independIT Integrative Technologies GmbH
Bergstraße 6
D–86529 Schrobenhausen

schedulix

Installationshandbuch Release 2.9

Dieter Stubler Ronald Jeninga

12. Mai 2022

Inhaltsverzeichnis

Kapitel 1

Voraussetzungen

Compile Umgebung

Um aus dem Source Paket auf einem Linux System die benötigten Executables zu erstellen, wird folgende Software benötigt:

- Oracle(Sun) Java 1.7 JDK oder höher
  http://www.oracle.com/technetwork/java/index.html
  Alternativ dazu ein OpenJDK 1.7 oder höher
  http://openjdk.java.net
- gcc, gcc-c++
  http://gcc.gnu.org
- gnu make
  http://www.gnu.org/software/make
- jflex (Version 1.4.x)
  http://jflex.de
- jay
  Das jay Executable wird zwar im Paket mitgeliefert, aber einen Hinweis auf die Originalsourcen bzw. -executables sollte hier nicht fehlen.
  http://www.cs.rit.edu/~ats/projects/lp/doc/jay/package-summary.html
- Eclipse SWT
  schedulix enthält einige Beispiele, die ein installiertes SWT voraussetzen. Damit der Compile nicht abbricht wird ein Eclipse-SWT benötigt.
  http://www.eclipse.org/swt
- Java Native Access (JNA)
  Um die Notwendigkeit einer JNI Library zu umgehen wird ab der Version 2.6

die JNA Bibliothek genutzt.
https://github.com/twall/jna

In vielen Fällen können die benötigte Software Pakete einfach über ein Package Manager wie yum, rpm oder dpkg installiert werden.

schedulix Server

Zur Installation des schedulix Servers wird folgende Software benötigt:

- Ein in einer Compile Umgebung für dieselbe Zielarchitektur erzeugtes schedulix-2.9.tgz
- Oracle(Sun) Java 1.7 SE jre
  http://www.oracle.com/technetwork/java/index.html
  Alternativ dazu ein OpenJDK 1.7 oder höher
  http://openjdk.java.net
- Eines der folgenden RDBMS Systeme mit zugehörigem JDBC Interface
  - PostgreSQL

    http://www.postgresql.org

    JDBC für PostgreSQL:
    http://jdbc.postgresql.org
  - MySQL

    http://www.mysql.com

    MySQL (Connector/J) JDBC Driver
    http://www.mysql.com
  - Ingres

    http://www.ingres.com
- Eclipse SWT
  schedulix enthält einige Beispiele, die ein installiertes SWT voraussetzen. Sollen diese Beispiele installiert werden, wird ein Eclipse-SWT benötigt.
  http://www.eclipse.org/swt
  Andernfalls ist ein SWT nicht erforderlich.

schedulix Client

Zur Installation eines schedulix Clients wird folgende Software benötigt:

- Ein in einer Compile Umgebung für dieselbe Zielarchitektur erzeugtes schedulix-2.9.tgz

- Oracle(Sun) Java 1.7 SE jre
  http://www.oracle.com/technetwork/java/index.html
  Alternativ dazu ein OpenJDK 1.7 oder höher
  http://openjdk.java.net

- Eclipse SWT
  schedulix enthält einige Beispiele, die ein installiertes SWT voraussetzen. Sollen diese Beispiele auch auf dem Client ausgeführt werden können, wird ein Eclipse-SWT benötigt.
  http://www.eclipse.org/swt
  Andernfalls ist ein SWT nicht erforderlich.

- Java Native Access (JNA)
  Um die Notwendigkeit einer JNI Library zu umgehen wird ab der Version 2.6 die JNA Bibliothek genutzt. Diese Biblithek wird nur für den Jobserver benötigt.
  https://github.com/twall/jna

Zope Application Server

Das Web Frontend wird von dem Zope Application Server bereitgestellt. Zur Installation des Zope Servers wird folgende Software benötigt:

- Python 2.7
  http://www.python.org

- Python development package (python-devel ode python-dev)
  http://www.python.org

- python-setuptools
  http://pypi.python.org

Kapitel 2

Compile des Systems

Generelle Vorbereitung

Es ist sinvoll eine Software mit zentraler Bedeutung unter einem eigenen Account zu installieren. Dies vereinfacht die Administration und schützt gegen Missbrauch.
Im nachfolgenden wird davon ausgegangen, dass die Umwandlung und Installation unter dem Account `schedulix` erfolgt. Als Home-Verzeichnis wird `/home/schedulix` angenommen. Selbstverständlich sind dies alles nur Vorschläge. Es gibt keine technische Notwendigkeit diese Vorschläge an zu nehmen. Allerdings muss die Anleitung bei Änderungen entsprechend interpretiert werden.
Wie ein Benutzer angelegt werden kann steht, im Installationskapitel auf Seite 7 dokumentiert.

Compile

Um nach der Installation der benötigten Pakete das System erfolgreich zu übersetzen, müssen noch einige Umgebungsvariablen gesetzt werden bevor "make" die eigentlichen Arbeit machen kann.
Da weder für das Umwandeln als auch für die Installation an sich keine besondere Rechte benötigt werden, wird unter dem User `schedulix` gearbeitet.

1. Download der schedulix Source Distribution von github
   Alle zur Übersetzung und Installation notwendigen Dateien stehen in einem github Repository zur Verfügung und können mit folgendem Kommando heruntergeladen werden:

   ```
   cd $HOME
   git clone https://github.com/schedulix/schedulix.git
      -b v2.9 schedulix-2.9
   ```

   Danach befinden sich alle Dateien der schedulix Source Distribution im Unterverzeichnis:

```
$HOME/schedulix-2.9
```

2. Setzen der Umgebungsvariablen
   Nun müssen einige Umgebungsvariablen gesetzt werden. Die folgende Kommandos stammen von einer Installation auf einer CentOS Linux Distribution (http://www.centos.org). In vielen Fällen können die Befehle eins-zu-eins übernommen werden, aber sie sind abhängig von der genauen Linux Distribution und installierten Software.

   ```
   export SDMSHOME=/home/schedulix/schedulix-2.9
   export CLASSPATH=$CLASSPATH:/usr/share/java/jflex.jar
   export JAVAHOME=/usr/lib/jvm/java-1.7.0-openjdk-1.7.0.0
   export SWTJAR=/usr/lib/java/swt.jar
   export JNAJAR=/usr/share/java/jna.jar
   ```

   Es empfiehlt sich diese Einstellungen in die `.bashrc` auf zu nehmen, zumindest so lange das Umwandeln des Systems noch nicht erledigt ist.

