

**CONTENTS INCLUDE:**

- What is EJB3?
- Resource Injection
- Injection of EJB References
- Injecting JPA Resources
- Injecting Spring Beans in EJB 3
- Hot Tips and more...

# Dependency Injection in **EJB 3**

By Debu Panda

## WHAT IS EJB 3?

Enterprise JavaBeans (EJB) is a platform for building portable, reusable, and scalable business applications using the Java programming language. Since its initial incarnation, EJB has been touted as a component model or framework that lets you build enterprise Java applications without having to reinvent services such as transactions, security, and automated persistence for building an application.

EJB 3 greatly simplifies development by adopting a POJO programming model. As shown in the following figure an annotation transforms a simple POJO to an EJB.



EJB 3 not only simplifies development of session and message driven beans but it also radically simplifies the persistence model by implementing a simplified Object-Relational Mapping approach similar to Oracle TopLink and JBoss Hibernate as a part of the Java Persistence API. Note that JPA replaces EJB 2 CMP Entity beans in the EJB 3 spec, while being available outside of the Java EE container.

Following is an example of a simple EJB 3 Stateless session bean.

```
import javax.ejb.Stateless;
import ejb3inaction.example.persistence.Bid;

@Stateless
public class PlaceBidBean implements PlaceBid {
    ...
    public PlaceBidBean() {}

    public Bid addBid(Bid bid) {
        System.out.println("Adding bid, bidder ID="

                + bid.getBidderID()
                + ", item ID=" + bid.getItemID()
                + ", bid amount="
                + bid.getBidAmount() + ".");
        return save(bid);
    }
    ...
}

import javax.ejb.Local;
import ejb3inaction.example.persistence.Bid;

@Local public interface PlaceBid {
    Bid addBid(Bid bid);
}
```

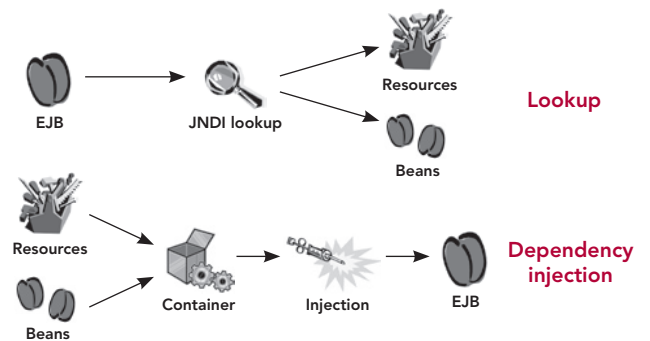
## DEPENDENCY INJECTION IN EJB 3

With EJB 3, dependency injection has greatly simplified accessing both EJB resources—such as JDBC DataSource, JMS Objects, and JPA Entity Manager—and services—such as Timer, User Transaction, and Web Services. You will find this Refcard useful when building enterprise Java applications with EJB 3 and JPA. It lists all metadata annotations, describes them and provides examples. It also provides descriptions for XML elements that you can use for injection.

Most enterprise Java applications use external resources and services such as Data Source, EJB, or web services. EJB 3 makes using resources and services simpler by implementing dependency injection.

Dependency injection allows you to simply declare component dependencies and let the EJB container deal with the complexities of instantiating, initializing, and sequencing resources and supplying service or resource references to clients as required. Development frameworks like Spring framework originally popularized dependency injection.

In EJB 3, you may think of dependency injection as the inverse of JNDI. It is the responsibility of the container to inject an object based on the dependency declaration. The figure below compares dependency injection with JNDI.





### Get More Refcardz (They're free!)

- Authoritative content
- Designed for developers
- Written by top experts
- Latest tools & technologies
- Hot tips & examples
- Bonus content online
- New issue every 1-2 weeks

**Subscribe Now for FREE!**  
[Refcardz.com](http://Refcardz.com)

## DEPENDENCY INJECTION IN EJB 3 *continued*



Note that annotations and descriptors are not mutually exclusive. In fact, you can use both together. Deployment descriptor entries override configuration values specified using metadata annotations.

You can use either metadata annotations or XML descriptors to use dependency injection.

Refer to the following Java EE 5 specifications and XML schema for more information on dependency injection.

