Enterprise Java Beans, EJB

JPA Slides #3

Developing a Persistence Layer

Mapping of Entity classes is the first step but it's not sufficient

- The entity classes aren't persisted by themselves (or some automagically mechanism)
- Nor removed (at garbage collection)
- Have to handle in application
- Complex; concurrency, transactions, …

Transactions

Transactions fundamental for database integrity

- Basic flow
    1. Begin the transaction
    2. Execute a set of data manipulations and/or queries
    3. If no errors occur then commit the transaction and end it
    4. If errors occur then rollback the transaction and end it

If program executes 2) then must handle the others

- Transaction demarcation
- Possible with statements (tx.begin(), tx.commit()) but better use…
- **Enterprise JavaBeans, EJB**
    - Java bean with some EJB annotations
    - As noted: We need to run in GlassFish (EJB Container)

EJB and EJB Containers

Services Provided by EJB/Container

- Life cycle of EJBs
- Dependency injection (inferior to CDI)
- Concurrency handling

  Therefore, a stateful or stateless session bean does <u>not have to be coded as reentrant</u> [~thread safe]..." more to come.
- <u>Automatic transaction demarcation</u> (and more)
- Aspect oriented programming, interceptors
- Security
- Scheduling
- Web services endpoint (EJB may act as resource class)
- ...

Types and Purpose of EJBs

Session Beans (SB)
- Business logic, processes, work flows
- Persistence layer facade (Data Access Objects, DAO)

Message Driven Beans (MDB)
- Used by Java Message Service (JMS). Asynchronous message service with high quality of service
- Not covered...

Session Beans

Characteristics

- Executes on behalf of a unique client, client running the session (i. e session scoped)
- Short lived, will not survive a server crash
- Stateless or …
- … Stateful
- Singleton: Singleton pattern (<u>NOT</u> same as CDI Singleton)
- <u>Transaction-aware, transactions inserted as needed, handled by container</u>
- Possible implement interface
- Possible asynchronous
- Variations: Local bean (in same JVM), Remote bean (possible other JVM)

Requirements Session Beans

Must obey

- Concrete class (not abstract)
- Not final
- Marked @Stateless or @Stateful or @Singleton, more to come…
- No arg constructor

Optionally

- Inheritance possible (involving EJBs only)
- Possible have an interface marked @Remote, @Local

Stateless SB

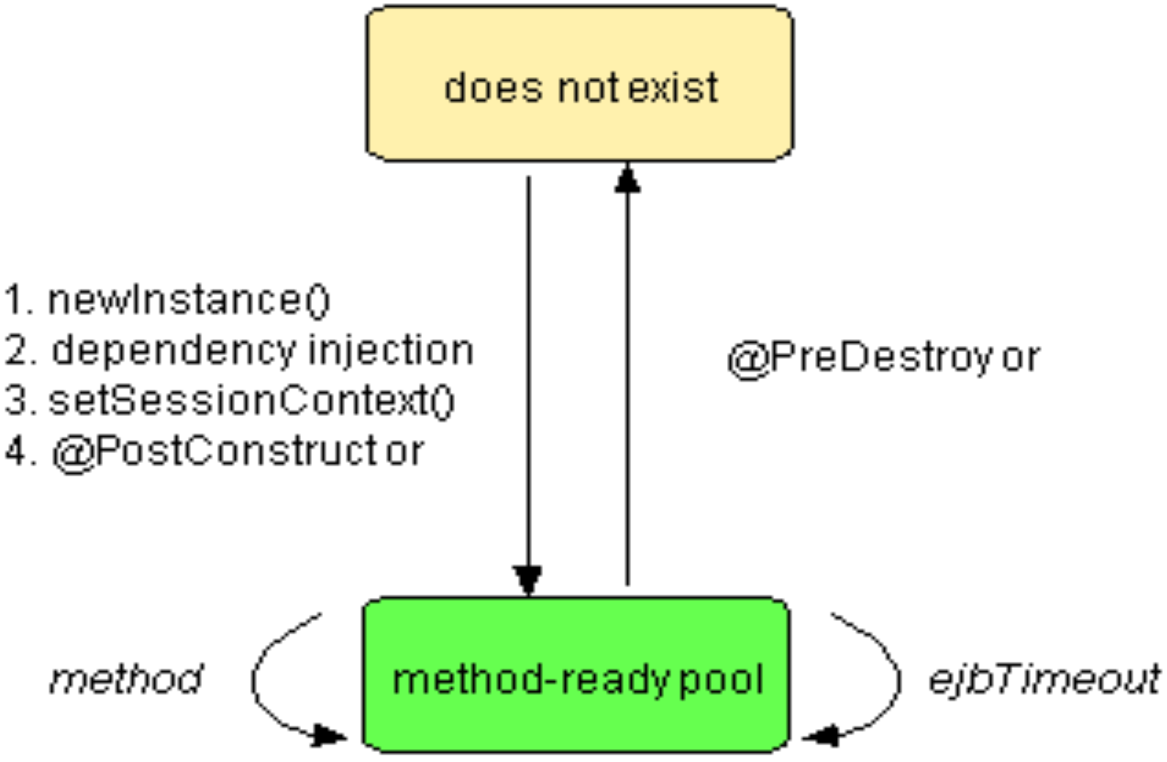
<u>No conversational state</u> (no instance variables)

