12.–15.09.2010
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Bohnen auf dem Prüfstand

Effiziente EJB3 Unit-Tests

Karol Rückschloss

MATHEMA Software GmbH

Unit Tests

Agenda

- Einführung
  - EJBs, Container, Unit Tests, JUnit
- Mögliche Ansätze für Unit Tests
  - (Integrations-)Tests mit AppServer, Unmanaged, Embedded
- Embedded OpenEJB
  - Starten des Containers, Scannen nach Beans, JNDI, Injection
- Testansätze mit Embedded OpenEJB
  - Persistenz, Security, MDBs
- Embedded Glassfish

Agenda

- **Einführung**
  - EJBs, Container, Unit Tests, JUnit
- Mögliche Ansätze für Unit Tests
  - (Integrations-)Tests mit AppServer, Unmanaged, Embedded
- Embedded OpenEJB
  - Starten des Containers, Scannen nach Beans, JNDI, Injection
- Testansätze mit Embedded OpenEJB
  - Persistenz, Security, MDBs
- Embedded Glassfish

Kurzer Überblick: EJB 3

- Session Beans
  @Stateless
  @Stateful
  Lokale und Remote Interfaces
- Message Driven Beans
  @MessageDriven
- Persistenz mit JPA
- Security
- Callbacks / Lifecycle

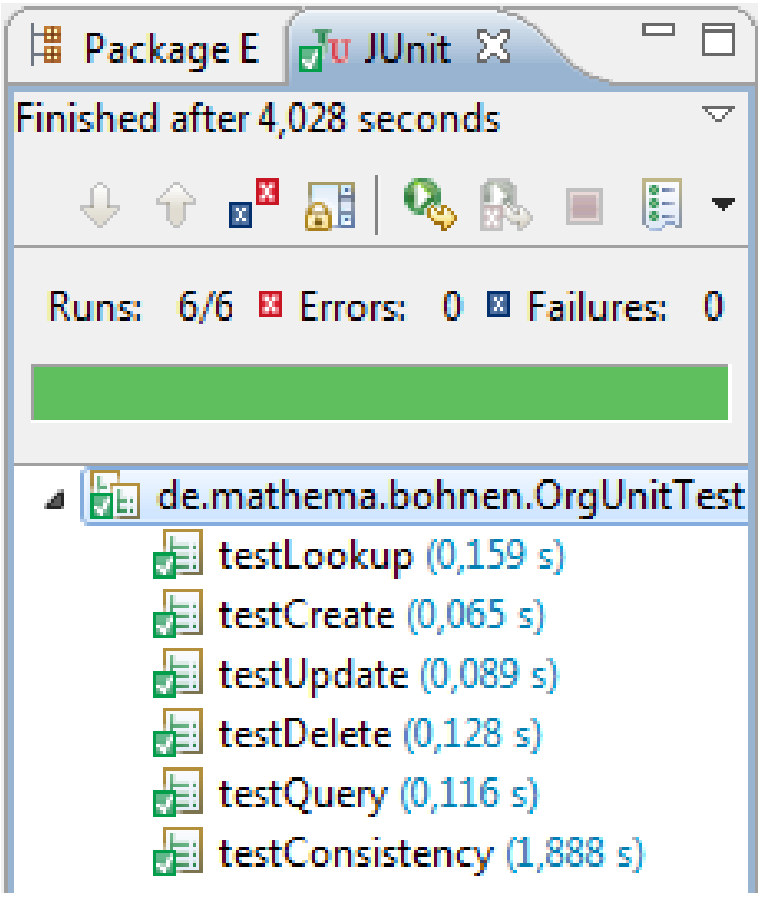
EJB-Container

EJB-Container

+ Lifecycle
+ Remoting
+ Callbacks
+ Pooling
+ Transaktionen

Kurzer Überblick: JUnit

- Framework für Unit Tests
- Testklassen
- @Test Methoden
- @Before, @After
- @BeforeClass, @AfterClass