3. make
   Es bleibt jetzt nur noch das tatsächliche Umwandeln des Systems übrig.

   ```
   cd ~/schedulix-2.9/src
   make
   ```

   Bei einem wiederholten Versuch das System um zu wandeln, empfiehlt es sich statt `make make new` ein zu geben.

   Als letzte Aktion wird ein jar-File erzeugt und unter `/schedulix-2.9/lib` abgelegt.

4. Erzeugen `/schedulix-2.9.tgz`

   ```
   cd $HOME
   tar czf schedulix-2.9.tgz schedulix-2.9
   ```

Kapitel 3

Installation in einer Linux-Umgebung

Installation des schedulix Servers

Die Installation des schedulix Scheduling Servers ist einfach. Es bedarf nur einiger Handlungen die im Folgenden erläutert werden:
Wenn (Beispiel-)Kommandos vorgestellt werden, wird als Prompt normalerweise ein `$` gezeigt. Diese Kommandos werden dann unter dem neu anzulegenden Account `schedulix` ausgeführt. In einigen Fällen wird der priviligierte Account `root` benötigt. Dies wird dadurch gekennzeichnet, dass als Prompt ein `#` gezeigt wird.

1. User `schedulix` anlegen
   Es gibt keine Notwendigkeit den User schedulix zu nennen. Damit kann der Name auch einer beliebigen Konvention angepasst werden. In diesem Dokument wird davon ausgegangen, dass der User schedulix heißt.

   Unter Ubuntu Linux kann ein User folgendermaßen angelegt werden:

   ```
   # useradd -d /home/schedulix -m -s /bin/bash -U schedulix
   # passwd schedulix
   ```

   Alle nachfolgenden Aktionen werden unter User schedulix ausgeführt, es sei denn es wird explizit anders angegeben.

2. Herunterladen und Installieren eines von schedulix unterstützten Datenbank Management Systems.

   schedulix für Linux unterstützt derzeit die Systeme:

   - Postgres (Seite 15)
   - MySQL (Seite 16)
   - Ingres (Seite 18)

Für die Installation des gewählten Datenbanksystems, sowie die Anpassung der Konfiguration des schedulix Enterprise Scheduling Systems, wird auf die entsprechenden nachfolgenden Abschnitte verwiesen.

3. Software auspacken

   tar-Archiv auspacken im Homeverzeichnis von schedulix. Etwa:

   ```
   $ tar xvzf schedulix-2.9.tgz
   ```

   Symbolic Link anlegen:

   ```
   $ ln -s schedulix-2.9 schedulix
   ```

4. Konfiguration erstellen

   a) Benutzerumgebung

   Um mit dem schedulix System arbeiten zu können, müssen folgende Variablen gesetzt werden:

   ```
   BICSUITEHOME=/home/schedulix/schedulix
   BICSUITECONFIG=/home/schedulix/etc
   PATH=$BICSUITEHOME/bin:$PATH
   SWTJAR=/usr/lib/java/swt.jar
   JNAJAR=/usr/share/java/jna.jar
   ```

   Es hat sich in der Praxis als vorteilhaft erwiesen die Konfiguration des Systems außerhalb des Installationsverzeichnisses zu legen. Damit werden spätere Upgrades wesentlich erleichtert. Da die Variablen von allen Benutzern des Systems gesetzt werden müssen, kann es sinnvoll sein die Zuweisungen (und Exports) in einer eigenen Datei zu schreiben, und diese dann im `.profile` oder `.bashrc` zu sourcen.

   b) Softwareumgebung

   Unter `$BICSUITEHOME/etc` liegen einige Vorlagen für Konfigurationsdateien, die als Basis für die Systemkonfiguration verwendet werden sollten. Diese müssen dazu ohne die Endung ".template" ins Konfigurationsverzeichnis `$BICSUITECONFIG` kopiert werden.

   Etwa

   ```
   $ cd $BICSUITEHOME/etc; for fff in *.template; do
   > TRG=`basename $fff .template`;
   > cp $fff $BICSUITECONFIG/$TRG;
   > done
   ```

   Anschließend müssen die Dateien natürlich der Umgebung angepasst werden.

   Die Datei `bicsuite.conf` setzt einige Default-Einstellungen und muss im Allgemeinen nicht angepasst werden. Allerdings kann man sich überlegen das Logging des Systems außerhalb des Installationsverzeichnisses stattfinden zu lassen. In diesem Fall muss lediglich die Variable

BICSUITELOGDIR entsprechend angepasst werden. Das in BICSUITELOGDIR gesetzte Verzeichnis muss vorhanden sein.

Die Datei java.conf beschreibt die zu verwendende Java-Umgebung. Insbesondere muss der Pfad zum JDBC-Treiber eingegeben werden. Weiterhin wird die Speicherkonfiguration des Servers geregelt. Dazu muss, auch in großen Umgebungen, normalerweise nur die Variable BICSUITEMEM angepasst werden.

Die Datei server.conf enthält die Serverkonfiguration. Angepasst werden müssen hier die Einstellungen für die Verbindung des schedulix Scheduling Serves zu seinem RDBMS Repository. Mehr dazu finden Sie im jeweiligen Kapitel zum eingesetzten RDBMS.
Weiterhin muss in dieser Datei das Property hostname auf den Hostnamen oder die IP-Adresse des Servers gesetzt werden.

Die Datei jobserver.conf wird hier nicht benötigt, dient aber als Vorlage für die Jobserver-Konfiguration.

5. Datenbank einrichten

   Abhängig davon welches Datenbanksystem Sie nutzen möchten, befolgen Sie die Anleitung zur Einrichtung der Datenbank.

   Für

   - Ingres, siehe Seite 18,
   - MySQL, siehe Seite 16, und für
   - PostgreSQL, siehe Seite 15.

6. Server hochfahren

   Die Installation ist nun im Wesentlichen abgeschlossen. Was noch bleibt ist das Starten des Servers und, bei Bedarf, das Einspielen der Beispiele.

   Der Server kann mittels

   ```
   $ server-start
   ```

   gestartet werden.