- Task possible to conclude in single method call
- Efficient, pooled by container, share by many clients
- May implement a Web Service

The preferred type to use

- We always strive for <u>statelessness</u>

Stateless SB Life Cycle



Annotations for life cycle callback methods
@PostConstruct
@PreDestroy

Stateful SB

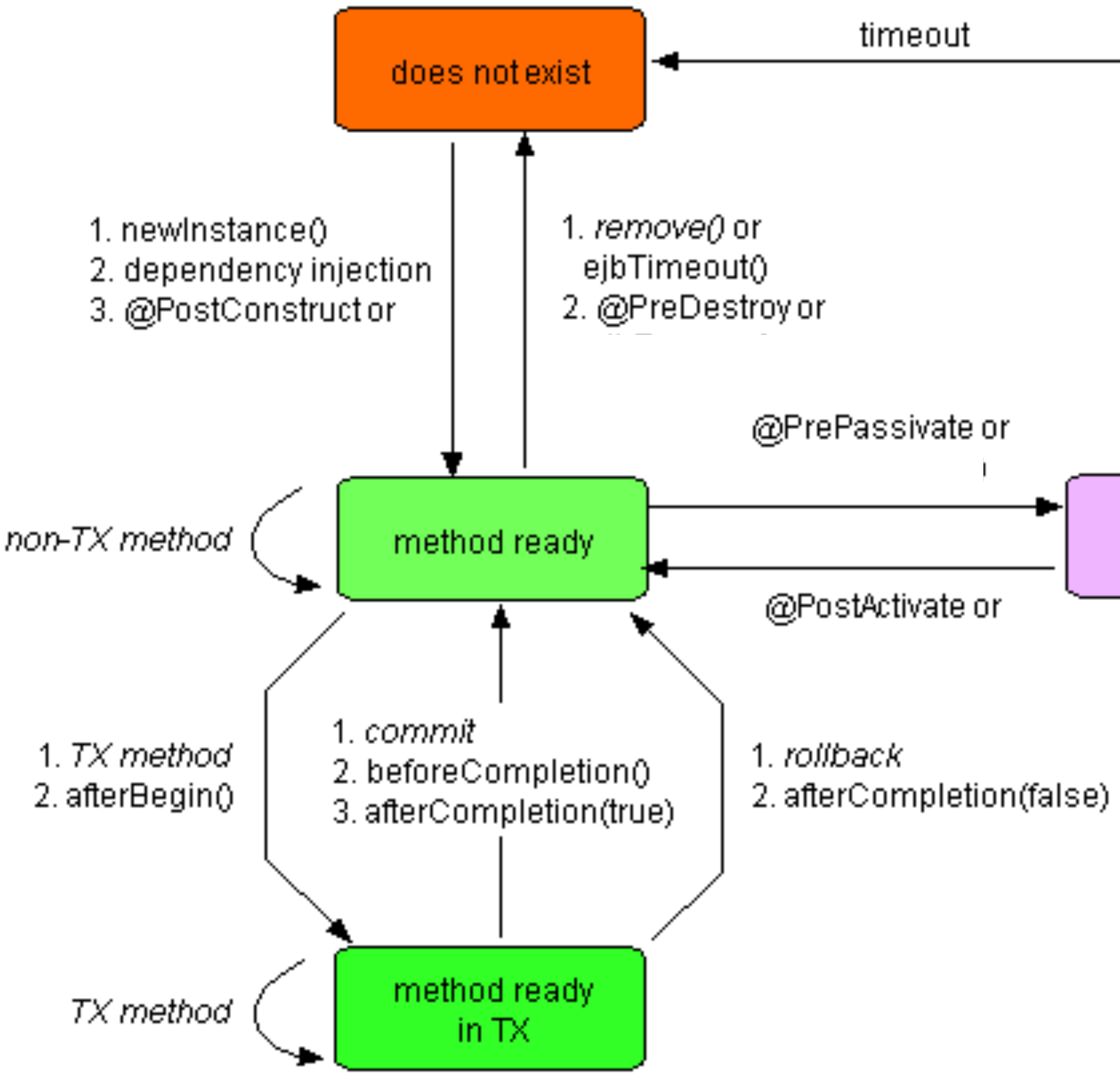
Conversational state (instance variables)