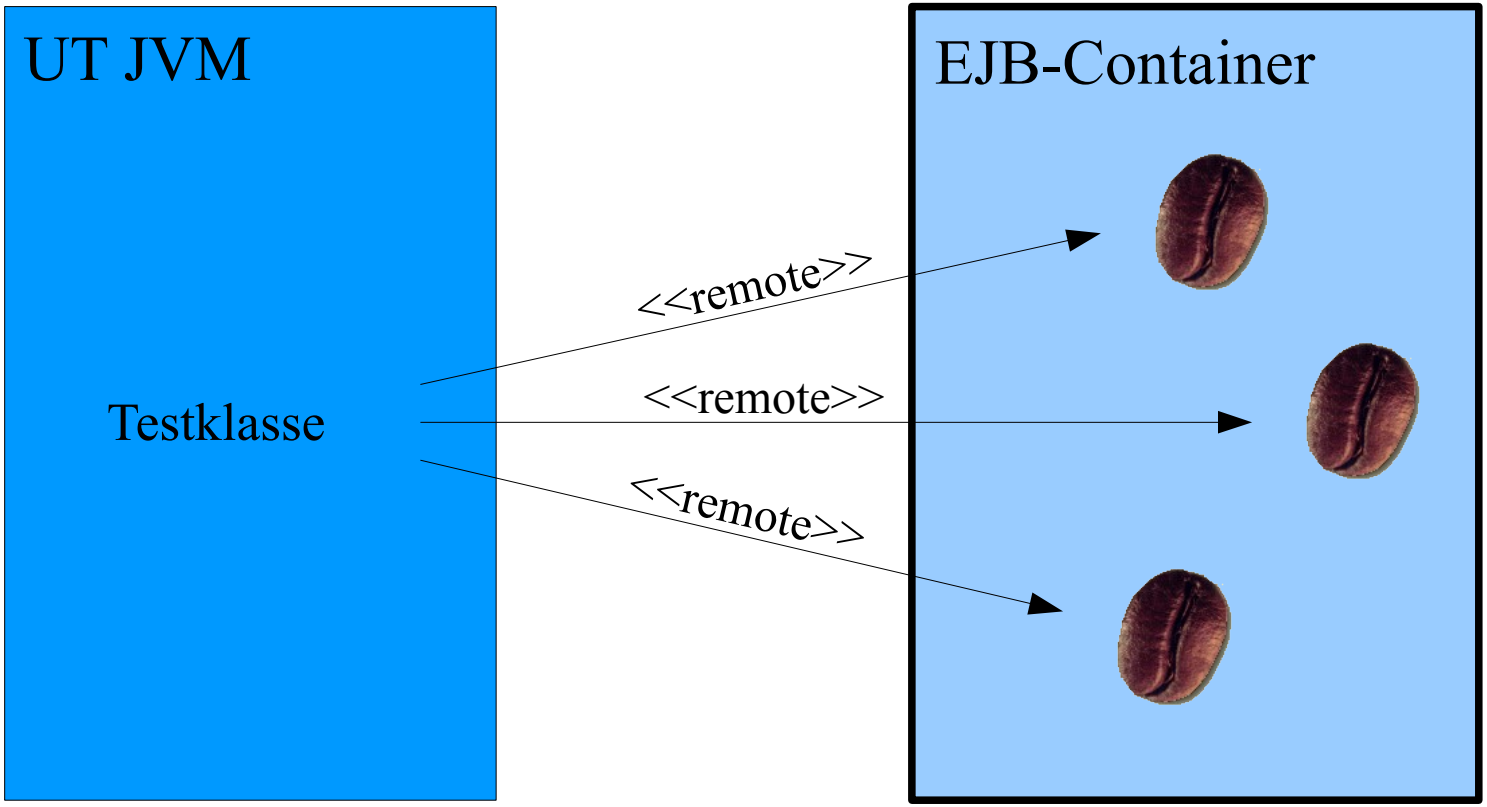


Agenda

- Einführung
  - Unit Tests, EJBs, Container, JUnit
- **Mögliche Ansätze für Unit Tests**
  - (Integrations-)Tests mit AppServer, Unmanaged, Embedded
- Embedded OpenEJB
  - Starten des Containers, Scannen nach Beans, JNDI, Injection
- Testansätze mit Embedded OpenEJB
  - Persistenz, Security, MDBs
- Embedded Glassfish