7. Anlegen der Datei .sdmshrc

   Die Datei .sdmshrc wird, falls vorhanden, von allen schedulix Kommandozeilen-Werkzeugen gelesen um Kommandozeilen-Parameter vorzubelegen. Im Folgenden wird davon ausgegangen, dass diese Datei existiert und für User, Passwort, Host und Port die korrekten Werte gesetzt enthält. Die Datei .sdmshrc wird im Home-Verzeichnis des Linux-Benutzers angelegt.

   Ein Beispiel für den Inhalt ist:

```
$ cat ~/.sdmshrc
User=SYSTEM
Password=G0H0ME
Host=localhost
Port=2506
Timeout=0
```

Wichtig: Da die Datei die Daten für den Zugang zum Scheduling Server enthält, sollten die Datei-Rechte so gesetzt sein, dass nur der Owner die Datei lesen kann.

```
$ chmod 600 ~/.sdmshrc
$ ls -lG ~/.sdmshrc
-rw------- 1 schedulix 73 2011-11-09 09:28 /home/schedulix/.sdmshrc
```

8. Convenience Package installieren

   Das Convenience Package installiert eine übliche Konfiguration eines Exit State-Modells.

   ```
   $ sdmsh < $BICSUITEHOME/install/convenience.sdms
   ```

9. Beispiele installieren (optional)

   Das Installieren der Beispiele besteht aus zwei Teilen: Zum einen werden drei sogenannte Jobserver angelegt, welche für die nachfolgenden Ablaufdefinitionen benötigt werden. Zum anderen werden Beispiele für Ablaufdefinitionen in den Server geladen.

   a) Anlegen der Jobserver

      Um die Jobserver anzulegen, muss nur ein Skript ausgeführt werden:

      ```
      $ cd $BICSUITEHOME/install
      $ setup_example_jobservers.sh
      ```

   b) Einspielen der Ablaufdefinitionen

      Zum Einspielen der Ablaufdefinitionen werden folgende Befehle eingegeben:

      ```
      $ cd $BICSUITEHOME/install
      $ sdmsh < setup_examples.sdms
      ```

   Da die Beispiele davon ausgehen, dass die Jobserver bereits angelegt wurden, ist die obige Reihenfolge **zwingend**.

Installation eines schedulix Clients

Die Installation eines schedulix Scheduling Clients ist einfach. Es bedarf nur einiger Handlungen die im Folgenden erläutert werden.

Wenn (Beispiel-)Kommandos vorgestellt werden, wird als Prompt normalerweise ein `$` gezeigt. Diese Kommandos werden dann unter dem neu anzulegenden Account `schedulix` ausgeführt. In einigen Fällen wird der priviligierte Account `root` benötigt. Dies wird dadurch gekennzeichnet, dass als Prompt ein `#` gezeigt wird.

1. User `schedulix` anlegen
   Es gibt keine Notwendigkeit den User schedulix zu nennen. Damit kann der Name auch einer beliebigen Konvention angepasst werden. In diesem Dokument wird davon ausgegangen, dass der User schedulix heißt.

   Unter Ubuntu Linux kann ein User folgendermaßen angelegt werden:

   ```
   # useradd -d /home/schedulix -m -s /bin/bash -U schedulix
   # passwd schedulix
   ```

   Alle nachfolgenden Aktionen werden unter User schedulix ausgeführt, es sei denn es wird explizit anders angegeben.

2. Software auspacken

   tar-Archiv auspacken im Homeverzeichnis von schedulix. Etwa:

   ```
   $ tar xvzf schedulix-2.9.tgz
   ```

   Symbolic Link anlegen:

   ```
   $ ln -s schedulix-2.9 schedulix
   ```

3. Konfiguration erstellen

   a) Benutzerumgebung

      Um mit dem schedulix System arbeiten zu können, müssen folgende Variablen gesetzt werden:

      ```
      BICSUITEHOME=/home/schedulix/schedulix
      BICSUITECONFIG=/home/schedulix/etc
      PATH=$BICSUITEHOME/bin:$PATH
      SWTJAR=/usr/lib/java/swt.jar
      JNAJAR=/usr/share/java/jna.jar
      ```

      Es hat sich in der Praxis als vorteilhaft erwiesen die Konfiguration des Systems außerhalb des Installationsverzeichnisses zu legen. Damit werden spätere Upgrades wesentlich erleichtert. Da die Variablen von allen Benutzern des Systems gesetzt werden müssen, kann es sinnvoll sein die Zuweisungen (und Exports) in einer eigenen Datei zu schreiben, und diese dann im `.profile` oder `.bashrc` zu sourcen.

   b) Softwareumgebung

      Unter `$BICSUITEHOME/etc` liegen einige Vorlagen für Konfigurationsdateien, die als Basis für die Systemkonfiguration verwendet werden sollten.

Für eine Client-Installation benötigen wir die Dateien `bicsuite.conf` und `java.conf`.

Diese müssen dazu ohne die Endung ".template" ins Konfigurationsverzeichnis `$BICSUITECONFIG` kopiert werden.

```
$ cp $BICSUITEHOME/etc/bicsuite.conf.template \
     $BICSUITECONFIG/bicsuite.conf
$ cp $BICSUITEHOME/etc/java.conf.template \
     $BICSUITECONFIG/java.conf
```

Die Datei `bicsuite.conf` setzt einige Default-Einstellungen und muss im Allgemeinen nicht angepasst werden.

Die Datei `java.conf` beschreibt die zu verwendende Java-Umgebung und muss im Allgemeinen nicht weiter angepasst werden.

4. Anlegen der Datei `.sdmshrc`

   Die Datei `.sdmshrc` wird, falls vorhanden, von allen schedulix Kommandozeilen-Werkzeugen gelesen um Kommandozeilen-Parameter vorzubelegen. Im Folgenden wird davon ausgegangen, dass diese Datei existiert und für User, Passwort, Host und Port die korrekten Werte gesetzt enthält. Die Datei `.sdmshrc` wird im Home-Verzeichnis des Linux-Benutzers angelegt.

   Ein Beispiel für den Inhalt ist:

   ```
   $ cat ~/.sdmshrc
   User=SYSTEM
   Password=G0H0ME
   Host=localhost
   Port=2506
   Timeout=0
   ```

   Wichtig: Da die Datei die Daten für den Zugang zum Scheduling Server enthält, sollten die Datei-Rechte so gesetzt sein, dass nur der Owner die Datei lesen kann.

