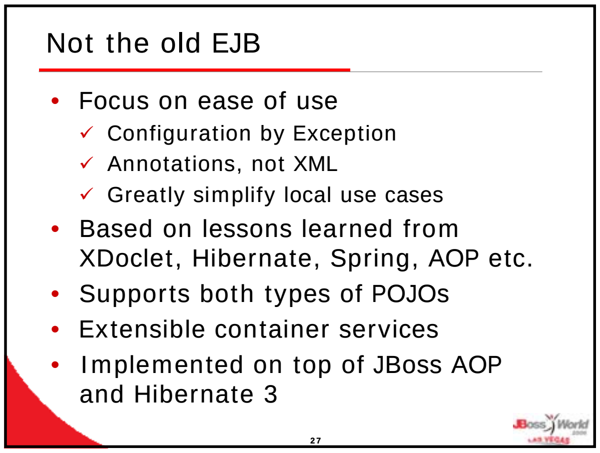
Further reading

- JBossWorld sessions
  - ✓ Tue, 9am, “Hibernate Tools”
  - ✓ Tue, 2:30pm, “Hibernate EntityManager: EJB3 Java Persistence”
- The project web site
  - ✓ http://www.hibernate.org
- Book and articles
  - ✓ Many, search Google and Amazon



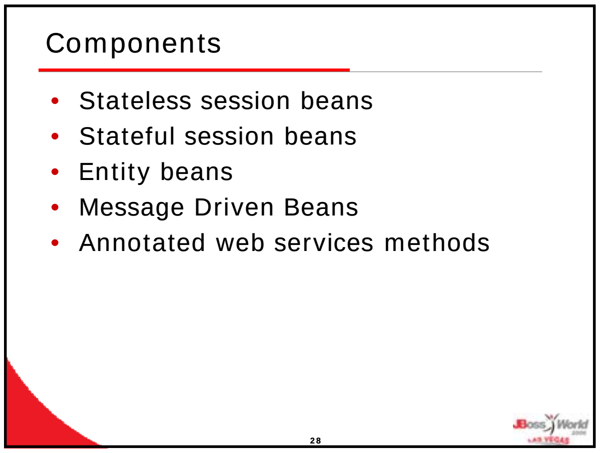
Agenda

- Lightweight Principals
- The JBoss MicroContainer
- JBoss AOP
- Hibernate
- EJB3
- Seam



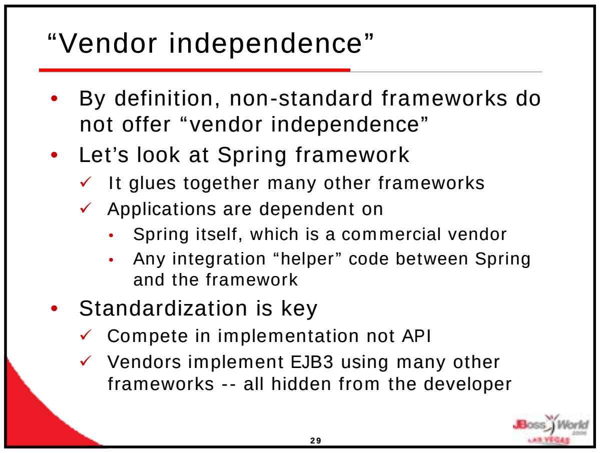
Not the old EJB

- Focus on ease of use
  - ✓ Configuration by Exception
  - ✓ Annotations, not XML
  - ✓ Greatly simplify local use cases
- Based on lessons learned from XDoclet, Hibernate, Spring, AOP etc.
- Supports both types of POJOs
- Extensible container services
- Implemented on top of JBoss AOP and Hibernate 3