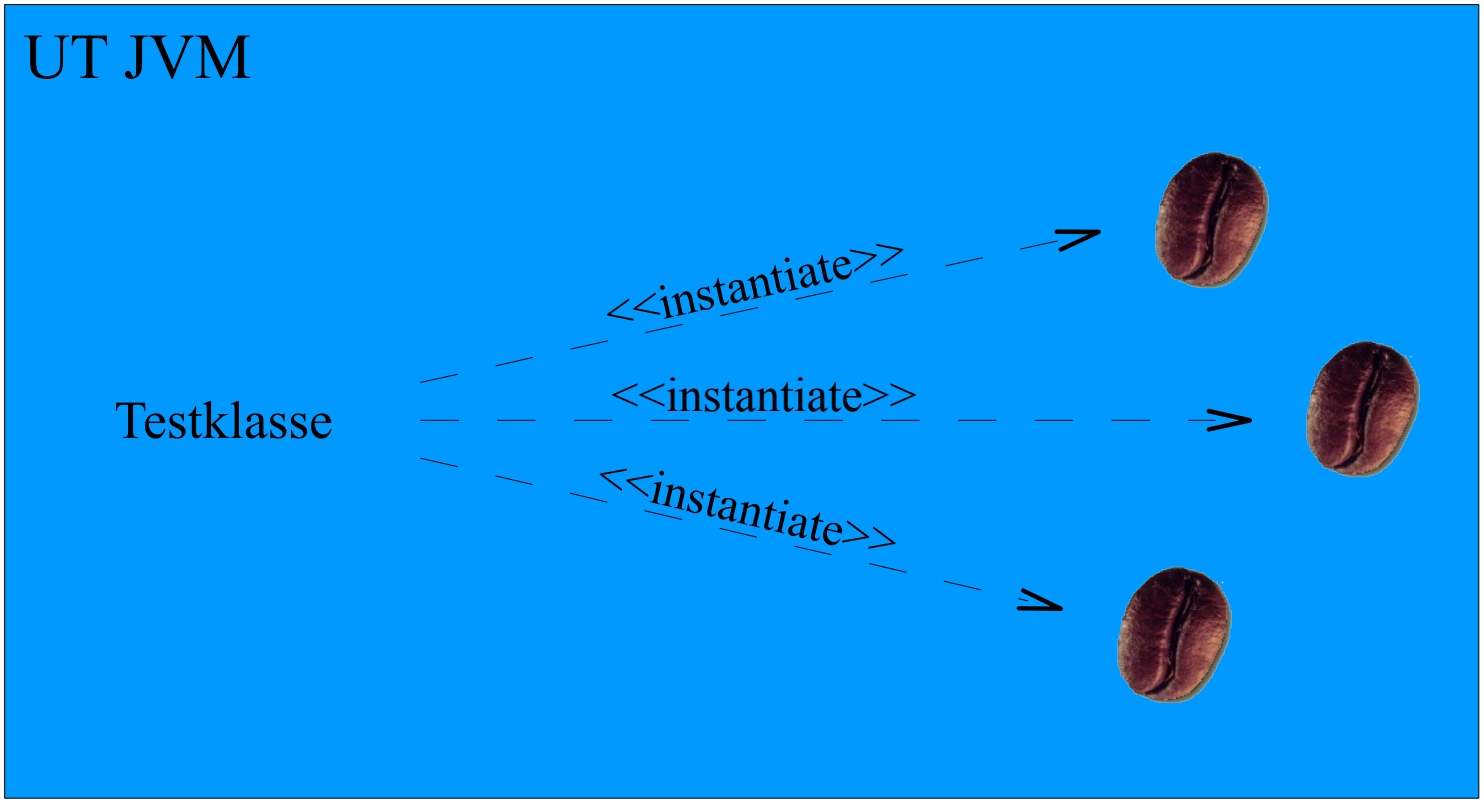
Mögliches Unit-Testen: Produktiv-Server

- App wird im „richtigen“ Ziel-Server deployed



Mögliches Unit-Testen: unmanaged Klassen

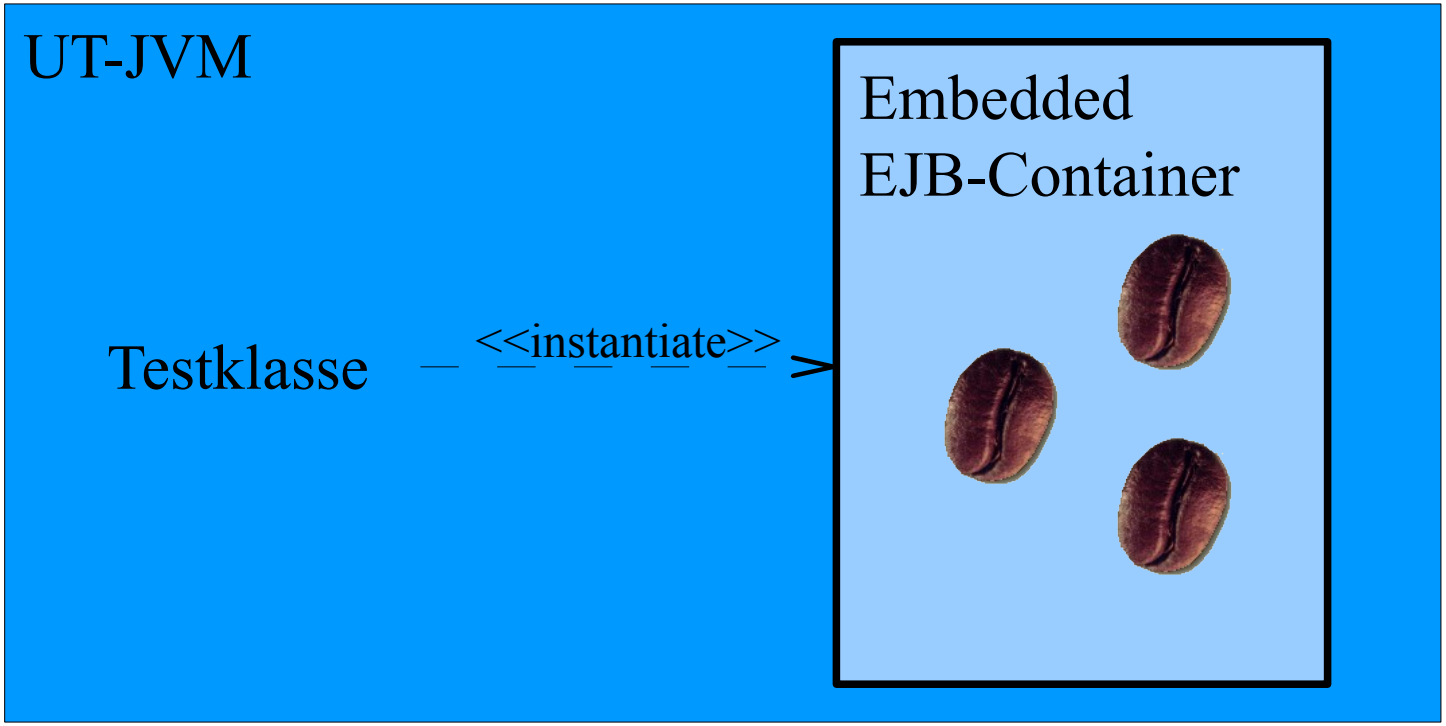
- Die Bean-Klassen werden direkt instanziiert



Testen mit Embedded Container

Embedded Container

- EJB Container, der in unserem Classloader gestartet werden kann



Agenda

- Einführung
  - Unit Tests, EJBs, Container, JUnit
- Mögliche Ansätze für Unit Tests
  - (Integrations-)Tests mit AppServer, Unmanaged, Embedded
- **Embedded OpenEJB**
  - Starten des Containers, Scannen nach Beans, JNDI, Injection
- Testansätze mit Embedded OpenEJB
  - Persistenz, Security, MDBs
- Embedded Glassfish

Apache OpenEJB

- www.openejb.org
- EJB 3.0 Implementierung
- Standalone und Embedded
- Unterstützt:
  - EJB 3.0, 2.1, 2.0, 1.1
  - JPA, JAX-WS, JMS, J2EE Connectors und mehr
- Bestandteil von Apache Geronimo, IBM Websphere Application Server CE, Apple WebObjects

OpenEJB: Starten des Containers

```
@BeforeClass
public static void init() throws Exception {

   Properties p = new Properties();
   p.put(Context.INITIAL_CONTEXT_FACTORY,
      "org.apache.openejb.client.LocalInitialContextFactory");

   InitialContext ctx = new InitialContext(p);

}
```

OpenEJB: Starten des Containers

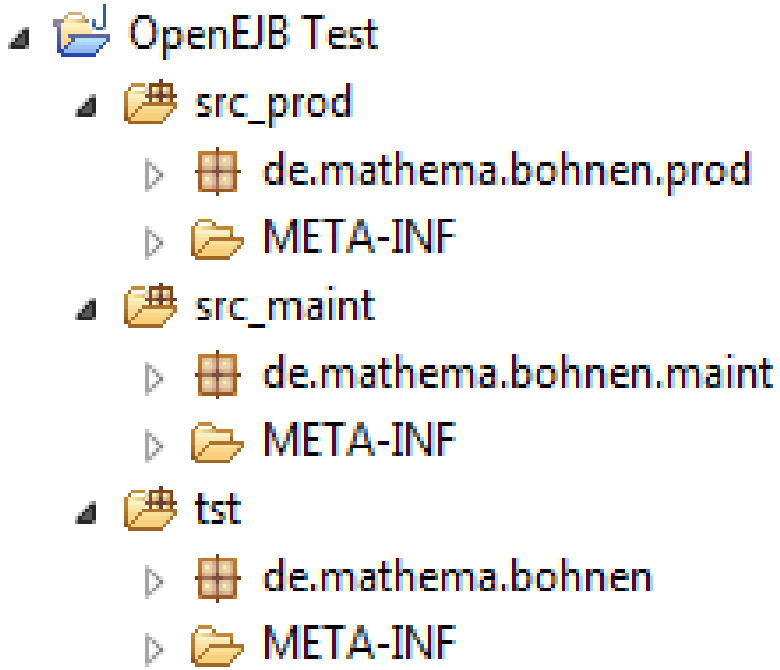
```
Apache OpenEJB 3.1.2    build: 20091010-03:11
http://openejb.apache.org/
INFO - openejb.home = C:\Java\eclipse\OpenEJB Test
INFO - openejb.base = C:\Java\eclipse\OpenEJB Test
INFO - Configuring Service(id=Default Security Service, type=SecurityService, provider-id=Default Security Service)
INFO - Configuring Service(id=Default Transaction Manager, type=TransactionManager, provider-id=Default Transaction
INFO - Found ClientModule in classpath: C:\Java\openejb-3.1.2\lib\xml-resolver-1.2.jar
INFO - Found ClientModule in classpath: C:\Java\openejb-3.1.2\lib\serializer-2.7.1.jar
INFO - Found EjbModule in classpath: C:\Java\eclipse\OpenEJB Test\bin\tst
INFO - Found EjbModule in classpath: C:\Java\eclipse\OpenEJB Test\bin\src
INFO - Beginning load: C:\Java\openejb-3.1.2\lib\xml-resolver-1.2.jar
INFO - Beginning load: C:\Java\openejb-3.1.2\lib\serializer-2.7.1.jar
INFO - Beginning load: C:\Java\eclipse\OpenEJB Test\bin\tst
INFO - Beginning load: C:\Java\eclipse\OpenEJB Test\bin\src
INFO - Configuring enterprise application: classpath.ear
INFO - Configuring Service(id=Default Stateless Container, type=Container, provider-id=Default Stateless Container)
INFO - Auto-creating a container for bean EchoBean: Container(type=STATELESS, id=Default Stateless Container)
INFO - Enterprise application "classpath.ear" loaded.
INFO - Assembling app: classpath.ear
INFO - Jndi(name=ejb/EchoBean) --> Ejb(deployment-id=EchoBean)
INFO - Created Ejb(deployment-id=EchoBean, ejb-name=EchoBean, container=Default Stateless Container)
INFO - Deployed Application(path=classpath.ear)
```

