Hochschule Darmstadt
Fachbereich Informatik
J. Reichardt

ANLEITUNG
ZUM 3. PRAKTIKUMSVERSUCH VERTEILTE SYSTEME

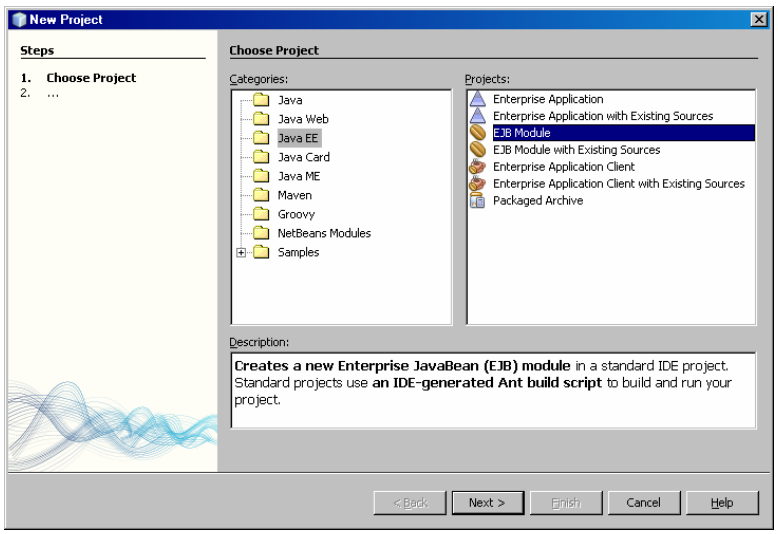
Publisher-Subscriber

mit JMS und Message Driven Beans

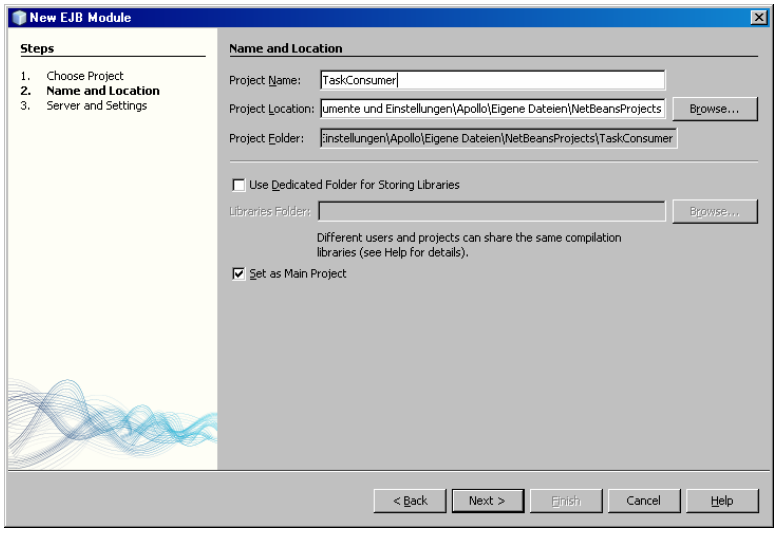
TEIL A: TaskBag-Topic mit Task-Producer und Task-Consumer erstellen

A.1 Task-Consumer erstellen

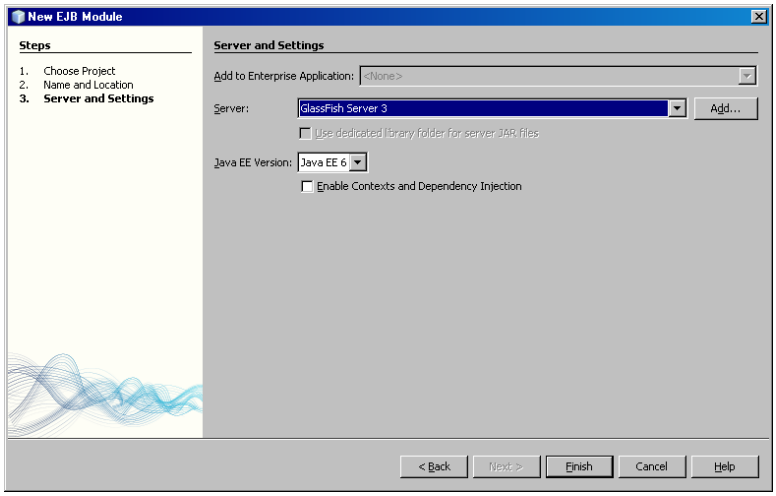
- **Projektdateien und –verzeichnisse erzeugen**
  - File -> New Project (*beachten: Java EE und EJB Module auswählen*)