### JSR 220

Enterprise JavaBeans 3.0:

<http://www.jcp.org/en/jsr/detail?id=220>

### JSR 224

Java API for XML-Based Web Services (JAX-WS) 2.0:

<http://jcp.org/en/jsr/detail?id=224>

### JSR 250:

Common Annotations for the Java Platform:

<http://www.jcp.org/en/jsr/detail?id=250>

Schema for EJB 3 deployment descriptor:

[http://java.sun.com/xml/ns/javaee/ejb-jar\\_3\\_0.xsd](http://java.sun.com/xml/ns/javaee/ejb-jar_3_0.xsd)

Schema that defines common schema components:

[http://java.sun.com/xml/ns/javaee/javaee\\_5.xsd](http://java.sun.com/xml/ns/javaee/javaee_5.xsd)

Dependency injection is supported only on managed classes and not on regular POJO.



Dependency injection is supported only on managed classes such as Session Beans or Interceptors and not on regular POJO. Hence you cannot use injection on helper classes.

The following table depicts what type of managed classes can inject what objects.

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

Java EE 5 introduced several metadata annotations as part of JSR 250. Although primarily geared toward EJB, these annotations also apply to Java EE components such as Servlets, JSF managed beans, and application clients.

The following annotations can be used in a field or a setter method for dependency injection. However, these annotations may also be applied at the class level for defining dependencies and then used with JNDI look up.

Annotations	Usage	Components that can use
javax.annotation.Resource	Dependency injection of resources such as DataSource, and JMS objects	EJB, Web, Application Client
javax.ejb.EJB	Dependency injection of Session beans	EJB, Web, Application Client
javax.xml.ws.WebServiceRef	Dependency injection of Web services	EJB, Web, Application Client
javax.persistence.PersistenceContext	Dependency injection of container-managed EntityManager	EJB, Web
javax.persistence.PersistenceUnit	Dependency injection of EntityManagerFactory	EJB, Web

## RESOURCE INJECTION

### javax.annotation.Resource

The @Resource annotation is used to inject any of the following: JDBC data sources, JMS connection factories, JMS destinations, mail resource, environment entries, timer service, UserTransaction, and EJBContext.

The following table shows attributes for the @Resource annotation.

Parameter	Type	Description	Default
authenticationType	enum AuthenticationType (CONTAINER, APPLICATION)	The type of authentication required for accessing the resource. The CONTAINER value means that the container's security context is used for the resource. The APPLICATION value means that authentication for the resource must be provided by the application.	CONTAINER
name	String	The referred resources is bound under this name in the environment-naming context as java:comp/env/<Name>	""
type	Object	Type of the resource being referenced. Example: javax.sql.DataSource	Object.class
shareable	Boolean	Specifies whether the resource can be shared.	True
description	String	The description of the resource.	""
mappedName	String	A vendor-specific name that the resource may be mapped to, as opposed to the JNDI name.	""

You can inject a resource at the field or setter method level.

The following example shows data source injection at the field level:

```
@Resource(name="jdbc/ActionBazaarDS")
private DataSource dataSource;
```

The following code shows data source injection at the setter method level:

```
private DataSource dataSource;
@Resource(name="jdbc/ActionBazaarDB")
public void setDataSource(DataSource dataSource) {
    this.dataSource = dataSource;
}
```

## RESOURCE INJECTION *continued*



Although setter injection might seem like a little more work, it provides a couple of distinct advantages. First, it is easier to unit-test by invoking the public setter method from a testing framework like JUnit. Second, it is easier to put initialization code in the setter if you need it.

EJB 3 allows you to explicitly specify a global JNDI name using the `mappedName` parameter of the `@Resource` annotation. For example, if you're using the Oracle Application Server and you have a data source with a global JNDI name of `jdbc/OracleDS`, you can specify the resource mapping as follows:

```
@Resource(name="jdbc/ActionBazaarDS", mappedName="jdbc/OracleDS")
private javax.jdbc.DataSource myDB;
```

### XML elements to define Resource Injection

If you are using deployment descriptor (`ejb-jar.xml`) instead of annotations, then you must use `resource-ref`, `resource-env-ref` or `env-entry` XML elements to define dependencies.