OpenEJB: Scannen nach Beans

OpenEJB: Scannen nach Beans

- Simulation eines deployten EAR „classpath.ear“
- **Möglichkeit 1: META-INF/ejb-jar.xml's**
  - Suche nach META-INF/ejb-jar.xml im Classpath
  - ejb-jar.xml darf auch nur aus <ejb-jar/> bestehen
- **Möglichkeit 2: Include/Exclude Properties**
  - openejb.deployments.classpath.filter.descriptors
  - openejb.deployments.classpath.include
  - openejb.deployments.classpath.exclude

Scannen nach META-INF/ejb-jar.xml




```
INFO - Found EjbModule in classpath: C:\Java\eclipse\OpenEJB Test\bin\src_prod
INFO - Found EjbModule in classpath: C:\Java\eclipse\OpenEJB Test\bin\src_maint
INFO - Found EjbModule in classpath: C:\Java\eclipse\OpenEJB Test\bin\tst
```

Scannen gemäß include/exclude

```
Properties p = new Properties();
// ...
p.put("openejb.deployments.classpath.filter.descriptors", "true");
p.put("openejb.deployments.classpath.exclude", ".*src_maint.*");

InitialContext ctx = new InitialContext(p);
```

```
INFO - Found EjbModule in classpath: C:\Java\eclipse\OpenEJB Test\bin\src_prod
INFO - Found EjbModule in classpath: C:\Java\eclipse\OpenEJB Test\bin\src_maint
INFO - Found EjbModule in classpath: C:\Java\eclipse\OpenEJB Test\bin\tst
```

OpenEJB: JNDI Naming

OpenEJB: JNDI Naming

```
@Stateless
public class EchoBean implements IEchoBean {
   public String echo(String s) {
        return "Hello " + s;
   }
}
```

OpenEJB bindet Beans mit dem JNDI-Namen:
{deploymentId}{interfaceType.annotationName}

INFO - Jndi(name=**EchoBeanRemote**) -->
Ejb(deployment-id=EchoBean)

OpenEJB: JNDI Naming

OpenEJB JNDI-Name: "EchoBeanRemote"

Lookup im Client von JBoss AS:

```
Object bean = ctx.lookup("echo/EchoBean/remote");
```

=> javax.naming.NameNotFoundException: Name "echo/EchoBean/remote" not found.

AppServer-abhängige JDNI-Namen...

1) EchoBean
2) EchoBeanRemote
3) java:comp/env/EchoBean
4) de.mathema.bohnen.prod.IEchoBean
5) ejb-jar_EchoBean

OpenEJB: JNDI Naming

- **Lösung 1:** Mapped Name

```
@Stateless(mappedName="myapp/EchoBean/remote")
public class EchoBean implements IEchoBean { … }
```

- **Lösung 2:** Property openejb.jndiname.format

  z.B.: myapp/{ejbName}/{interfaceType.annotationNameLC}

- **Lösung 3:** Dependency Injection

```
public class MyTest {
  @EJB IEchoBean bean;
  @Test
  public void testEchoBean() { bean.echo("Foo"); }
}
```

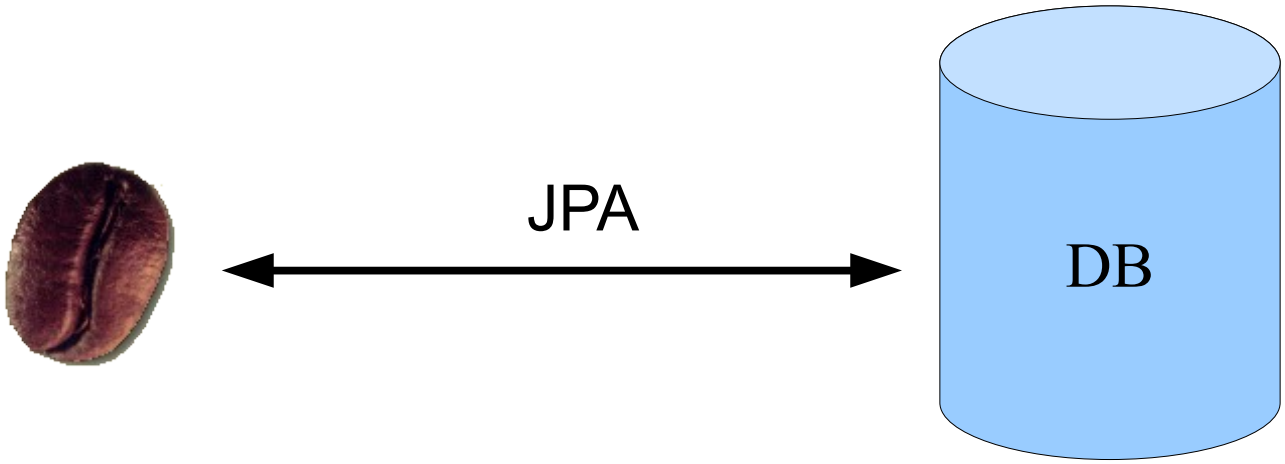
OpenEJB: @LocalClient Injection

```
@LocalClient
public class MyTest {
    protected static InitialContext ctx;

    @EJB
    private IEchoBean bean;
    @PersistenceContext
    private EntityManager em;
    @Resource
    private UserTransaction userTransaction;

    @Before
    public void before() throws NamingException {
        ctx.bind("inject", this);
    }

}
```

Agenda

- Einführung
  - Unit Tests, EJBs, Container, JUnit
- Mögliche Ansätze für Unit Tests
  - (Integrations-)Tests mit AppServer, Unmanaged, Embedded
- Embedded OpenEJB
  - Starten des Containers, Scannen nach Beans, JNDI, Injection
- **Testansätze mit Embedded OpenEJB**
  - Persistenz, Security, MDBs
- Embedded Glassfish

Testen mit JPA-Persistenz



Testen mit JPA-Persistenz

```
@Entity
public class Address implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long id;
    @Column(nullable = false)
    private String street;
    private String zipCode;
    @Column(nullable = false)
    private String city;
    private String country;

    // Setter, Getter, usw.
}
```