  - Next

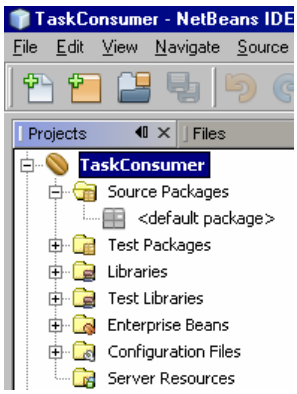


- Next

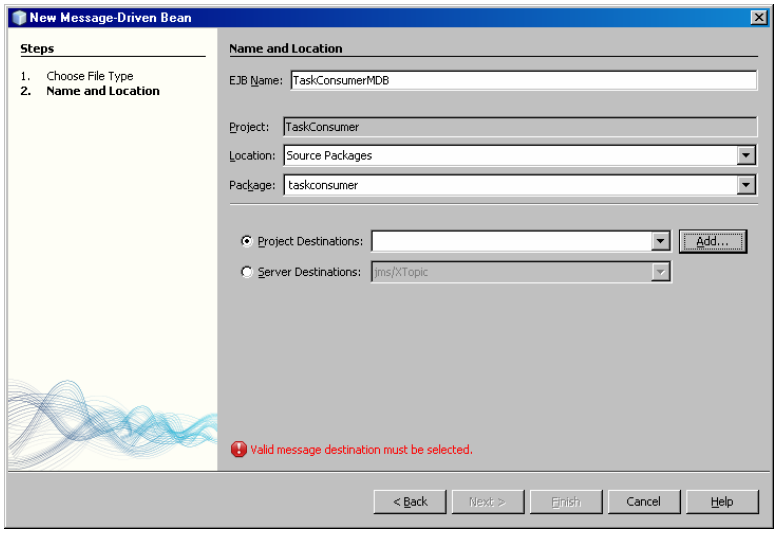


- Finish

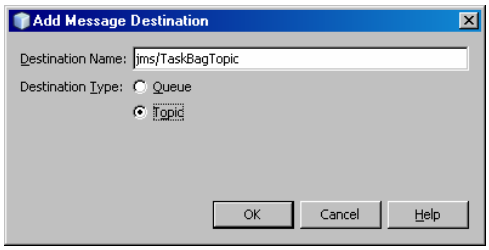
- Folgende Projektdateien und –verzeichnisse wurden erzeugt:



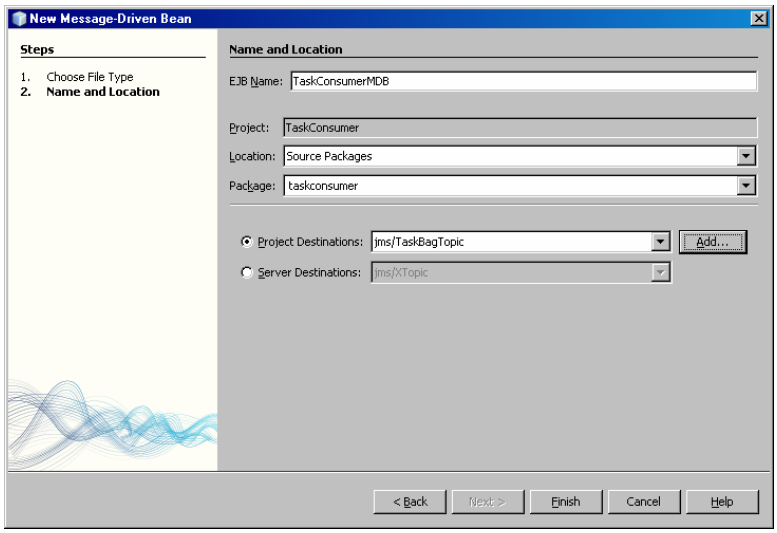
- **Message Driven Bean erzeugen**
  - Rechts-Click auf Ordner ‚Source Packages/<default package>' -> New -> **Message Driven Bean**