#### *resource-ref*

The `resource-ref` is used to specify resource references. Example: data source and JMS connection factories.

Element/Attribute Name	Description
<code>res-ref-name</code>	The name used to bind the referenced resource into the ENC. Same as the name element in the <code>@Resource</code> annotation.
<code>mapped-name</code>	A vendor-specific global JNDI name for the referenced resource.
<code>res-type</code>	Fully qualified class of the type of resource referenced. Example: <code>javax.sql.DataSource</code> .
<code>res-auth</code>	Authentication type for the resource. Valid values are <code>Container</code> or <code>Application</code> .
<code>res-sharing-scope</code>	Specifies whether multiple beans can share the resource. Valid values are <code>Shareable</code> and <code>Unshareable</code> .
<code>injection-target</code>	Target where the referenced resource is injected when dependency injection is used.

```
<resource-ref>
  <res-ref-name>jdbc/ActionBazaarDS</res-ref-name>
  <res-type>javax.sql.DataSource</res-type>
  <res-auth>Container</res-auth>
  <res-sharing-scope>Shareable</res-sharing-scope>
  <injection-target>
    <injection-target-class>
      actionbazaar.buslogic.BidManagerBean</injection-target-class>
    <injection-target-name>dataSource</injection-target-name>
  </injection-target>
</resource-ref>
```

#### *resource-env-ref*

The `resource-env-ref` is used to specify references to JMS destination resources such as a `Queue` or `Topic`.

Element/Attribute Name	Description
<code>resource-env-ref-name</code>	The name used to bind the referenced JMS destination to the ENC. Same as the name element in the <code>@Resource</code> annotation.
<code>mapped-name</code>	A vendor-specific global JNDI name for the referenced JMS destination.
<code>resource-env-type</code>	Type of JMS destination referenced, such as <code>javax.jms.Queue</code> or <code>javax.jms.Topic</code> .
<code>injection-target</code>	Target where the referenced destination is injected when dependency injection is used.

```
<resource-env-ref>
<resource-env-ref-name>jms/OrderBillingQueue</resource-env-ref-name>
<resource-env-ref-type>
  javax.jms.Destination
</resource-env-ref-type>
<injection-target>
  <injection-target-class>
    ejb3inaction.example.buslogic.PlaceOrderBean
  </injection-target-class>
  <injection-target-name>billingQueue</injection-target-name>
</injection-target>
</resource-env-ref>
```

#### *env-entry*

The `env-entry` defines environment entries for an EJB.

Element/Attribute Name	Description
<code>env-entry-name</code>	The name used in the environment entry in the ENC. Same as the name element in the <code>@Resource</code> annotation.
<code>env-entry-type</code>	Type of the env entry used. Legal types are <code>java.lang.Boolean</code> , <code>java.lang.Byte</code> , <code>java.lang.Character</code> , <code>java.lang.String</code> , <code>java.lang.Short</code> , <code>java.lang.Integer</code> , <code>java.lang.Long</code> , <code>java.lang.Float</code> , and <code>java.lang.Double</code> .
<code>env-entry-value</code>	Value specified for the environment entry.
<code>injection-target</code>	Target where the referenced destination is injected when dependency injection is used.

The `injection-target` defines the name of a class and a name (field or property) within that class into which a resource, EJB, or entity manager should be injected as we saw in the above examples.

Element/Attribute Name	Description
<code>injection-target-class</code>	The fully qualified name of the class into which a resource, EJB, or entity manager should be injected.
<code>injection-target-name</code>	Name of the injection target, i.e., the name of the property or field in the injection target class.

## INJECTION OF SESSION EJB REFERENCES

Like resource injection, you can inject Session bean references into another EJB or other managed class using annotations or deployment XML. If you prefer annotations you can use `@javax.ejb.EJB` annotations to inject either remote or local EJB references. Using deployment XML, you can inject remote EJB references with `ejb-ref` and local EJB references with `ejb-local-ref`.

**@javax.ejb.EJB**

Injects a session bean reference into a field or method.