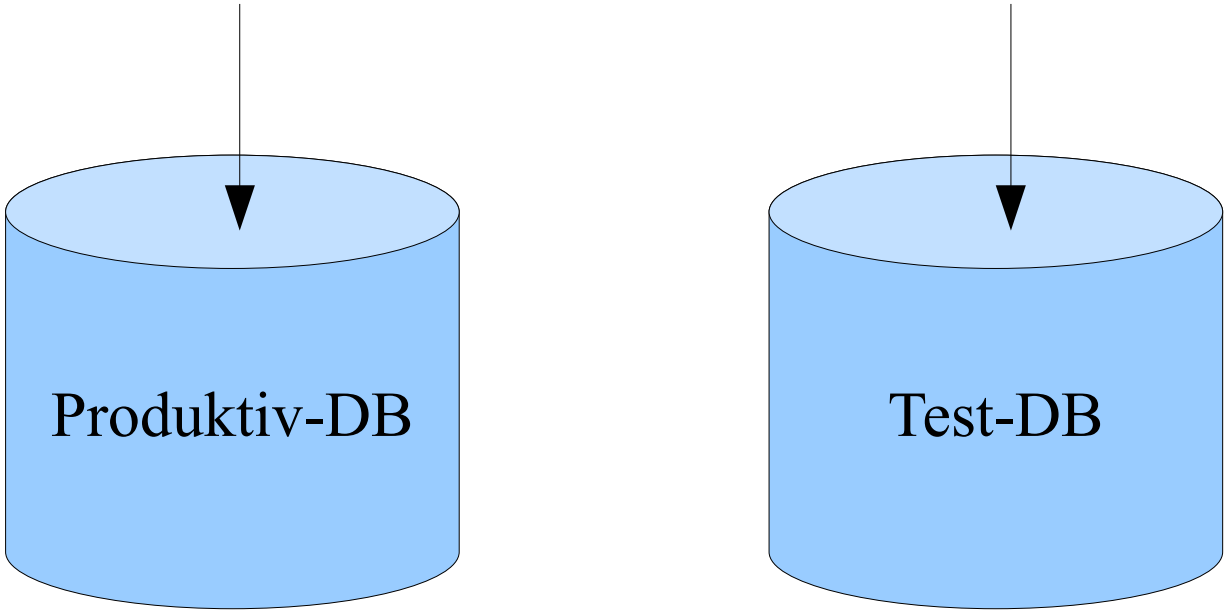
Testen mit JPA-Persistenz

```
@Stateless
@PersistenceUnit(unitName = "address-db")
public class AddressManagerBean {

   @PersistenceContext
   private EntityManager em;

   @TransactionAttribute(TransactionAttributeType.REQUIRED)
   public Address create(Address a) {
      em.persist(a);
      return a;
   }

}
```

Testen mit JPA-Persistenz

Produktiv-Umgebung
persistence.xml in
EAR/ejb-jar.jar/META-INF

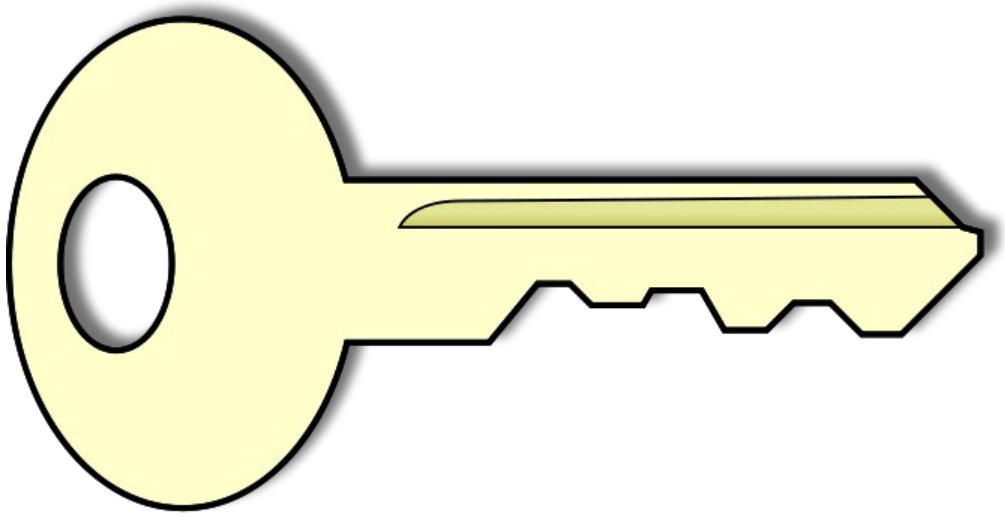
Test-Umgebung
persistence.xml in
tst/META-INF



Testen mit JPA-Persistenz

```
Properties p = new Properties();
p.put(Context.INITIAL_CONTEXT_FACTORY,
        "org.apache.openejb.client.LocalInitialContextFactory");

p.put("addressDS", "new://Resource?type=DataSource");
p.put("addressDS.JdbcDriver", "org.apache.derby.jdbc.EmbeddedDriver");
p.put("addressDS.JdbcUrl", "jdbc:derby:derbyDB;create=true");
p.put("addressDS.JtaManaged", "true");

InitialContext ctx = new InitialContext(p);
```