- Add Project Destination *(nur falls die Topic noch nicht existiert. Anderenfalls, z.B. wenn die Topic von anderen Consumern bereits benutzt wird: Topic unter ‚Server Destinations' auswählen)*



- OK



- Finish

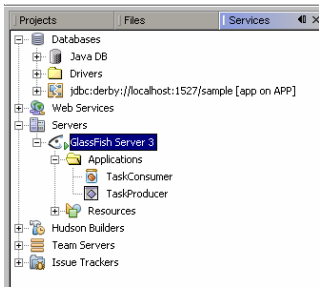
- Im Verzeichnis 'Source Packages/taskconsumer' wurde die Datei TaskConsumerMDB.java angelegt, mit folgendem Inhalt:

```
@MessageDriven(mappedName = "jms/TaskBagTopic", activationConfig =
{
    @ActivationConfigProperty(propertyName = "acknowledgeMode",
           propertyValue = "Auto-acknowledge"),
     @ActivationConfigProperty(propertyName = "destinationType",
           propertyValue = "javax.jms.Topic"),
     @ActivationConfigProperty(propertyName =  "subscription
           Durability", propertyValue = "Durable"),
     @ActivationConfigProperty(propertyName = "clientId",
           propertyValue = "TaskConsumerMDB"),
     @ActivationConfigProperty(propertyName = "subscriptionName",
           propertyValue = "TaskConsumerMDB") })

public class TaskConsumerMDB implements MessageListener {

 public TaskConsumerMDB() {
 }

 public void onMessage(Message message) {
 }
}
```

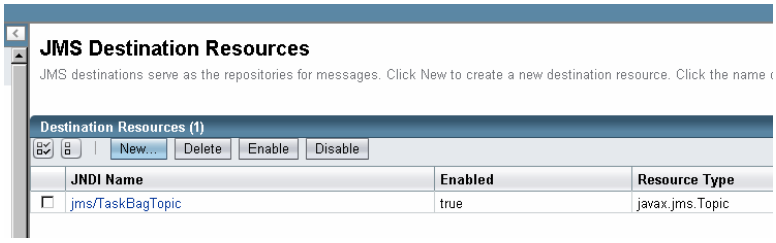
- Die Methode *onMessage(Message)* ausprogrammieren:
  - *Message* casten nach *TextMessage*.
  - Zugriff auf Text mittels *getText()*.
  - JMSException abfangen.

- **TaskConsumer bereitstellen**
  - Rechts-Click auf Ordner ‘TaskConsumer’ -> Deploy.

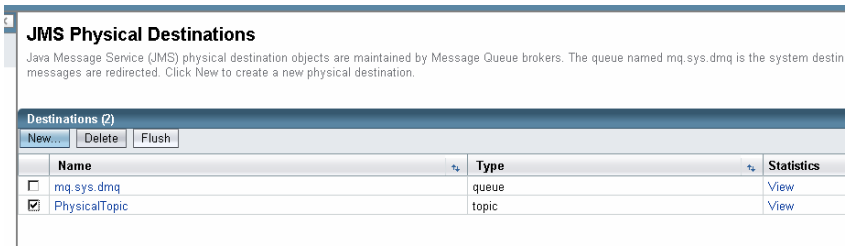
- **Admin-Konsole starten**
  - Rechts-Click auf ‚Services/Servers/GlassFish Server 3’ und ‚View Admin Console’ wählen.