Parameter	Type	Description	Default
name	String	The name used to bind the referenced EJB to the ENC.	""
beanInterface	Class	The bean interface used to access the EJB	Object.class
mappedName	String	Type of the resource being referenced. Example: javax.sql.DataSource	""
beanName	String	Specifies whether the resource can be shared.	""
description	String		

```
@EJB(name="BidManagerRemote")
private BidManager bidManager;
```



You must not inject a Stateful session bean into a stateless object, such as a stateless session bean or servlet that may be shared by multiple concurrent clients (you should use JNDI in such cases instead). However, injecting an instance of a stateless session bean into a stateful session bean is perfectly legal.

**ejb-local-ref**

Used to specify a dependency on the local interface of a session bean.

Element/Attribute Name	Description
ejb-ref-name	The name used to bind the referenced EJB to the ENC. Same as the name element in the @EJB annotation. ejb-ref-name must be specified.
ejb-link	The name of the target enterprise bean. This optional setting is used to link an EJB reference to a target enterprise bean.
local	The EJB 3 local business interface.
ref-type	The EJB reference type, i.e. "session".
injection-target	Target where the EJB reference is injected when dependency injection is used.

**ejb-ref**

Used to specify a dependency on the remote interface of a session bean.

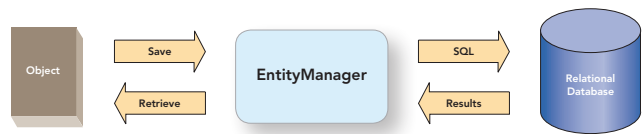
Element/Attribute Name	Description
ejb-ref-name	The name used to bind the referenced EJB to the ENC. Same as the name element in the @EJB annotation. ejb-ref-name must be specified.
ejb-link	The name of the target enterprise bean. This optional setting is used to link an EJB reference to a target enterprise bean.
remote	The EJB 3 remote business interface type.
ref-type	The EJB reference type, i.e. "session".
injection-target	Target where the EJB reference is injected when dependency injection is used.



EJB 3 specification does not require injecting references of remote EJB into a different instance of container. For example if you have an EJB with a remote interface deployed into an instance of a container, EJB 3 specification does not require injecting instance of that EJB into another EJB deployed in another container.

**INJECTING JPA RESOURCES**

The JPA EntityManager is the bridge between the OO and relational worlds as depicted in the following figure.



JPA supports two types of EntityManager: container-managed and application-managed. As the name suggests the lifecycle of the container manages container-managed EntityManager whereas the lifecycle (creation, destruction) is performed by application. An application-managed EntityManager is created from an EntityManagerFactory.

**Injecting container-managed EntityManager**

You can inject a container-managed EntityManager (either transaction scoped or extended) using @javax.persistence.PersistenceContext annotation.

**@javax.persistence.PersistenceContext**

The following table defines the parameters/attributes of this annotation.

Parameter	Type	Description	Default
name	String	The name used to bind the referenced persistence context to the ENC	""
unitName	String	Name of the persistence unit referenced	""
type	Persistence ContextType	Type of persistence context, i.e., Transaction or Extended. Extended is supported only in stateful session beans.	TRANSACTION
properties	Persistence Property[]	A name value-pair of vendor-specific persistence properties	{}

You can inject a transaction scoped EntityManager as follows:

```
@PersistenceContext(unitName="actionBazaar")
private EntityManager entityManager;
```

An extended scoped EntityManager can only be used in a stateful session bean as in the following example:

```
@Stateful
@TransactionAttribute(TransactionAttributeType.NOT_SUPPORTED)
public class PlaceOrderBean implements PlaceOrder {
    @PersistenceContext(unitName = "actionBazaar", type =
        PersistenceContextType.EXTENDED)
    EntityManager em;
```

## Injecting JPA Resources, continued

The persistence-context-ref is similar to @PersistenceContext; it defines references to a container-managed entity manager.