Testen von Security



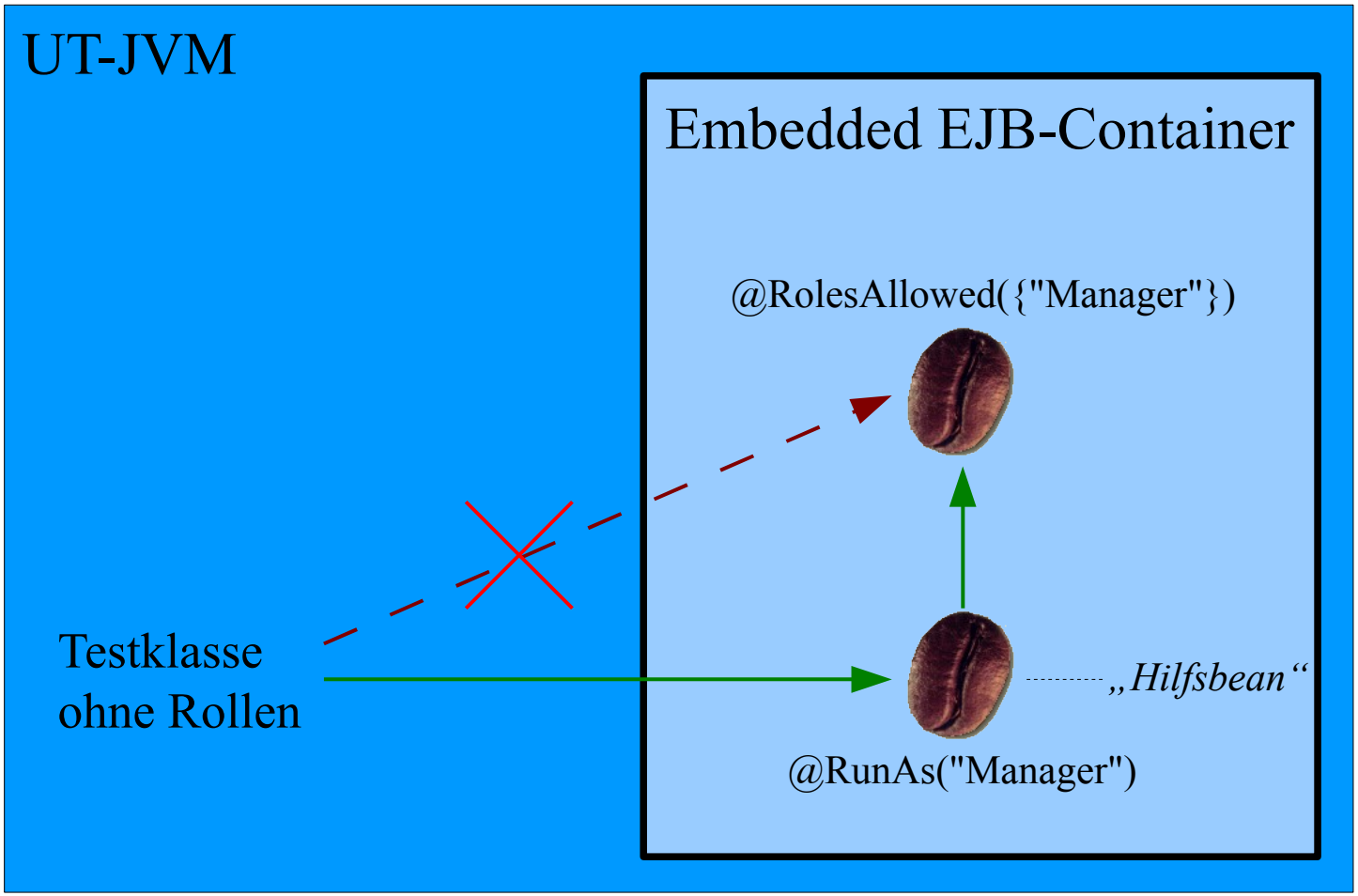
Testen von Security

```
@Stateless
public class ConfidentialBean  implements IConfidentialBean {
   @RolesAllowed({"Manager"})
   public float getSalary(Employee e) {
      float salary = 42f; // lookup in DB
      return salary;
   }
}
```

Anruf ohne Rolle:

javax.ejb.EJBAccessException: Unauthorized Access by Principal Denied

Testen von Security



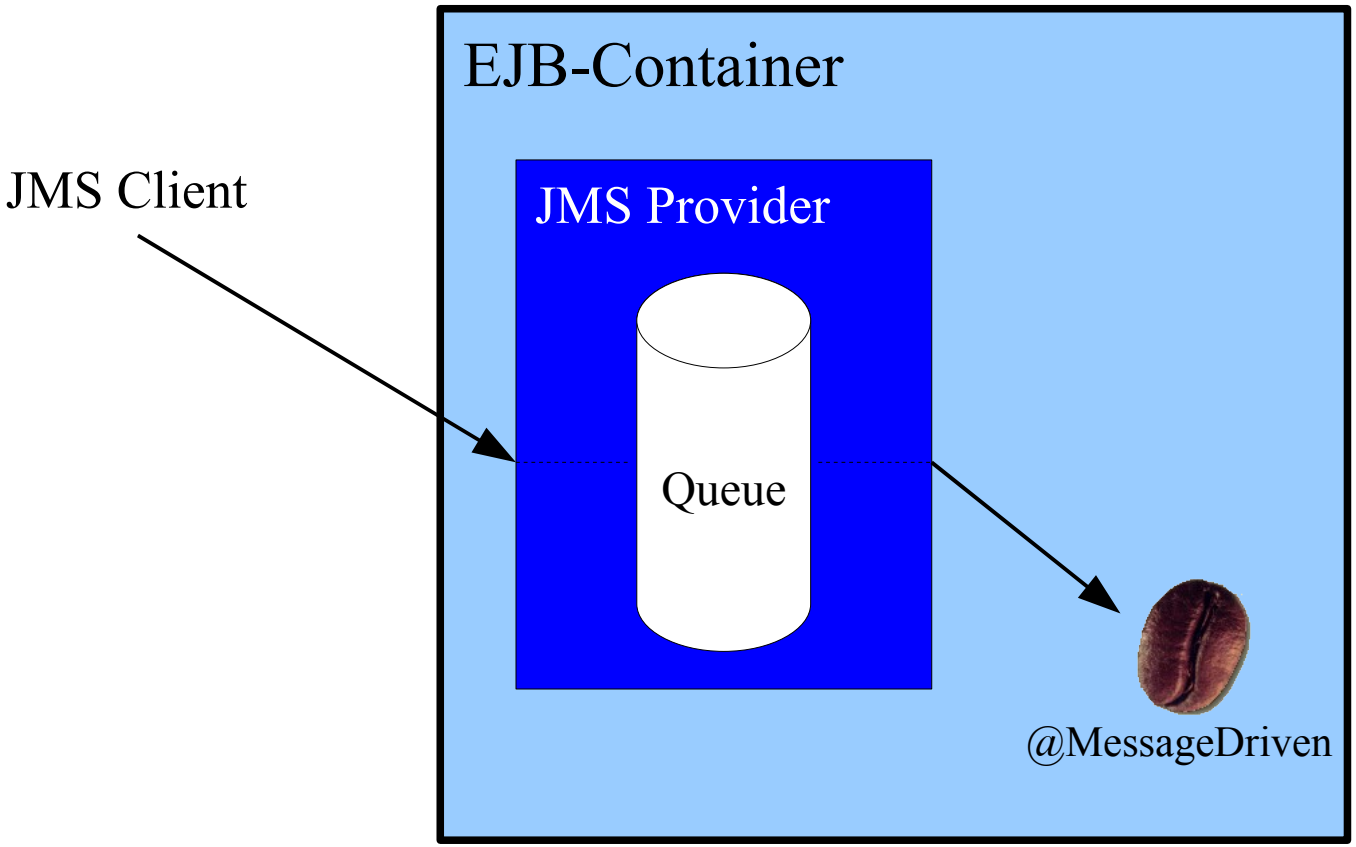
Testen von Security

```
@Local
public interface IManagerExecutor<V> {
   public <V> V call(Callable<V> callable) throws Exception;
}

@Stateless
@RunAs("Manager")
public class ManagerExecutor implements IManagerExecutor {
   public <V> V call(Callable<V> callable) throws Exception {
      return callable.call();
   }
}
```