   ```
   $ chmod 600 ~/.sdmshrc
   $ ls -lG ~/.sdmshrc
   -rw------- 1 schedulix 73 2011-11-09 09:28 /home/schedulix/.sdmshrc
   ```

Beispiel Installation eines Jobservers

Im Folgenden wird die Installation eines Jobservers anhand eines Beispieles durchgeführt.

Szenario

Auf dem Rechner `machine_42` soll ein Jobserver Prozesse als Benutzer `arthur` ausführen. Das HOME-Verzeichnis des Benutzers sei `/home/arthur`.
Der schedulix Server sei auf dem Rechner `scheduling_server` installiert und hört auf den Port 2506. Das System-Passwort sei `G0H0ME`.

Vorraussetzungen

Auf dem Rechner `machine_42` wurde eine schedulix Client-Installation im HOME-Verzeichnis `/home/schedulix` durchgeführt.
Der Benutzer `arthur` hat folgende Zugriffsberechtigungen auf die Dateien der Client-Installation:

- Leserechte auf die Dateien `java.conf` und `bicsuite.conf` im Konfigurationsverzeichnis `$BICSUITECONFIG`. Auf alle Dateien unter `/home/schedulix/lib` werden ebenfalls Leserechte benötigt.
- Lese- und Ausführungsrechte auf alle Dateien unter `/home/schedulix/bin`.

Installation

Damit ein Jobserver sich am schedulix Scheduling Server anmelden kann muss der Jobserver dem schedulix Scheduling Server bekannt gemacht und konfiguriert werden. Im Folgenden führen wir die notwendigen Schritte dazu auf Kommandozeilenebene mit dem Werkzeug `sdmsh` durch. Dies kann jedoch alternativ auch über das Web GUI getan werden.

1. Anmelden als Benutzer `arthur` auf dem Rechner `machine_42`
2. Setzen der Umgebungsvariablen in der Shell und `.bashrc`.
   ```
   export BICSUITEHOME=/home/schedulix/schedulix
   export BICSUITECONFIG=/home/schedulix/etc
   export PATH=$BICSUITEHOME/bin:$PATH
   ```
3. Testen, ob die Umgebung korrekt ist
   ```
   sdmsh --host localhost --port 2506 --user SYSTEM --pass G0H0ME
   ```
   Ein `SDMS>` Prompt sollte erscheinen ('exit' beendet sdmsh).
4. Anlegen der Verzeichnisse
   ```
   cd $HOME
   mkdir etc
   mkdir taskfiles
   mkdir work
   mkdir log
   ```
5. Erzeugen einen Scopes für die Maschine `machine_42` mit sdmsh

```
SDMS> CREATE OR ALTER SCOPE GLOBAL.'MACHINE_42'
 WITH
  CONFIG = (
   'JOBEXECUTOR' = '/home/schedulix/schedulix/bin/jobserver',
   'HTTPHOST' = 'machine_42'
  );
```

Die Pfade müssen explizit angegeben werden, die Benutzung von Umgebungsvariablen ist hier nicht möglich.

6. Erzeugen des Jobservers mit sdmsh

```
SDMS> CREATE OR ALTER JOB SERVER GLOBAL.'MACHINE_42'.'ARTHUR'
 WITH
  PASSWORD =  'dent',
  NODE = 'machine_42',
  CONFIG = (
   'JOBFILEPREFIX' = '/home/arthur/taskfiles/',
   'DEFAULTWORKDIR' = '/home/arthur/work',
   'HTTPPORT' = '8905',
   'NAME_PATTERN_LOGFILES' = '/home/arthur/work/.*\\.log'
  );
```

Der `HTTPPORT` ist für die Anzeige von Job Log-Dateien aus dem Browser notwendig und ist beliebig wählbar, muss aber für alle Jobserver auf derselben Maschine eindeutig sein.
Die Pfade müssen explizit angegeben werden, die Benutzung von Umgebungsvariablen ist hier nicht möglich.

7. Anlegen der Named Resource für den Jobserver mit sdmsh

```
SDMS> CREATE OR ALTER NAMED RESOURCE RESOURCE.'JOBSERVERS'
      WITH USAGE = CATEGORY;
SDMS> CREATE NAMED RESOURCE RESOURCE.'JOBSERVERS'.'ARTHUR@MACHINE_42'
      WITH USAGE = STATIC;
```

8. Anlegen eines Environments für den Jobserver mit sdmsh

```
SDMS> CREATE ENVIRONMENT 'ARTHUR@MACHINE_42'
      WITH RESOURCE = (RESOURCE.'JOBSERVERS'.'ARTHUR@MACHINE_42');
```

9. Anlegen der Resource im Jobserver mit sdmsh

```
SDMS> CREATE RESOURCE RESOURCE.'JOBSERVERS'.'ARTHUR@MACHINE_42'
      IN GLOBAL.'MACHINE_42'.'ARTHUR' WITH ONLINE;
```

10. Erzeugen der Konfigurationsdatei `$HOME/etc/jobserver.conf` für den Jobserver mit folgendem Inhalt:

```
RepoHost= scheduling_server
RepoPort= 2506
RepoUser= "GLOBAL.'MACHINE_42'.'ARTHUR'"
RepoPass= dent
```

11. Starten des Jobservers

    ```
    jobserver-run $HOME/etc/jobserver.conf $HOME/log/jobserver.out
    ```

    Soll der Jobserver automatisch mit dem Start des Rechners gestartet werden, ist dies vom Systemadministrator entsprechend einzurichten.

Installation mit Postgres

Einleitung

Diese Anleitung erhebt nicht den Anspruch eine genaue Beschreibung der Installation des Datenbanksystems zu sein. Dazu wird auf die Postgres-Dokumentation verwiesen. Im Normalfall sollte es mit dieser Anleitung allerdings möglich sein eine "Standard"-Installation durchzuführen.

Installation

1. Herunterladen und Installieren der aktuellen Postgres-Version

   Normalerweise wird für jede Linux Distribution ein Postgres Package angeboten. Dieses Package, sowie ein Package für den JDBC Treiber für Postgres, sollte problemlos installiert werden können.