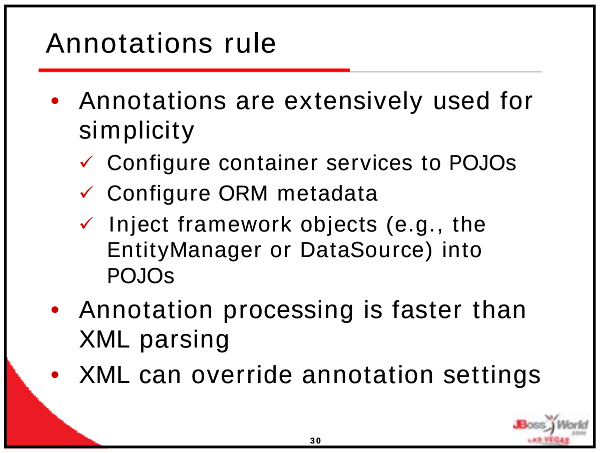
Components

- Stateless session beans
- Stateful session beans
- Entity beans
- Message Driven Beans
- Annotated web services methods



“Vendor independence”

- By definition, non-standard frameworks do not offer “vendor independence”
- Let’s look at Spring framework
  - ✓ It glues together many other frameworks
  - ✓ Applications are dependent on
    - Spring itself, which is a commercial vendor
    - Any integration “helper” code between Spring and the framework
- Standardization is key
  - ✓ Compete in implementation not API
  - ✓ Vendors implement EJB3 using many other frameworks -- all hidden from the developer



Annotations rule

- Annotations are extensively used for simplicity
  - ✓ Configure container services to POJOs
  - ✓ Configure ORM metadata
  - ✓ Inject framework objects (e.g., the EntityManager or DataSource) into POJOs
- Annotation processing is faster than XML parsing
- XML can override annotation settings

Interceptors

- Use a POJO method as interceptor
- Apply interceptors via
  - ✓ The @interceptors annotation
  - ✓ Custom service config annotations
  - ✓ XML configuration file
- Almost everything AOP interceptors can do …

Testing

- Just create EJB3 beans using "new" and run any unit test
- Integration tests can be done outside of JBoss AS using JBoss Embeddable EJB3
  - ✓ Test in plain Tomcat
  - ✓ Test in Java SE

When to use it

We recommend using EJB3 in most new development projects.

Further reading

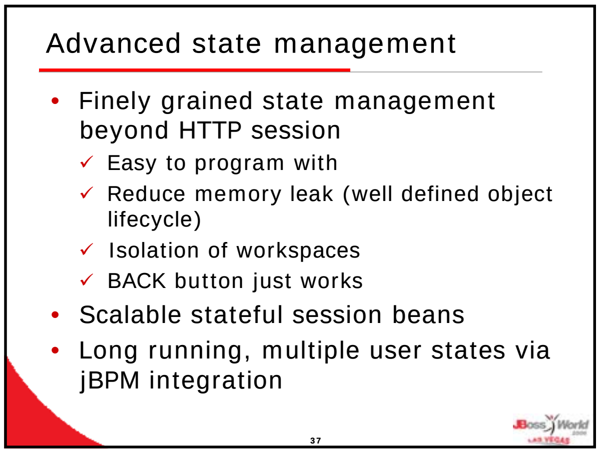
- JBossWorld sessions
  - ✓ Tue, 9am, "EJB 3.0"
  - ✓ Tue, 2:30pm, "Hibernate EntityManager: EJB3 Java Persistence"
  - ✓ Wed, 2:20pm, "Java EE 5"
  - ✓ Wed, 3:20PM, "EJB3/Seam performance and scalability on Dell PowerEdge 1855"
  - ✓ Web, 4:30pm, "Merging EJB3 and Spring Frameworks"
- The project web site
  - ✓ http://labs.jboss.com/portal/jbossejb3
- Trailblazers and online demos
  - ✓ http://trailblazer.demo.jboss.com/EJB3Trail/
- Books
  - ✓ "Enterprise Java Beans 3.0" by O'Reilly
  - ✓ "Lightweight web application development", by Prentice Hall

Agenda

- Lightweight Principals
- The JBoss MicroContainer
- JBoss AOP
- Hibernate
- EJB3
- Seam

