# ediarum.DB: Setup und erweiterte Einrichtung

## Wiederherstellung

Ist die Datenbank korrupt oder soll sie auf null zurückgesetzt werden, d.h. alle Daten,
Benutzerkonten (inkl. des *admin*) und installierte Apps gelöscht werden,
reicht es nach dem Stoppen der Datenbank den separaten Daten-Ordner auf dem Server zu löschen.
Mit einem erneuten Starten der Datenbank erhält man eine leere Datenbank
und kann dort wieder neue Apps installieren oder Daten einspielen. Zuerst sollte man
allerdings das *admin*-Passwort neu setzen.

## Scheduler

In der Datei `conf.xml` im eXist-Installationsordner können
verschiedene Einstellungen verändert werden. Unter `<scheduler>` können wiederkehrende Routinen eingerichtet
werden, (s.a. Scheduler Module in der eXist-Dokumentation). Dazu muss
dort die Zeile `<module uri="http://exist-db.org/xquery/scheduler" class="org.exist.xquery.modules.scheduler.SchedulerModule" />` entkommentiert sein.
Die Einstellungen für ein vierstündliches inkrementelles Backup mit täglichem Vollbackup sehen etwa so aus:

```
<job type="system" name="backup"
    class="org.exist.storage.ConsistencyCheckTask"
    cron-trigger="0 0 0/4 * * ?">
        <parameter name="output" value="path/to/backup/directory"/>
        <parameter name="backup" value="yes"/>
        <parameter name="incremental" value="yes"/>
        <parameter name="incremental-check" value="no"/>
        <parameter name="max" value="6"/>
</job>
```

Die Einrichtung eines Scheduler-Jobs zur Ausführung eines in der Datenbank liegenden xQuerys sieht etwa folgendermaßen aus (dabei können Parameter übergeben werden, die im XQuery als externe Variablen eingelesen werden):

```
<job type="user" name="validation"
    xquery="/db/path/to/xquery.xql"
    cron-trigger="0 5 1 * * ?">
        <parameter name="username" value="username"/>
        <parameter name="password" value="pass"/>
</job>
```

## Port-Konfiguration

Um den Port der Datenbank zu verändern (s. a. Port-Konflikte),
muss in der Datei `tools/jetty/etc/jetty.xml` im eXist-Installationsordner in der Zeile

```
<Set name="port"><SystemProperty name="jetty.port" default="8080"/></Set>
```

die Portnummer 8080 auf die gewünschte Portnummer eingestellt werden. Der Port 8443 für sichere Verbindungen kann in der gleichen Datei in den beiden Zeilen

```
<Set name="confidentialPort"><SystemProperty name="jetty.port.ssl" default="8443"/></Set>
```

```
<Set name="Port"><SystemProperty name="jetty.port.ssl" default="8443"/></Set>
```

in den gewünschten Port geändert werden.

## Ordnerstruktur und Berechtigungen

Damit die Datenbank nur für angemeldete Benutzer zugänglich ist, müssen die Berechtigungen der einzelnen Dateien korrekt gesetzt sein. Die Standard-Einstellung von eXist sieht einen Lesezugriff auch für Gäste vor, was hier aber insbesondere für die Projektverzeichnisse und das dortige `data`-Verzeichnis explizit nicht gewünscht ist.

Gewünscht ist ein Zugriff nur für angemeldete Benutzer, d.h. nach dem Berechtigungsschema von eXist `-rwxrwx--- (group: PROJECT-nutzer)` für alle Dateien im Ordner `/db/projects/PROJECT/data`. Die Ordner `data`, `data/Dokumente` und `data/Register` werden mit der Berechtigung `crwxrwx--- (owner: admin, group:PROJECT-nutzer)` erstellt. Neue Benutzer im Projekt werden mit der `umask: 0007` angelegt, wodurch von ihnen angelegte Dateien automatisch die Berechtigung `rwxrwx---` bekommen.

Beim Anlegen eines neuen Projektes wird das Projekt im Verzeichnis `/db/projects/PROJECTNAME` erstellt. Dabei wird darunter folgende Ordnerstruktur angelegt:

- `data`: Enthält die Forschungsdaten. Im Ordner `data/Register` werden die notwendigen Registerdateien gespeichert.
- `druck`: Hier können die notwendigen Dateien gespeichert werden, um den Druckvorgang auf dem Server anzustoßen.
- `exist`: Enthält die Routinen (`exist/routinen`), die von der Datenbank angesteuert werden können.
- `oxygen`: Enthält die Schnittstellen, die für den Zugriff von Oxygen notwendig sind. Insbesondere für das Auslesen der aktuellen Register.
- `web`: Hier können die notwendigen Dateien (xQuery-Skripte, CSS, etc.) für eine Webpräsentationen gespeichert werden.