2. Konfiguration der Datei `pg_hba.conf`

   Damit sich der schedulix Scheduling Server mit Benutzer und Passwort bei PostgreSQL authentifizieren kann, muss folgende Zeile in die Postgres-Konfigurationsdatei `pg_hba.conf` aufgenommen werden. Diese Datei liegt typischerweise im Verzeichnis `/var/lib/pgsql/<version>/data`.

   ```
   host        all        all        127.0.0.1/32        md5
   ```

   PostgreSQL muss dann neu gestartet werden.

3. Anlegen des Postgres Users `schedulix`

   Führen Sie als User postgres den Befehl `createuser` wie im Beispiel (Version 8) aus:

   ```
   $ createuser -P schedulix
   Enter password for new role:
   Enter it again:
   Shall the new role be a superuser? (y/n): n
   Shall the new role be allowed to create databases? (y/n): y
   Shall the new role be allowed to create more new roles? (y/n): n
   ```

beziehungsweise, für Version 9:

```
$ createuser -P -d schedulix
```

Das eingegebene Passwort wird später noch benötigt.

4. Anlegen der Repository Datenbank `schedulixdb`

   Legen Sie nun als Benutzer schedulix die Datenbank für das Repository wie im untenstehenden Beispiel an:

   ```
   $ createdb schedulixdb
   ```

5. Anlegen und Initialisierung der Datenbanktabellen

   Um das benötigte Datenbankschema anzulegen, wechseln Sie in das schedulix SQL-Verzeichnis und rufen Sie das Postgres Utility `psql` wie im untenstehenden Beispiel auf:

   ```
   $ cd $BICSUITEHOME/sql
   $ psql -f pg/install.sql schedulixdb
   ```

6. Konfigurieren der Datenbankverbindung in der schedulix Server Konfigurationsdatei `$BICSUITECONFIG/server.conf`

   Ändern Sie folgende Properties wie angegeben:

   ```
   DbPasswd=schedulix passwort
   DbUrl=jdbc:postgresql:schedulixdb
   DbUser=schedulix
   JdbcDriver=org.postgresql.Driver
   ```

   Die `DbUrl` ist etwas abhängig von der installierten PostgreSQL-Version. Unter Version 8 lautet sie

   ```
   DbUrl=jdbs:postgresql:schedulixdb
   ```

7. Konfigurieren Sie den schedulix Java Class Path für Postgres JDBC

   In der Konfigurationsdatei `$BICSUITECONFIG/java.conf` muss nun nur noch der Pfad zum Postgres JDBC-Treiber am `CLASSPATH` angehängt werden.

   Etwa

   ```
   BICSUITECLASSPATH=$BICSUITEJAR:/usr/share/java/postgresql-jdbc4-9.2.jar
   ```

Installation mit MySQL

Einleitung

Diese Anleitung erhebt nicht den Anspruch eine genaue Beschreibung der Installation des Datenbanksystems zu sein. Dazu wird auf die MySQL-Dokumentation verwiesen. Im Normalfall sollte es mit dieser Anleitung allerdings möglich sein eine ”Standard”-Installation durchzuführen.

Installation

1. Herunterladen und Installieren der aktuellen MySQL-Version.

   Für die meisten Linux-Distributionen gibt es fertige MySQL Packages. Diese können mit den entsprechenden Tools einfach installiert werden.

   Im Rahmen dieser Installation wird nach einem Passwort für den MySQL root-User gefragt (nicht zu verwechseln mit dem Linux root-User). Dieses Passwort wird im nächsten Schritt wieder benötigt.

   Da schedulix für den Zugriff auf die Datenbank eine JDBC Connection aufbaut, muss auch der MySQL JDBC-Treiber installiert werden.

2. Anlegen des MySQL Users `schedulix` und der Datenbank schedulixdb

   Starten Sie das Utility `mysql` und melden Sie sich als MySQL root-User an um den User schedulix sowie die Datenbank schedulixdb anzulegen:

```
$ mysql --user=root --password=mysql-root-password

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 41
Server version: 5.1.54-1ubuntu4 (Ubuntu)

Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL v2 license

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create user schedulix identified by 'schedulix_passwort';
Query OK, 0 rows affected (0.01 sec)

mysql> create database schedulixdb;
Query OK, 1 row affected (0.00 sec)

mysql> grant all on schedulixdb.* to schedulix;
Query OK, 0 rows affected (0.00 sec)

mysql> quit
Bye
```

3. Anlegen und Initialisierung der Datenbanktabellen

   Führen Sie folgende Kommandos aus:

```
$ cd $BICSUITEHOME/sql
$ mysql --user=schedulix --password=schedulix_passwort
     --database=schedulixdb --execute="source mysql/install.sql"
```

4. Konfigurieren der Datenbankverbindung in der schedulix Server-Konfigurationsdatei `$BICSUITECONFIG/server.conf`

   Ändern Sie folgende Properties wie angegeben:

```
DbPasswd=schedulix_passwort
DbUrl=jdbc:mysql:///schedulixdb
DbUser=schedulix
JdbcDriver=com.mysql.jdbc.Driver
```

5. Konfigurieren Sie den schedulix Java Class Path für MySQL JDBC

   In der Konfigurationsdatei `$BICSUITECONFIG/java.conf` muss nun nur noch der Pfad zum MySQL JDBC-Treiber an dem `CLASSPATH` angehängt werden.

   Etwa

   ```
   BICSUITECLASSPATH=$BICSUITEJAR:/usr/share/java/mysql-connector-java.jar
   ```

Installation mit Ingres

Einleitung

Diese Anleitung erhebt nicht den Anspruch eine genaue Beschreibung der Installation des Datenbanksystems zu sein. Dazu wird auf die Ingres-Dokumentation verwiesen. Im Normalfall sollte es mit dieser Anleitung allerdings möglich sein eine ”Standard”-Installation durchzuführen.

Installation

1. Installation von Ingres

   Wir gehen davon aus, dass das Ingres-System unter User `ingres` installiert wird. Der Installations-Identifier wird hier als `II`, was dem Standardwert entspricht, angenommen.

2. Anlegen des Users `schedulix`

   Um den Benutzer im Ingres-System bekannt zu machen, gibt es zwei Möglichkeiten. Als Erste kann der Benutzer mit Hilfe des Tools `accessdb` angelegt werden. Diese Möglichkeit wird hier nicht weiter erläutert.