Element/Attribute Name	Description
persistence-context-ref-name	The name used to bind the referenced persistence context to the ENC. Same as the name element in the @PersistenceContext annotation.
persistence-unit-name	Name of the persistence unit referenced.
persistence-context-type	Type of persistence context, i.e., Transaction or Extended. Extended is supported only in stateful session beans.
persistence-property	A name value-pair of vendor-specific persistence properties.
injection-target	Target where the EntityManager is injected when dependency injection is used.

```
<persistence-context-ref>
  <persistence-context-ref-name>
    ActionBazaar
  </persistence-context-ref-name>
  <persistence-unit-name>actionBazaar</persistence-unit-name>
  <persistence-context-type>Transaction</persistence-context-type>
  <injection-target>
    <injection-target-class>
      ejb3inaction.example.buslogic.PlaceBidBean
    </injection-target-class>
    <injection-target-name>em</injection-target-name>
  </injection-target>
</persistence-context-ref>
```

## Injecting EntityManagerFactory

### @javax.persistence.PersistenceUnit

```
@Stateless
public class ItemManagerBean implements ItemManager {
  @PersistenceUnit
  private EntityManagerFactory entityManagerFactory;
  private EntityManager entityManager;
  @PostConstruct
  public void initialize() {
    entityManager = entityManagerFactory.createEntityManager();
  }
}
```

Parameter	Type	Description	Default
name	String	The name used to bind the referenced persistence context to the ENC	""
unitName	String	Name of the persistence unit	""

The persistence-unit-ref is similar to @PersistenceUnit that is used to define references to a persistence unit (i.e., entity manager factory).

Element/Attribute Name	Description
persistence-unit-ref-name	The name used to bind the referenced persistence unit (EntityManagerFactory) to the ENC. Same as the name element in the @PersistenceUnit annotation.
persistence-unit-name	Name of the persistence unit referenced.
injection-target	Target where the EntityManagerFactory is injected when dependency injection is used.

## INJECTING WEB SERVICE REFERENCES

### @javax.xml.ws.WebServiceRef

Element	Description
name	The JNDI name for the web service.
wSDLLocation	The WSDL location for the service. If not specified, then it is derived from the referenced service class.
type	The Java type of the resource.
value	The service class, always a type extending javax.xml.ws.Service.
mappedName	Vendor-specific global JNDI name for the service.

You can inject an endpoint interface as follows:

```
@WebServiceRef(TrackDeliveryService.class)
private TrackDeliverySEI deliveryService;
```

You can inject a service interface as follows:

```
@WebServiceRef
private TrackingService service;
```

### service-ref

The service-ref XML element is used to specify dependency on a web service if you are using deployment descriptor. The following table contains only the elements that are used from EJB clients.

Element/Attribute Name	Description
service-ref-name	The name used to bind the referenced web service into the ENC. Same as the name element in the @WebServiceRef annotation.
service-interface	Fully qualified class for the JAX-WS service interface the client depends on., i.e. javax.xml.rpc.Service.
service-ref-type	Type of service that will be returned.
wSDL-file	The URL location of the WSDL.
handler-chains	Defines handler chain.
injection-target	Target where the web service reference is injected when dependency injection is used.

The detailed schema for the service-ref can be found online at [java.sun.com/xml/ns/javaee/javaee\\_web\\_services\\_client\\_1\\_2.xsd](http://java.sun.com/xml/ns/javaee/javaee_web_services_client_1_2.xsd).

## INJECTING SPRING BEANS IN EJB 3

The Spring Framework is one of the driving forces behind popularizing the POJO programming model and dependency injection. In this section we will examine how you can inject and use Spring POJOs into EJB 3 Session beans or Message Driven Beans.

The Spring framework provides factory classes based on the following abstract classes that you use to develop Spring-enabled EJBs.

Support Class	Purpose
AbstractStatelessSessionBean	Used for Spring-enabled stateless session beans.
AbstractStatefulSessionBean	Used for Spring-enabled stateful session beans.
AbstractJMSMessageDrivenBean	Used for Spring-enabled JMS message-driven beans.
AbstractMessageDrivenBean	Used for Spring-enabled connector-based MDBs.

To use a Spring factory class to access a Spring bean, your EJB 3 bean class must implement the onEjbCreate() method.