  - Auf der Admin-Konsole ist unter ‚Resources/JMS Resources’ die während des Deploy-Vorgangs erzeugte TaskBag-Topic mit ihrem globalen JNDI-Namen zu sehen.



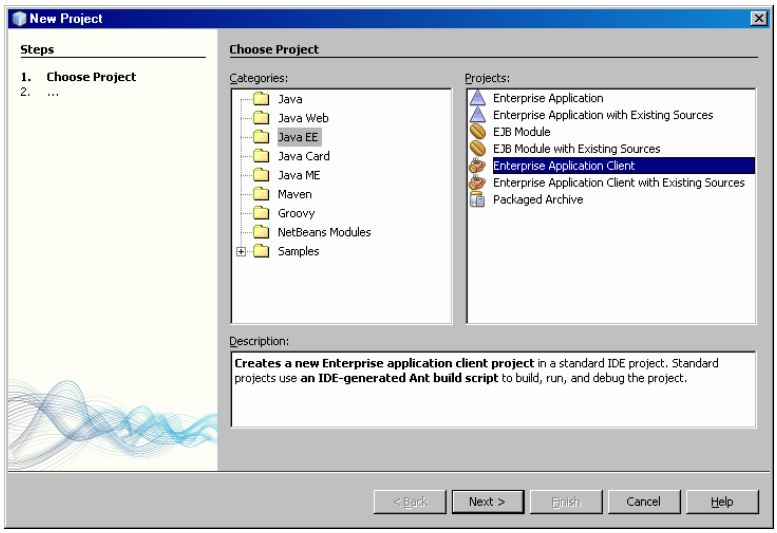
  - Auf der Admin-Konsole ist unter ‚Configuration/Java Message Service/Physical Destinations/Statistics/View’ eine Zustandsanzeige der Topic zu erhalten (Number of Consumers, Producers, Messages, Bytes, …).