   Die zweite Möglichkeit ist das Anlegen des Benutzers mittels SQL-Befehl. Dazu starten Sie als Ingres den SQL Terminal Monitor:

   ```
   $ su - ingres
   Password:
   ingres@cheetah:~$ sql iidbdb
   INGRES TERMINAL MONITOR Copyright 2008 Ingres Corporation
   Ingres Linux Version II 9.2.1 (a64.lnx/103)NPTL login
   Mon Jun 13 10:05:19 2011

   continue
   * create user schedulix with privileges = (createdb);
   * \g
   Executing . . .
   ```

```
continue
* commit;\g
Executing . . .

continue
* \q
Ingres Version II 9.2.1 (a64.lnx/103)NPTL logout
Mon Jun 13 10:07:58 2011
ingres@cheetah:~$
```

3. Anlegen der Repository Datenbank `schedulixdb`

```
$ $II_SYSTEM/ingres/bin/createdb schedulixdb
Creating database 'schedulixdb' . . .

  Creating DBMS System Catalogs . . .
  Modifying DBMS System Catalogs . . .
  Creating Standard Catalog Interface . . .
  Creating Front-end System Catalogs . . .

Creation of database 'schedulixdb' completed successfully.
```

4. Anlegen und Initialisierung der Datenbanktabellen

   Zum Anlegen der benötigten Tabellen führen Sie folgende Befehle durch:

```
$ cd $BICSUITEHOME/sql
$ sql schedulixdb < ing\install.sql
```

5. Konfigurieren der Datenbankverbindung in der schedulix Server Konfigurationsdatei `$BICSUITECONFIG/server.conf`

   Ändern Sie folgende Properties wie angegeben:

```
DbPasswd=<schedulix OS passwort>
DbUrl=jdbc:ingres://localhost:II7/schedulixdb;
DbUser=schedulix
JdbcDriver=com.ingres.jdbc.IngresDriver
```

6. Konfigurieren Sie den schedulix Java Class Path für Ingres JDBC

   In der Konfigurationsdatei `$BICSUITECONFIG/java.conf` muss nun nur noch der Pfad zum Ingres JDBC Treiber an dem `CLASSPATH` angehängt werden.

   Etwa

```
BICSUITECLASSPATH=$BICSUITEJAR:$II_SYSTEM/ingres/lib/iijdbc.jar
```

Installation des Zope Servers

Einleitung

Um die schedulix!Web User Interface-Oberfläche nutzen zu können, muss ein Zope Application Server aufgesetzt werden.

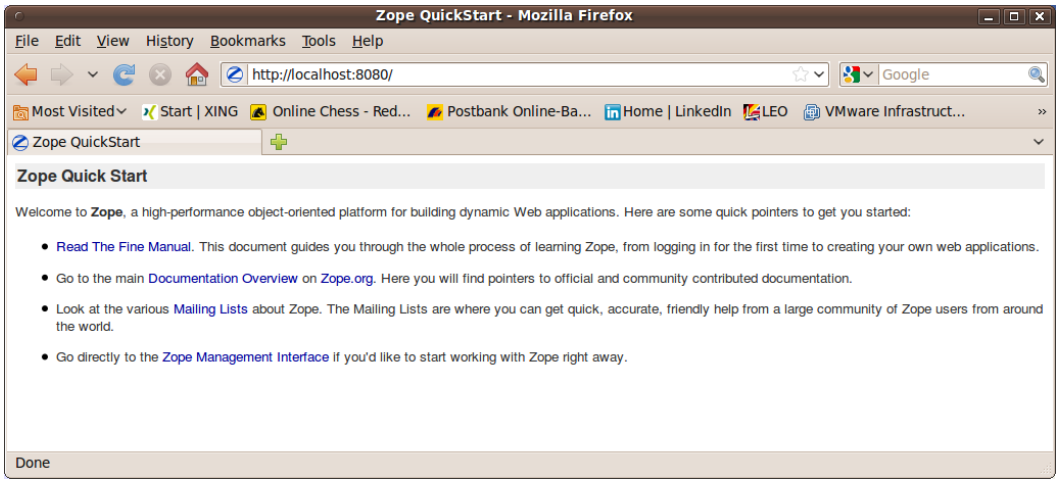

Abbildung 3.1: Zope Quick Start Seite

Installation

1. Installation virtualenv

   ```
   $ easy_install virtualenv
   ```

2. Erzeugen der virtuellen Python-Umgebung für die Zope-Installation

   ```
   $ mkdir $HOME/software
   $ cd $HOME/software
   $ virtualenv --no-site-packages Zope
   ```

3. Installieren der Zope2 Software

   Installieren Sie die neueste Release von Zope2. Zum Zeitpunkt der Erstellung dieses Dokumentes war die Release 2.13.29 die Aktuellste.

   ```
   $ cd $HOME/software/Zope
   $ bin/pip install -r \
   https://raw.githubusercontent.com/zopefoundation/Zope/2.13.29/requirements.txt
   ```

   Kann bei der Installation nicht auf das Internet zugegriffen werden, kann Zope auch offline installiert werden. Dazu ist wie folgt vorzugehen:

a) Download python packages

Auf einem möglichst identischen System mit Internetzugang führen Sie folgende Kommandos aus:

```
$ wget \
https://raw.githubusercontent.com/zopefoundation/Zope/2.13.29/requirements.txt
$ pip download -r requirements.txt -d packages
```

b) Dateien auf das Zielsystem übertragen

Die Datei requirements.txt und das Verzeichnis packages müssen nun auf das Zielsystem ohne Internetzugang übertragen werden. Legen Sie die Dateien in $HOME/software ab.

c) Installation auf dem Zielsystem

Folgendes Kommando installiert Zope aus den heruntergeladenen Dateien:

```
$ cd $HOME/software
$ Zope/bin/pip install --no-index --use-wheel \
--find-links=./packages  -r requirements.txt
```

4. Erzeugen einer Zope-Instanz für schedulix!Web

```
$ cd $HOME/software/Zope
$ bin/mkzopeinstance -d $HOME/schedulixweb -u sdmsadm:sdmsadm_passwort
```

Das Passwort kann beliebig gewählt werden und wird später wieder benötigt. Der Benutzer muss aber `sdmsadm` heißen.

Zum Testen wird der Zope Server kurz gestartet

```
$ $HOME/schedulixweb/bin/zopectl start
```

Im Internet Browser sollte die URL

```
http://localhost:8080
```

nun die Zope Quick Start Seite wie im Bild 3.1 anzeigen.