- Task done in several steps
- ShoppingCart, add items...
- Container can't share (less efficient compared to stateless)
- State retained for the session
- Passivated/Activated (moved out/in memory)

Complex

- Conversational state <u>not transactional</u> (rollbacks could lead to inconsistency)
- Have to handle in code
- <u>Normally avoid</u>

Stateful SB LifeCycle



Annotations for life cycle callback methods
@PostConstruct
@PreDestroy
@PrePassivate
@PostActivate

Session Beans and Interfaces

Possible to use Session Beans without interface

- No-interface view
- Possible for bean to implement a local interface with @Local annotation …
- … or a remote interface with @Remote annotation
- Possible implement both

Term: **“Business interface”**, methods visible (useful) to client (other bean, ...)

Local Interface

Same as no interface, but we should of course program to interfaces for SE reasons, i.e. use local interface! (not always in sample code)

- Client end SB in same JVM
- Call by reference
- Inexpensive

Remote Interface

Possible to make remote calls directly between Java objects

- Using **CORBA/IIOP** or **RMI**
- Remote beans must implement a remote interface

Client end SB in different JVMs

- Possible different servers
- Call by value
- Parameters/return values must be serializable
- Calls potentially expensive, network...
- Container must support CORBA/IIOP and Java RMI

EJB Singleton

Instantiated once per application

- Possible use : "the database", "printer",...
- Shared by many clients
- Maintain state between client invocations
- Concurrency variations: Container Managed (@Lock), bean Managed (@ConcurrencyManagement)...or not allowed (exception if...)

Initialization (@Startup)

- Dependencies between Singleton → initialization order matters (@DependsOn)

NOT SAME AS CDI Singleton

Asynchronous requests

Bean calls default synchronous

Asynchronous request possible
- Long running methods
- Send mail, search database, ...
- Method annotation: @Asynchronous
- If annotation on class, all methods asynchronous
- IFuture<T> as return type (or void)

Restrictions on EJBs (and helper classes)

-Not use the java.lang.reflect Java Reflection API to access information unavailable by way of the security rules of the Java runtime environment
- Not read or write non-final static fields
- Not use this to refer to the instance in a method parameter or result
- Not use the java.awt/swing packages to create a user interface
- Not create or modify class loaders and security managers
- Not redirect input, output, and error streams
- Not obtain security policy information for a code source
- Not create or manage threads
- Not use thread synchronization primitives to synchronize access with other enterprise bean instances

…

…

- Not.....stop the Java virtual machine....

SUMMARY: Use as intended!

Calls: Beans and Non Beans

SB calling SB: OK

SB calling non bean (i.e. using some service): OK

Non bean calling SB
- Avoid
- Not possible with dependency injection in non beans, possible to fix with JNDI possible but tedious
- Also transactions issues etc.