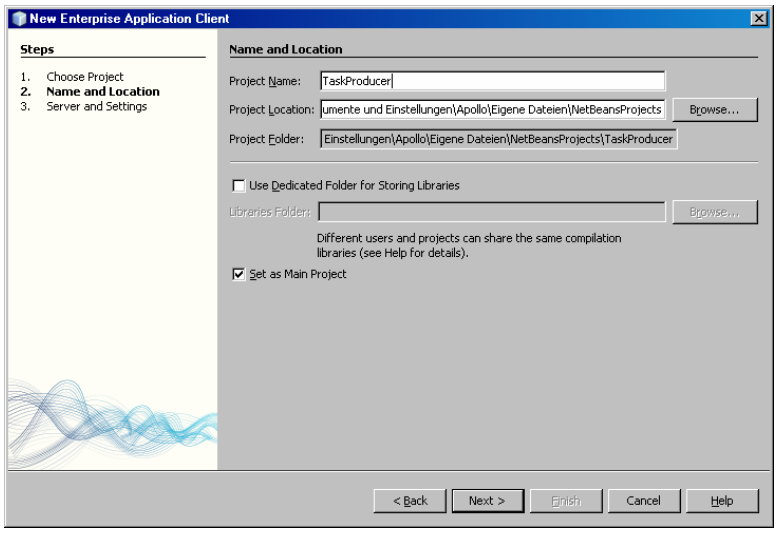


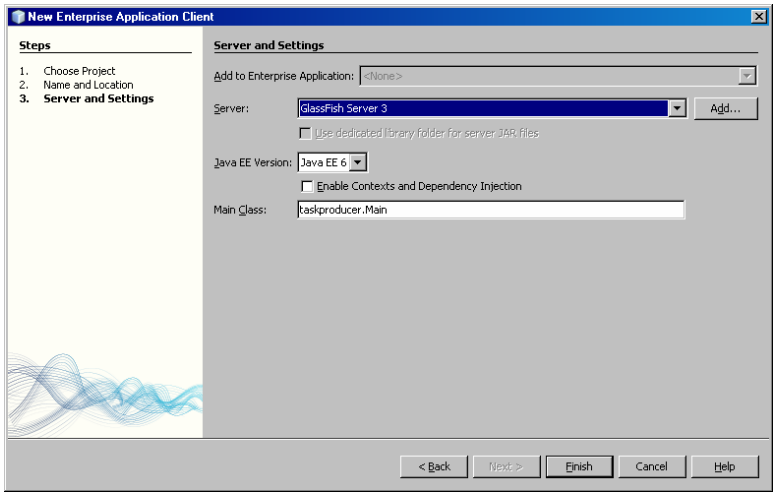
  - Bei Überlauf der Topic (Fehlermeldung „Topic akzeptiert keine weiteren Nachrichten“), beispielsweise durch Nichtauslieferung von Nachrichten aufgrund des Message Selektors in Teil B, kann die Topic mithilfe der *Flush*-Funktion wieder geleert werden.

A.2 Task-Producer erstellen

- **Projektdateien und –verzeichnisse erzeugen**
  - File -> New Project (*beachten: Java EE und Enterprise Application Client auswählen; für Injection*)

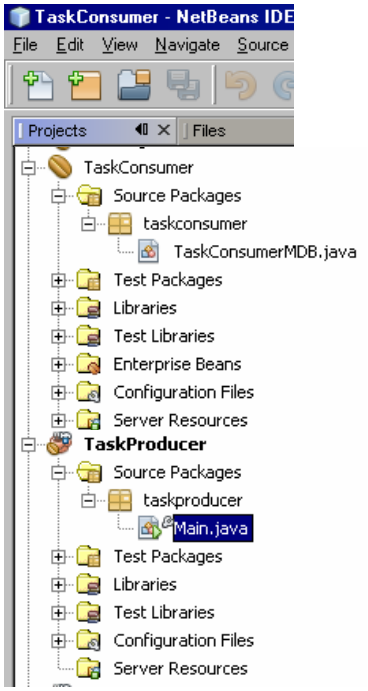




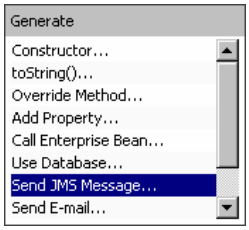


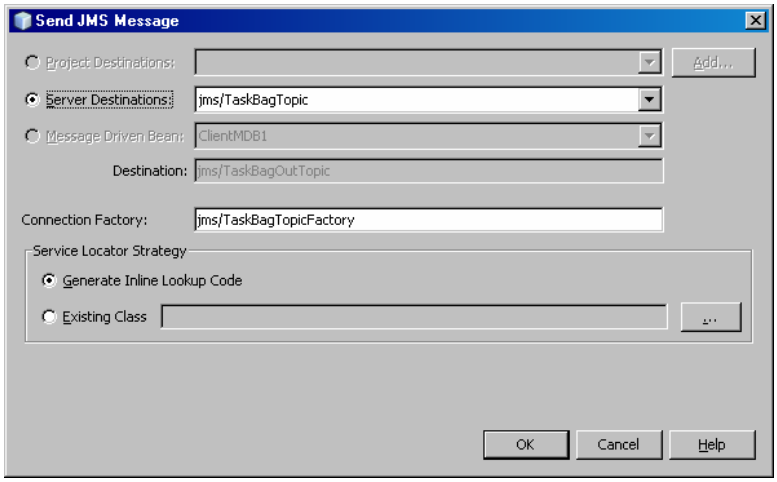
  - Next -> Next -> Finish

- Folgende Projektdateien und -verzeichnisse wurden bisher erzeugt:



- **JMS-Nachricht versenden**
  - Datei ‚Main.java' öffnen und Leerzeile in `main()` einfügen.
  - Rechts-Click in Leerzeile von `main()` -> Insert Code