Die Zope-Instanz wird nun wieder angehalten.

```
$ $HOME/schedulixweb/bin/zopectl stop
```

5. Installieren der schedulix!Web-Komponenten

Um die schedulix!Web-Komponenten zu installieren, muss die Zope-Installation um einige Module erweitert werden:

```
$ cd $HOME/schedulixweb
$ mkdir Extensions
$ cd Extensions
$ ln -s $HOME/schedulix/zope/*.py .
$ cd ../Products
$ ln -s $HOME/schedulix/zope/BICsuiteSubmitMemory .
$ cd ../import
$ ln -s $HOME/schedulix/zope/SDMS.zexp .
```

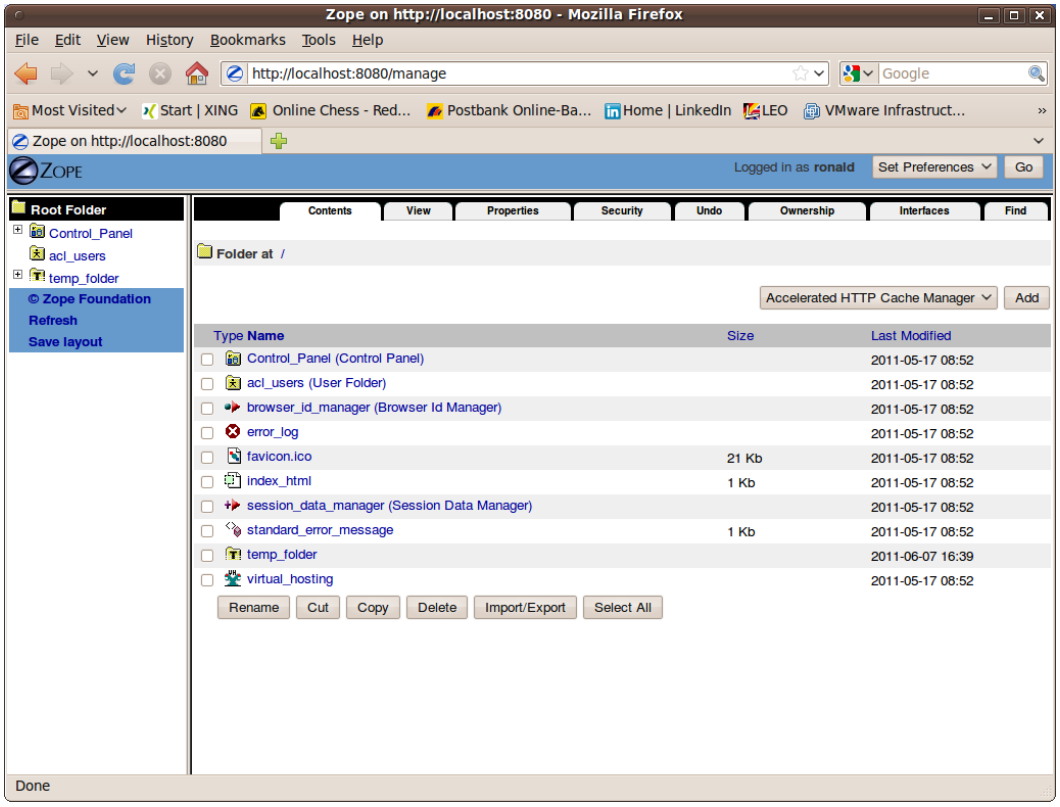

Abbildung 3.2: Zope Management Oberfläche

Nun muss die Zope-Instanz wieder gestartet werden, um die Änderungen auch Zope-seitig bekannt zu machen.

```
$ $HOME/schedulixweb/bin/zopectl start
```

Die Zope Management Oberfläche wird nun unter der Adresse

`http://localhost:8080/manage`

mit Hilfe eines Browsers geöffnet (siehe Bild 3.2). Dazu wird der Benutzer `sdmsadm` mit dem von Ihnen vergebenen Passwort benutzt.

Es wird jetzt die Frontend Software in Zope geladen (Import Button, siehe Bild 3.3)

a) im Folder / SDMS.zexp importieren

b) im Folder /SDMS/Install die Folder User und Custom anwählen und mit Copy kopieren

c) im Folder / mit Paste die Folder User und Custom erzeugen

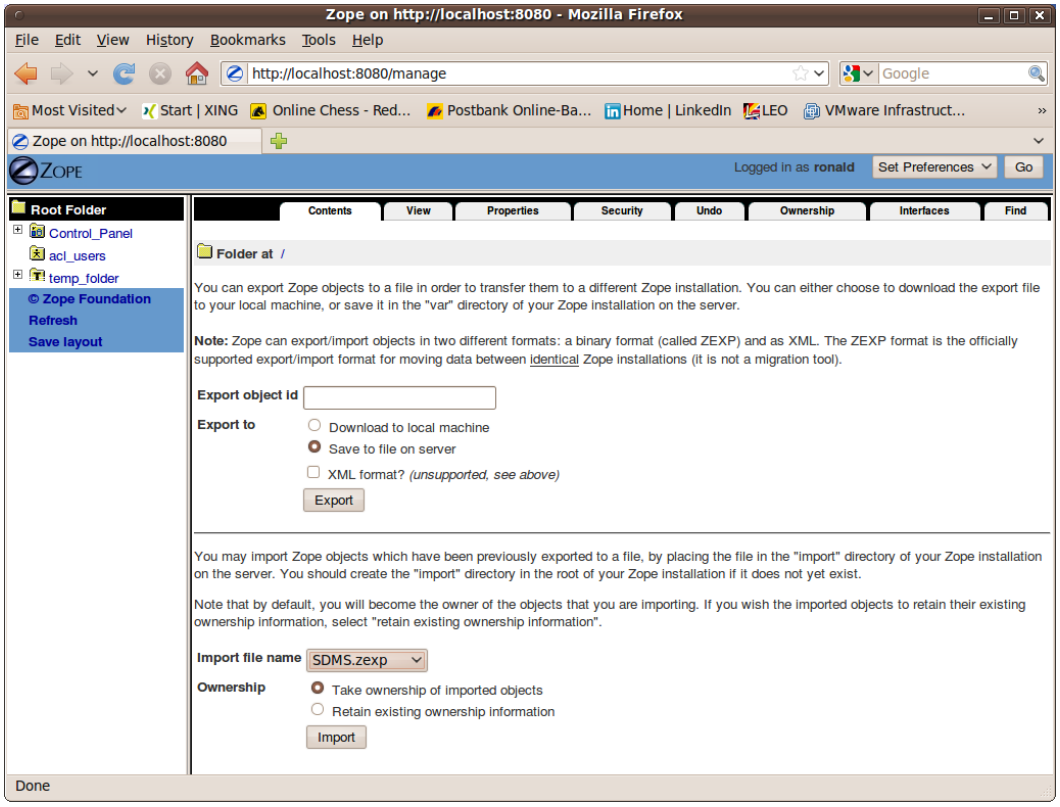

Abbildung 3.3: Zope Import Dialog

Wenn nun alles fehlerfrei durchgeführt werden konnte, sieht die Oberfläche wie auf dem Bild 3.4 aus.