Below is the PlaceBidBean EJB 3 example transformed into a Spring-enabled stateless session bean. Here the PlaceBidBean

### Injecting Spring Beans in EJB 3, continued

acts as a façade and delegates the actual business logic to the PlaceBidServiceBean. The PlaceBidServiceBean is a Spring POJO that may use the full power of the Spring framework.

```
@Stateless(name = "PlaceBid")
public class PlaceBidBean extends AbstractStatelessSessionBean
implements PlaceBid {

    private BidServiceBean bidService;
    public PlaceBidBean() {

    }

    protected void onEjbCreate() {
        bidService =
        (BidServiceBean) getBeanFactory().getBean("bidService");
    }

    public Long addBid(String userId, Long itemId, Double bidPrice) {
        return bidService.addBid(userId, itemId, bidPrice);
    }
}
```

When an EJB instance is created (when a client invokes an EJB), the onEjbCreate method is invoked automatically. A JNDI look-

up is performed to obtain the path for the bean factory by using an environment entry named ejb/BeanFactoryPath. So you have to define it in the EJB deployment descriptor for the EJB:

```
<session>
  <display-name>PlaceBid</display-name>
  <ejb-name>PlaceBid</ejb-name>
  <env-entry>
    <env-entry-name>ejb/BeanFactoryPath</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>
    <env-entry-value>/actionBazaar-service.xml</env-entry-value>
  </env-entry>
</session>
```

Although EJB 3 made deployment descriptor optional there are a few cases where you still have to use it. In our example we've set the env-entry-value for the ejb/BeanFactoryPath environment variable at /actionBazaar-service.xml. So you have to package the EJB classes, Spring classes, and Spring configuration file into your ejb-jar package. The Spring bean factory (configuration file) defines the Spring beans. Refer to the **Spring Configuration Refcard** for details about Spring configuration.

### ABOUT THE AUTHOR



#### Debu Panda

Debu Panda is a Senior Principal Product Manager of the System Management Products team at Oracle, where he drives development of the middleware management product. He has more than 15 years of experience in the IT industry and has published numerous articles on enterprise Java technologies in several magazines and has presented at many conferences. His J2EE-focused weblog can be found at [debupanda.com](http://debupanda.com).

#### Publications

- *EJB3 in Action*, 2007

#### Projects

- Oracle Application Server

#### Blog

- Personal weblog: [debupanda.com](http://debupanda.com)

#### Articles

- *Spring and Java EE 5, Part 1*
- *Spring and Java EE 5, Part 2*
- *ONJava.com—Standardizing Java Persistence with the EJB3 Java Persistence API*
- *Migrating JDBC DAO to EJB3*

### RECOMMENDED BOOK



*EJB 3 in Action* starts with a tour of the EJB 3 landscape, then moves quickly into core topics like building business logic with session and message-driven beans. It covers the JPA along with practical code samples, design patterns, performance tuning tips, and best practices for building and deploying scalable applications.

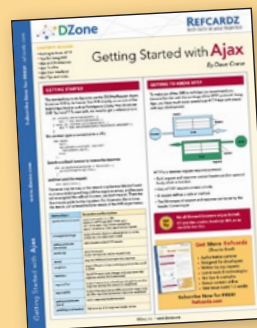
### BUY NOW

[books.dzone.com/books/ejb3-in-action](http://books.dzone.com/books/ejb3-in-action)

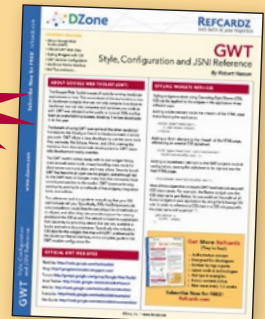
## Subscribe Now for FREE! [refcardz.com](http://refcardz.com)

### Upcoming Refcardz:

- RSS and Atom
- Flexible Rails: Flex 3 on Rails 2
- C#
- GlassFish Application Server
- jQuery Selectors
- Design Patterns
- MS Silverlight 2
- NetBeans IDE 6 Java Editor
- Groovy
- Apache Struts 2



Getting Started with Ajax



GWT Style, Configuration and JSNI Reference



The **DZone Network** is a group of free online services that aim to satisfy the information needs of software developers and architects. From news, blogs, tutorials, source code and more, DZone offers everything technology professionals need to succeed.

To quote *PC magazine*, "DZone is a developer's dream."

DZone, Inc.  
1251 NW Maynard  
Cary, NC 27513  
888.678.0399  
919.678.0300

**Refcardz Feedback Welcome**  
[refcardz@dzone.com](mailto:refcardz@dzone.com)

**Sponsorship Opportunities**  
[sales@dzone.com](mailto:sales@dzone.com)



\$7.95