Testen von Security

```
public class SalaryTest extends OpenEJBTestCase {
  @EJB ConfidentialBean confidentialBean;
  @EJB IManagerExecutor manager;

  @Test
  public void testAsManager() throws Exception {
    final Employee employee = getEmployee();
    Float salary = manager.call(new Callable<Float>() {
      public Float call() throws Exception {
        return confidentialBean.getSalary(employee);
      }
    });
  }}
```

Testen von @MessageDriven Beans



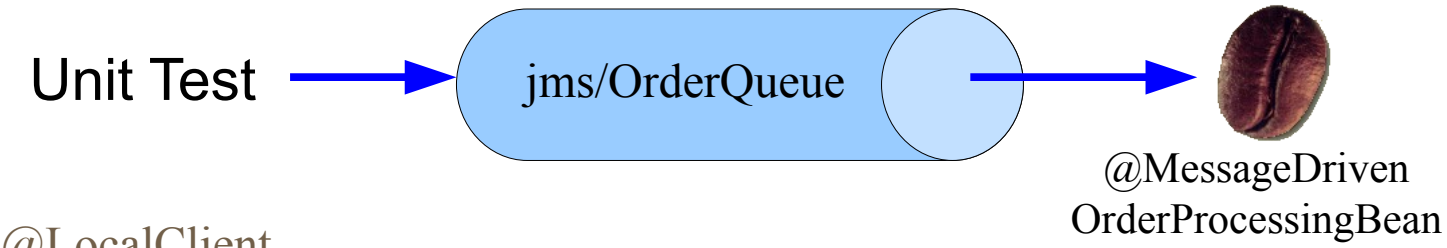
Testen von @MessageDriven Beans

```
@MessageDriven (activationConfig = {
        @ActivationConfigProperty(
          propertyName = "destinationType",
          propertyValue = "javax.jms.Queue"),
        @ActivationConfigProperty(
          propertyName = "destination",
          propertyValue = "jms/OrderQueue") })
public class OrderProcessingBean implements MessageListener {

  @TransactionAttribute(TransactionAttributeType.REQUIRED)
  public void onMessage(Message message) { //... }

}
```

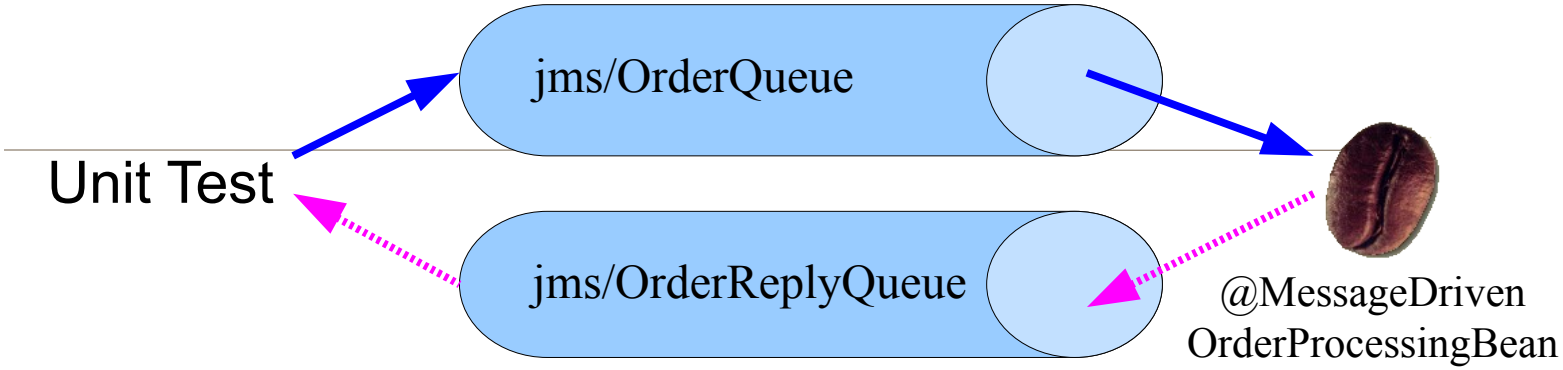
Testen von @MessageDriven Beans

```
Properties p = new Properties();
p.put(Context.INITIAL_CONTEXT_FACTORY,
      "org.apache.openejb.client.LocalInitialContextFactory");

p.put("Default JMS Resource Adapter",
      "new://Resource?type=ActiveMQResourceAdapter");
p.put("jms/QueueConnectionFactory",
      "new://Resource?type=QueueConnectionFactory");
p.put("jms/OrderQueue", "new://Resource?type=Queue");

InitialContext ctx = new InitialContext(p);
```

