# Installationsanleitung X-Sharing Router

Max Wiederhold

30. Dezember 2015

## Grundsätzliches

Der X-Sharing Router ist konzipiert als ergänzende Maßnahme zur Integration von Individualverkehr in intermodales Routing. Dies geschieht in Form von *Vehicle Sharing* Systemen, welche im Kontext mit dem MobilityBroker Projekt eingebunden werden. Der Router ermöglicht in Folge das Stellen von Anfragen über eine definierte Schnittstelle auf Basis von Java Messaging Service (JMS) API. Die Implementierung erfolge in Java EE 7 auf dem Wildfly Application Server[1] in der Version 8.2. Daher die Empfehlung, mindestens diese Version zu verwenden. Alternativ sind aktuellere Releases möglich, jedoch eventuell nicht vollständig kompatibel. Getestet wurde Version 9.0.2.Final. Als Datenbank wird PostgreSQL verwendet.

## Vorbereitung des Wildfly Servers

Zur Vorbereitung der Server-Installation wird Wildfly in das Zielverzeichnis entpackt und anschließend die erforderlichen Libraries eingefügt.

### PostgreSQL

Um PostgreSQL als Datenbank zu verwenden, wird der entsprechende JDBC Treiber benötigt. Dieser ist zu finden unter `https://jdbc.postgresql.org/download.html` und ist auszuwählen entsprechend der PostgreSQL und JDBC Versionen. Da der X-Sharing Router PostGIS für geographische Daten verwendet, ist der entsprechende Treiber beigefügt unter
`etc/wildfly/org/postgresql/main/`
zu finden. Dort befindet sich ebenfalls eine Version des PostgreSQL-JDBC Treibers sowie die `module.xml`, welche die Integration seitens Wildfly übernimmt. Hier ist der Pfad an der Stelle `path="postgresql-9.3-1103.jdbc41.jar"` auf die vorher angesprochenen verwendeten Versionen anzupassen, indem der Dateiname des heruntergeladenen Treibers eingetragen wird. Der Ordner `postgresql` wird anschließend in das Verzeichnis

[1] http://wildfly.org/

`%WILDFLY%/modules/system/layers/base/org`
kopiert.

Weiterhin wird eine Anpassung der primären Wildfly Konfiguration vorgenommen. Um das JMS Subsystem im Standalone-Modus zu aktivieren, muss der Server in mit der `standalone-full.xml`-Konfiguration (zu finden unter `%WILDFLY%/standalone/configuration`) gestartet werden. Eine beispielhafte Datei ist im Verzeichnis zu finden. Diese ist wie folgt zu erweitern:

Im Subsystem `urn:jboss:domain:datasources` ist zunächst der Treiber zu registrieren, sowie die zugehörigen Datenquellen einzutragen.

Die mit `xSharingDS` bezeichnete Datenbank **muss** hierbei mit der PostgreSQL-Extension `PostGIS` ausgestattet sein! Außerdem ist zu beachten, dass für die Datasource des MobilityBroker Adapter JTA ausgeschaltet werden muss (siehe folgendes Listing).

```
<datasources>
  <datasource jta="true" jndi-name="java:jboss/
     datasources/xSharingDS" pool-name="xSharingDS"
     enabled="true" use-ccm="true">
    <connection-url>jdbc:postgresql://127.0.0.1:5432/
       xsharing</connection-url>
    <driver-class>org.postgresql.Driver</driver-class>
    <driver>postgresql</driver>
    <security>
      <user-name>xsharing</user-name>
      <password>***</password>
    </security>
    ...
  </datasource>
  <datasource jta="false" jndi-name="java:jboss/
     datasources/MBDS" pool-name="MBDS" enabled="true"
      use-ccm="true">
    ...
  </datasource>
  <driver name="postgresql" module="org.postgresql">
    <xa-datasource-class>org.postgresql.xa.
       PGXADataSource</xa-datasource-class>
  </driver>
  ...
</datasources>
```

## JMS

Um den Aufbau von JMS Verbindungen zu ermöglichen, müssen die in der Application verwendeten Queues registriert werden. Dies geschieht ebenfalls in der `standalone-full.xml`, im Subsystem `urn:jboss:domain:messaging`.

```
<hornetq-server>
  ...
  <jms-destinations>
    ...
    <jms-queue name="xSharingMinimalRequestQueue">
      <entry name="java:jboss/exported/jms/queue/
          xSharingMinimalRequestQueue"/>
    </jms-queue>
    <jms-queue name="xSharingCompactRequestQueue">
      <entry name="java:jboss/exported/jms/queue/
          xSharingCompactRequestQueue"/>
    </jms-queue>
    <jms-queue name="xSharingDetailsRequestQueue">
      <entry name="java:jboss/exported/jms/queue/
          xSharingDetailsRequestQueue"/>
    </jms-queue>
  </jms-destinations>
</hornetq-server>
```

Für den Zugriff von außerhalb muss außerdem das Interface `public` anzupassen auf die entsprechend erlaubten Zugänge. An dieser Stelle lassen sich außerdem noch sicherheitskritische Details wie Userverwaltung und Gruppezugehörigkeiten einbinden. Um Authentifizerung zu ermöglichen, müssen über das Wildfly-CLI entsprechende Application-User angelegt werden.

## Hibernate

Die Datenbankintegration von X-Sharing erfolgt mittels Hibernate und der Erweiterung Hibernate Spatial. Um die korrekte Interaktion zu gewährleisten, müssen die unter `etc/wildfly/org/hibernate/main` zu findenden Libaries in das Verzeichnis `%WILDFLY$/modules/system/layers/base/org/hibernate/main` kopiert werden und die `module.xml` angepasst werden. Dazu ist unter den Dependencies der Eintrag
`<module name="org.postgresql"/>`
hinzuzufügen, sowie die kopierten Dateien unter den Resources als `resource-root` einzutragen. Zusätzlich ist im selben Verzeichnis eine beispielhafte module.xml angefügt.

## Sonstiges

Im Falle von schwächerer Systemhardware oder sehr großer Datenmengen können der initiale Ladevorgang oder lange andauernde Berechnungen die standardmäßigen Transaktions-Timeouts überschreiten. Daher empfielt es sich nötigenfalls, im Subsystem `urn:jboss:domain:transactions` den Eintrag
`<coordinator-environment default-timeout="300"/>`
hinzuzufügen und den Wert (in Sekunden) entsprechend anzupassen.

## Deployment

Die X-Sharing Applikation beginnt nach dem Start automatisch mit dem Import der Daten und startet anschließend die Vorberechnung der Teilrouten. Daher ist eine korrekte Anbindung der Datenquellen vorher zu gewährleisten. Mit Hilfe des Wildfly-Maven-Plugins kann ein Deployment auf die in der POM angegebene location erfolgen. Dazu sind die Goals `clean package wildfly:deploy` auszuführen.