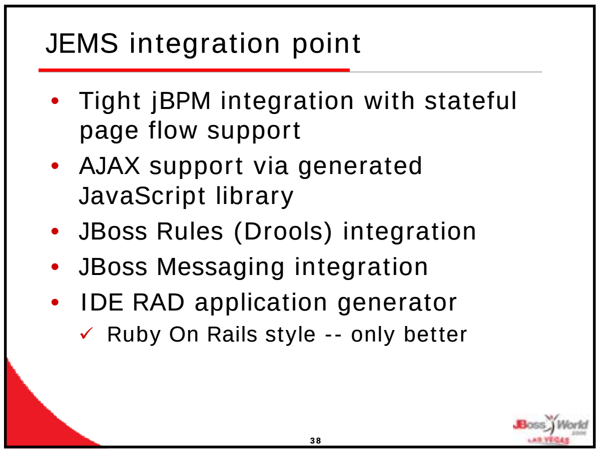
Pervasive annotations

- Eliminates JSF backing beans
- All components are "one kind of stuff" tied together by stateful contexts
  - ✓ Use annotation to declare names in the context
  - ✓ Dependency bi-jection
- Integration from model to UI
  - ✓ End-to-end validation
  - ✓ Give model objects UI behaviors
- Tie server states with user actions
  - ✓ Declare begin/end of web conversations and business processes



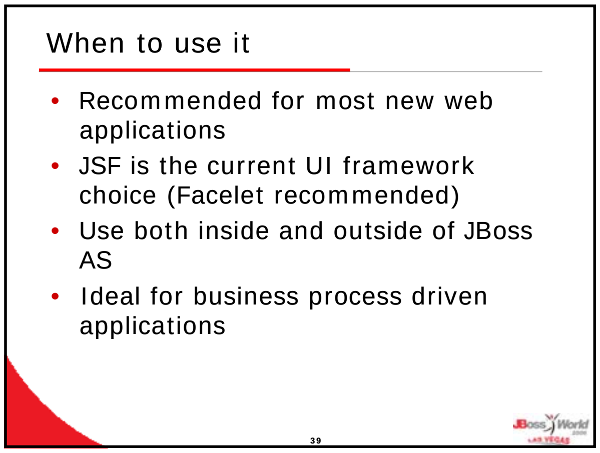
Advanced state management

- Finely grained state management beyond HTTP session
  - ✓ Easy to program with
  - ✓ Reduce memory leak (well defined object lifecycle)
  - ✓ Isolation of workspaces
  - ✓ BACK button just works
- Scalable stateful session beans
- Long running, multiple user states via jBPM integration



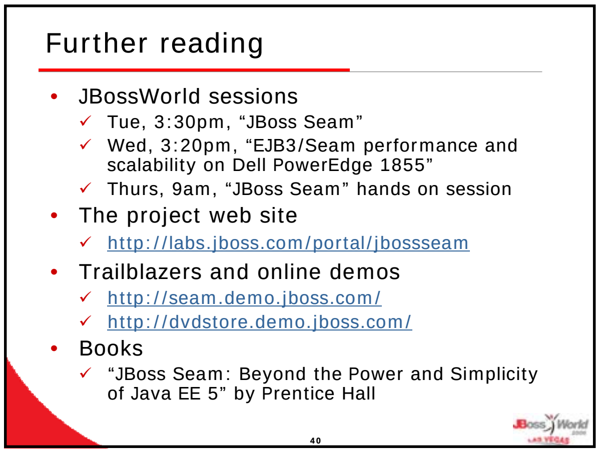
JEMS integration point

- Tight jBPM integration with stateful page flow support
- AJAX support via generated JavaScript library
- JBoss Rules (Drools) integration
- JBoss Messaging integration
- IDE RAD application generator
  - ✓ Ruby On Rails style -- only better



When to use it

- Recommended for most new web applications
- JSF is the current UI framework choice (Facelet recommended)
- Use both inside and outside of JBoss AS
- Ideal for business process driven applications



Further reading

- JBossWorld sessions
  - ✓ Tue, 3:30pm, “JBoss Seam”
  - ✓ Wed, 3:20pm, “EJB3/Seam performance and scalability on Dell PowerEdge 1855”
  - ✓ Thurs, 9am, “JBoss Seam” hands on session
- The project web site
  - ✓ http://labs.jboss.com/portal/jbossseam
- Trailblazers and online demos
  - ✓ http://seam.demo.jboss.com/
  - ✓ http://dvdstore.demo.jboss.com/
- Books
  - ✓ “JBoss Seam: Beyond the Power and Simplicity of Java EE 5” by Prentice Hall