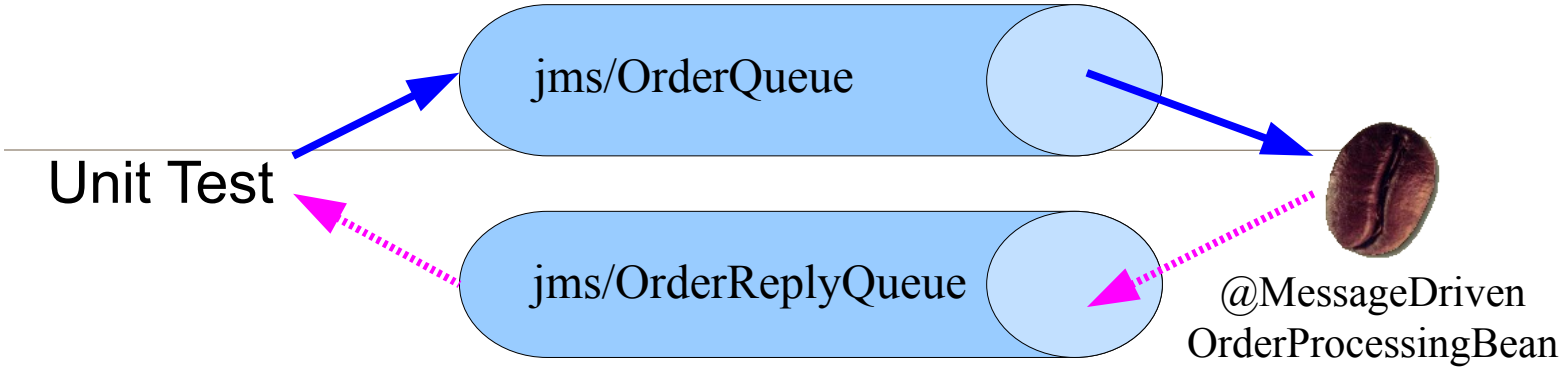
Testen von @MessageDriven Beans



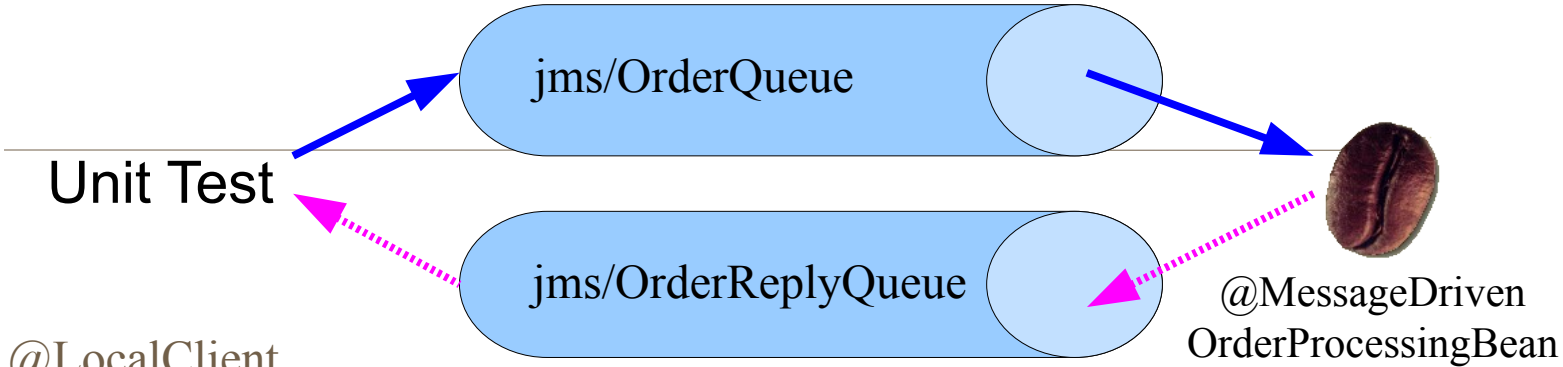

```
@LocalClient
public class JMSTest {

    @Resource(mappedName = "jms/OrderQueue")
    private Queue orderQueue;

    @Test
    public void testOrderProcessing() {
        // ...initialisiere Connection, Session und Producer
        TextMessage message = session.createTextMessage();
        message.setText(orderMessage);
        messageProducer.send(message);
    }

}
```




```
@LocalClient
public class JMSTest extends OpenEJBTest {
    @Resource(mappedName = "jms/OrderQueue")
     private Queue orderQueue;
    @Resource(mappedName = "jms/OrderReplyQueue")
     private Queue orderReplyQueue;
    @Test
    public void testOrderProcessing() {
          // ...initialisiere Connection, Session und Producer
          TextMessage message = session.createTextMessage();
          message.setText(orderMessage);
          message.setJMSReplyTo(orderReplyQueue);
          messageProducer.send(message);
    }
```




```
@MessageDriven
public class OrderProcessingBean implements MessageListener {
public void onMessage(Message message) {
    // ...
    if (textMessage.getJMSReplyTo() != null) {
        MessageProducer messageProducer = session
                .createProducer(textMessage.getJMSReplyTo());
        TextMessage replyMessage = session.createTextMessage();
        replyMessage.setText("Order accepted...");
        messageProducer.send(replyMessage);
    }
}
```




```
@LocalClient
public class JMSTest extends OpenEJBTest {
    @Test
    public void testOrderProcessing() {
        // ...
        message.setJMSReplyTo(orderReplyQueue);
        messageProducer.send(message);
        Message replyMessage = messageConsumer.receiveNoWait();
        int tries = 0;
        while (replyMessage == null && tries < 20) {
            Thread.sleep(100);
            tries++;
            replyMessage = messageConsumer.receiveNoWait();
        }
    }
}
```

OpenEJB: Beans als innere Klassen

```
@LocalClient
public class JMSTest extends OpenEJBTest {
    @EJB
    private ISupportBean support;
    @Test
    public void testSupport() {
        support.doSupportStuff("Hallo");
    }

    @Stateless
    public static class SupportBean implements ISupportBean {
        public String doSupportStuff(String s) {
            return s;
        }
    }
}
```

Agenda

- Einführung
  - Unit Tests, EJBs, Container, JUnit
- Mögliche Ansätze für Unit Tests
  - (Integrations-)Tests mit AppServer, Unmanaged, Embedded
- Embedded OpenEJB
  - Starten des Containers, Scannen nach Beans, JNDI, Injection
- Testansätze mit Embedded OpenEJB
  - Persistenz, Security, MDBs
- **Embedded Glassfish**

Embedded Glassfish

- Glassfish v3: https://glassfish.dev.java.net/
- RI für Java EE 5 und 6
- Embedded Glassfish: Bestandteil von v3
- Unterstützt EJB 3.1 und Embeddable API

Embedded Glassfish

```
Map<String, Object> p = new HashMap<String, Object>();
p.put(EJBContainer.MODULES,
        new File[] { new File("bin/src_prod"),
                     new File("bin/src_maint") });

EJBContainer container = EJBContainer.createEJBContainer(p);

Context ctx = container.getContext();
IEchoBean echoBean =
        (IEchoBean) ctx.lookup("de.mathema.bohnen.gf.IEchoBean");
```

Embedded Glassfish

```
Map<String, Object> p = new HashMap<String, Object>();
p.put(EJBContainer.MODULES,
        new File[] { new File("bin/src_prod"),
                     new File("bin/src_maint") });
p.put("org.glassfish.ejb.embedded.glassfish.configuration.file",
        "glassfish/domain.xml");

EJBContainer container = EJBContainer.createEJBContainer(p);

Context ctx = container.getContext();
IEchoBean echoBean =
        (IEchoBean) ctx.lookup("de.mathema.bohnen.gf.IEchoBean");
```

Embedded Glassfish

```
<domain>
   <applications />
   <resources>
        <jdbc-resource jndi-name="address-db" pool-name="APool"
             object-type="user" enabled="true" />
        <jdbc-connection-pool datasource-
classname="org.apache.derby.jdbc.EmbeddedDataSource"
             res-type="javax.sql.DataSource" name="oe-db-pool" ping="true">
             <property name="ConnectionAttributes" value="create=true" />
             <property name="DatabaseName" value="./target/unit-test" />
             <property name="Password" value="" />
             <property name="User" value="" />
        </jdbc-connection-pool>
   </resources>
</domain>
```

Überschrift

- Unterpunkt 1
- Unterpunkt 2
- ...

Vielen Dank!

Karol Rückschloss

MATHEMA Software GmbH