Folgende Tabelle gibt einen Überblick über die Berechtigungen in den Ordnern eines ediarum-Projektes:

| Verzeichnis / Nutzergruppe | /data | /exist | /oxygen | /website | Beschreibung |
|---|---|---|---|---|---|
| admin | X | X | X | X | Administrator mit allen Berechtigungen. |
| PROJECT-nutzer | X | | | | Nutzer in einem ediarum-Projekt |
| exist-bot | X | X | X | X | Führt die Routinen aus. |
| oxygen-bot | | | X | | Für Oxygen in globale Optionen und im CSS, liest aus data_copy. |
| website-user | | | | X | Mit Zugriff auf die Webseite und den Druck. |

## Routinen und xQuery-Skripte

xQuery-Skripte werden im Ordner `exist/routinen` abgelegt. Zu Testzwecken existiert dort schon das Skript `test.xql`, an welchem Ausgaben und Funktionen ausprobiert werden können. Im **ediarum**-Modul, welches in der Datei `modules/ediarum.xql` in der **ediarum.DB**-App liegt, sind zentrale Funktionen vordefiniert, die in die Skript über die Zeile

```
import module namespace ediarum="http://www.bbaw.de/telota/software/ediarum/ediarum-app" at "/db/apps/ediarum/modules/edia
```

in das aktuelle Skript geladen w erden können.

Soll das Skript mit erw eiterten Berechtigungen ausgeführt w erden können, müssen die Rechte entsprechend gesetzt w erden (etw a: rw xr-sr-x, mit 'dba' als Gruppe).

## Trigger

Für eXist-db können verschiedene Trigger eingerichtet w erden, die bei bestimmten Ereignissen ausgeführt w erden, etw a w enn Dateien in einem bestimmten Ordner erstellt oder verändert w erden (s. auch Trigger).

Die Trigger sind praktisch, falls man Daten in verschiedenen Dateien verw altet, die aber auch gemeinsam benutzt w erden sollen. Dies ist z. B. bei Registern der Fall, w o jeder Registereintrag in einer eigenen Datei gespeichert w ird. Die verschiedenen Dateien lassen sich durch einen Trigger mit einem xQuery zusammenführen und separat speichern. Ein anderes Beispiel sind Dateien auf die auch ein Lesezugriff existieren soll, w enn an ihnen gearbeitet w ird. Auch hier lassen sich die Dateien durch einen Trigger und entsprechendes xQuery-Skript separat speichern.

Trigger für den Ordner `/db/mein/Pfad` und dessen Subordner lassen sich einrichten, indem im Ordner `/db/system/config/db/mein/Pfad` eine Datei `collection.xconf` abgelegt w ird, die etw a auf eine xQuery-Datei verw eisen kann. Diese muss zum Trigger-Namespace gehören und kann verschiedene Aktionen definieren (s. xQuery-Funktionen). Die Einrichtung w ird im folgenden an einem Beispiel erläutert.

Beispiel: Jeder Registereintrag eines gemeinsamen Register liegt in einer eigenen Datei.
Nach der Erstellung eines Registereintrags (d.h. einer neuen Datei im Ordner Register) sollen die Berechtigungen für diese Datei neu gesetzt w erden.

Im Ordner `/db/system/config/db/projects/PROJECT/Register` w ird eine Datei `collection.xconf` mit folgendem Inhalt gespeichert:

```
<collection xmlns="http://exist-db.org/collection-config/1.0">
    <triggers>
        <trigger class="org.exist.collections.triggers.XQueryTrigger">
            <parameter name="url" value="xmldb:exist//db/apps/ediarum/routinen/set-permissions-for-document.xql"/>
        </trigger>
    </triggers>
</collection>
```

Die Datei `/db/apps/ediarum/routinen/set-permissions-for-document.xql` gehört zum Namespace trigger und definiert einen Trigger, der die gew ünschten Aktion durchführt:

```
xquery version "3.0";

module namespace trigger="http://exist-db.org/xquery/trigger";

declare namespace xmldb="http://exist-db.org/xquery/xmldb";
declare namespace sm="http://exist-db.org/xquery/securitymanager";

declare variable $local:group-name external;

declare function trigger:after-create-document($uri as xs:anyURI) {
    let $chmod := sm:chmod($uri, "rwxrwx---")
    let $chgrp := sm:chgrp($uri, $local:group-name)
    return ()
 };
```

Die auszuführenden xQueries müssen über die im Trigger angegebene URL erreichbar sein, d.h. sie müssen auch mit Gastrechten ausführbar sein.