  - OK

- Folgender Code wurde in der Klasse `Main` erzeugt:

```
public class Main {
      @Resource(name = "jms/TaskBagTopic")
private static Topic taskBagTopic;
      @Resource(name = "jms/TaskBagTopicFactory")
private static ConnectionFactory taskBagTopicFactory;

/**
* @param args the command line arguments
*/
public static void main(String[] args) {
      // TODO code application logic here

}

private Message createJMSMessageForjmsTaskBagTopic(Session session,
      Object messageData) throws JMSException {
      // TODO create and populate message to send
      TextMessage tm = session.createTextMessage();
      tm.setText(messageData.toString());
      return tm;
}

private void sendJMSMessageToTaskBagTopic(Object messageData) throws
            JMSException {
      Connection connection = null;
      Session session = null;
      try {
            connection = taskBagTopicFactory.createConnection();
            session = connection.createSession(false,
                        Session.AUTO_ACKNOWLEDGE);
            MessageProducer messageProducer =
                        session.createProducer(taskBagTopic);

            messageProducer.send(createJMSMessageForjmsTaskBagTopic
                        (session, messageData));
      } finally {
            if (session != null) {
                  try {
                        session.close();
                  } catch (JMSException e)
                        Logger.getLogger(this.getClass().getName()).
                        log(Level.WARNING, "Cannot close session",
                                    e);
            }} if (connection != null) {connection.close()}}}}
```

  - Erzeugten Code anpassen:
    - @Resource-Parameter `name` ersetzen durch `mappedName` (zwei mal).
    - Die Methoden **`sendJMSMessageToTaskBagTopic()`** und **`createJMSMessage-ForjmsTaskBagTopic()`** als `static` deklarieren (wegen `static main()`).
    - Den Aufruf des Loggers (nicht-`static`) auskommentieren (wegen `static main()`).
  - Methode **`sendJMSMessageToTaskBagTopic()`** in `main()` aufrufen:
    - Aktuellen Parameter `messageData` aus dem generierten Code bestimmen und eingeben.
    - JMSException abfangen.

- **Starten des Task-Producers**
  - Rechts-Click auf Ordner ‚TaskProducer' -> Run.

TEIL B: Tasks über einen Message Selector an mehrere Task-Consumer senden

B.1 Message Selector

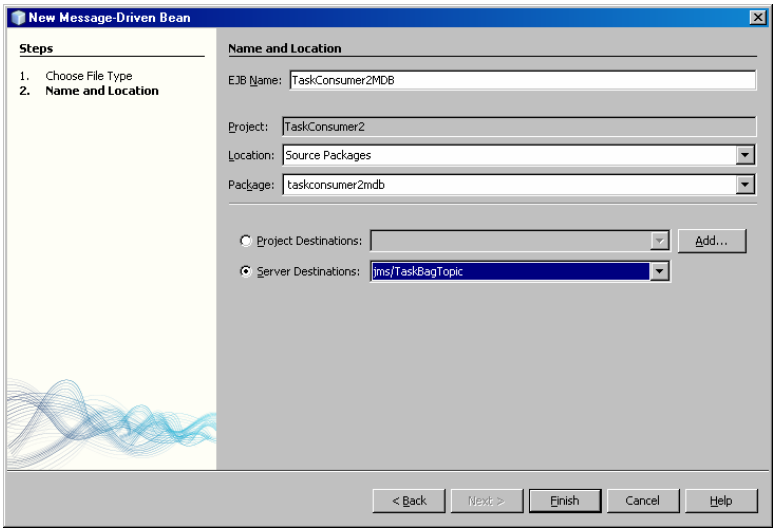
- **Task-Producer ergänzen**
  - Nach der Anweisung `setText()` ist die Anweisung `setStringProperty("taskType", "garten")` für die TextMessage auszuführen (mit frei gewähltem Selektortyp „taskType“ und frei gewähltem Selektorwert „garten“) .