Inject EJBs into CDI's

Possible to inject EJBs in various other parts of application

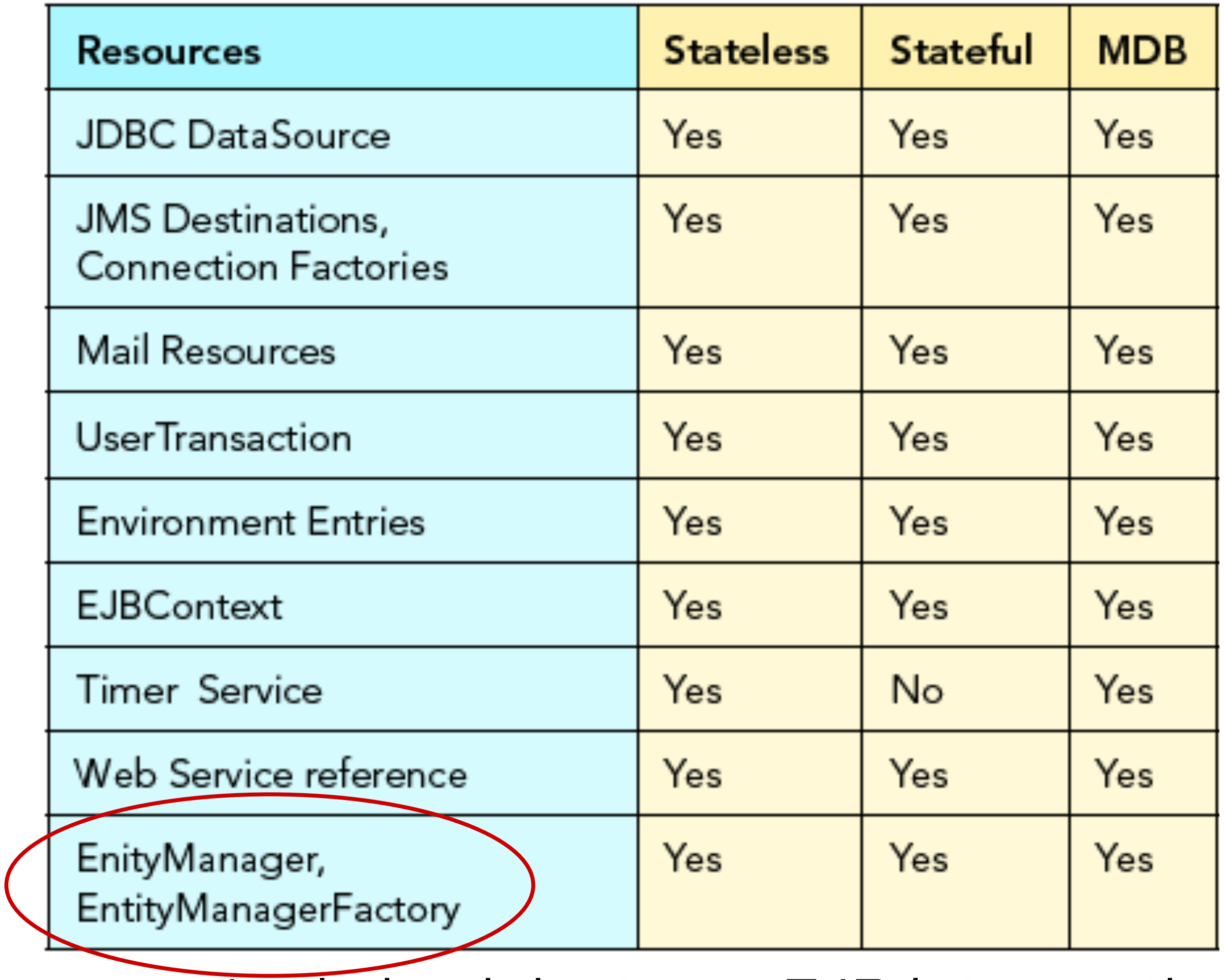
```
@Named("bean")    // CDI Bean
@RequestScoped
public class MyBean {
    @EJB                   // Inject EJB, @Inject should work but … ?!
    MyEJB myEjb;
```

Use: NetBeans > Insert Code ... > Call Enterprise Bean

CDI Scopes and EJBs

CDI scopes handled automagically, should be no problem

Other Resources to Inject into EJB's



Resources	Stateless	Stateful	MDB
JDBC DataSource	Yes	Yes	Yes
JMS Destinations, Connection Factories	Yes	Yes	Yes
Mail Resources	Yes	Yes	Yes
UserTransaction	Yes	Yes	Yes
Environment Entries	Yes	Yes	Yes
EJBContext	Yes	Yes	Yes
Timer Service	Yes	No	Yes
Web Service reference	Yes	Yes	Yes
EnityManager, EntityManagerFactory	Yes	Yes	Yes

And also inject one EJB into another

Timer Service

Used for long lived business processes
- Send an email each year: "Time for Christmas … buy, buy, buy"..
- EJB's define callback methods annotated with @Schedule, @Schedules
- Container call back at scheduled times
- A **persistent timer** will survive server crash

BUG in NetBeans 8.0.1
/GlassFish 4 b89
(worked last year)???

EJB as RESTful End Points

Possible, ..

```
@Stateless  // An EJB
public class MyResourceBean {

    @Context
    private UriInfo ui;
    @EJB
    OtherBean otherBean;

    @GET  // JAX-RS
    @Produces({"application/xml", "text/html"})
    public List<Message> getMessages() {...}

    @POST
    public Response addMessage(String msg) throws URISyntaxException
    {...}

}
```

EJB Modules

Possible to collect EJBs in reusable packages (jar-files).

- Possible for many front-ends to share common back-end
- Advanced deployment issues