  - Nach einem ersten Test mit nur einer Nachricht, ist der Task-Producer weiter zu ergänzen durch eine Nachrichten-Schleife. In jedem Schleifendurchlauf sollen Tasks mit vier verschiedenen Selektorwerten erzeugt und versendet werden. Der Nachrichtentext soll zusätzlich um eine fortlaufende Nummer ergänzt werden, die auf Producer- und Consumer-Seite wieder ausgedruckt wird.

- **Task-Consumer ergänzen**
  - In der @MessageDriven-Annotation ist eine zusätzliche ActivationConfigProperty aufzunehmen mit den Parametern `propertyName = „messageSelector“` und `propertyValue = „taskType = ‚garten‘“`.

  - Die Ausgabe der Textnachricht (mittels `getText()`) ist zu ergänzen um die Ausgabe des Selektorwertes mithilfe der Anweisung `getStringProperty(„taskType“)`.

- **Weitere Task-Consumer erstellen** *(mindestens einen weiteren Consumer)*
  - Projektdateien und –verzeichnisse erzeugen (wie A.1).
  - Message Driven Bean erzeugen (wie A.1, außer ‚Server Destination‘ auswählen):

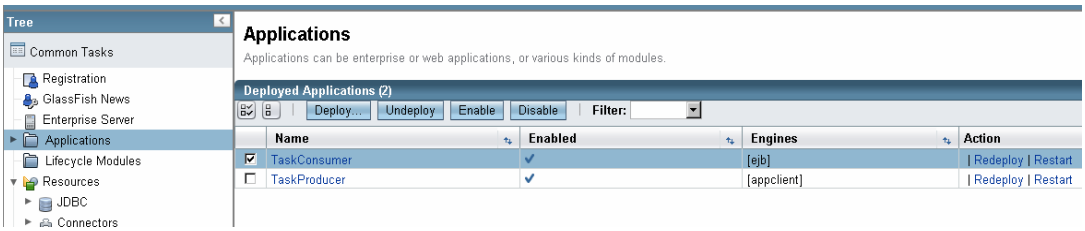


  - Finish.
  - Methode *onMessage()* der Message Driven Bean ausprogrammieren (wie A.1).
  - Neuen MessageSelector definieren (wie A.1). Dazu die Operatoren *OR*, *IN* und *LIKE* benutzen (siehe Tabelle der Aufgabenstellung, Seite 6).
  - Neuen Task-Consumer bereitstellen (wie A.1).

- **Tests**
  - Task-Producer mindestens 16 Nachrichten von 4 verschiedenen Task-Typen senden lassen.
  - Message-Selektoren aller Task-Consumer in verschiedenen Varianten durchspielen.
  - Auf der Admin-Konsole unter ‚Configuration/Java Message Service/Physical Destinations/Statistics/View‘ den Inhalt der Topic prüfen (siehe oben, Seite 4).

B.2 Abgekoppelte Task-Consumer

- **Abkoppeln aller Task-Consumer <u>mit</u> ActivationConfigProperty „Durable“**
  - Auf der Admin-Konsole unter ‚Applications’ die Task-Consumer (sicherheitshalber einzeln) auswählen und undeployen.



  - Starten des Task-Producers (wie A.2). Ausgabe von TaskProducer (run) beachten. Inhalt der Topic prüfen (siehe oben, Seite 4).

  - Deploy für Task-Consumer-1 (wie A.1). Ausgabe von GlassFish Server 3 beachten. Inhalt der Topic prüfen (siehe oben, Seite 4).

  - Deploy für Task-Consumer-2, usw. (wie oben).

- **Abkoppeln aller Task-Consumer <u>ohne</u> ActivationConfigProperty „Durable“**
  - In allen Task Consumers die Codezeile `@ActivationConfigProperty(propertyName = "subscriptionDurability", propertyValue = "Durable")` auskommentieren.

  - Danach Test wie oben durchführen und unterschiedliche Ausgaben beachten.