6. Serververbindungen konfigurieren

   Das Konfigurieren der Serververbindungen erfolgt ebenfalls aus der Zope Management-Oberfläche heraus. Dazu meldet man sich als Benutzer `sdmsadm` an.

   Im Folder Custom wird das PythonScript `SDMSServers` editiert. Dieses Skript liefert ein Dictionary, welches für jeden schedulixServer, der von dieser schedulix!Web Installation angesprochen werden soll, einen Eintrag der Form

```
# Servername unter dem der Server in der schedulix!web Oberflaeche
# sichtbar ist
'servername' : {
    # IP Adresse oder Hostname auf dem der schedulix!server laeuft
    'HOST'    : 'hostname',

    # Port unter dem der schedulix!server angesprochen wird
    'PORT'    : '2506',
```

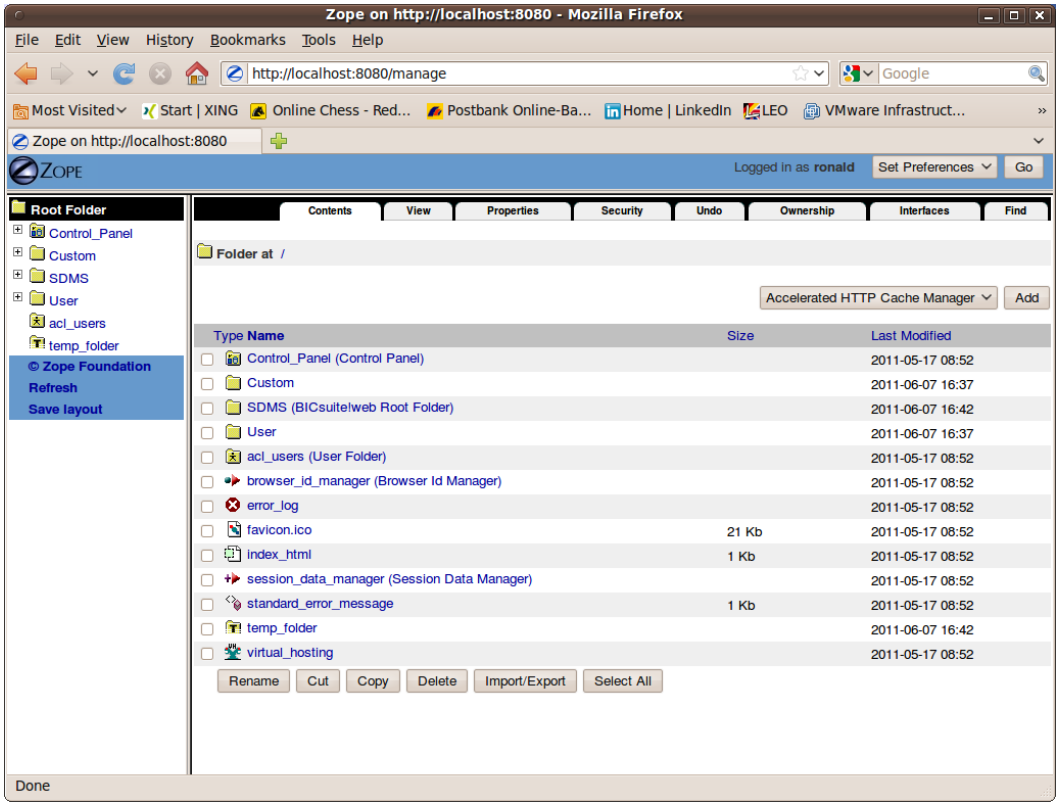

Abbildung 3.4: Zope Resultat Ansicht

```
    # BASIC
    'VERSION' : 'BASIC',

    # optionales Property, ob Zope Serververbindungen cachen soll
    'CACHE'   : 'Y'

    # optionales Property, wie lange gecachete schedulix!web
    # Serververbindungen gueltig sein sollen
    # default ist 60 sekunden, nur von Bedeutung falls 'CACHE' : 'Y'
    'TIMEOUT' : '60'
}
```

enthalten muss. Fürs Bootstrapping muss ein Eintrag mit Namen `DEFAULT` vorhanden sein. Dieser Eintrag kann nach dem Einrichten der Benutzer (die dann diese Connection natürlich nicht benutzen sollten) entfernt werden.

Soll ein Server über eine sichere SSL-Verbindung angesprochen werden, dann müssen folgende weitere Eigenschaften definiert werden:

```
# Verbindung wird ueber Secure Socket Layer aufgebaut
```

```
'SSL'        : 'true',

# falls angegeben, wird die Identitaet des BICsuite Servers
# ueberprueft. Die angegebene Datei muss das Server Certificate
# des BICsuite Servers enthalten
'TRUSTSTORE' : 'truststore.pem',

# falls der BICsuite Server eine Client Authentication fordert,
# muss dieses Property definiert sein und die angegebene Datei
# muss das Certificate und den Private Key des Clients enthalten.
# Das Certificate muss dem Server in seinem Truststore bekannt sein.
'KEYSTORE'   : 'keystore.pem'
```

Anmerkung:
Bei Verwendung von SSL wird aus Performancegründen die Verwendung von cached Serververbindungen empfohlen, da der Aufbau einer gesicherten Verbindung eine rechenintensive Operation ist.

7. Die schedulix!Web Oberfläche öffnen

   Die Benutzeroberfläche steht nun unter der Adresse

   ```
   http://localhost:8080/SDMS
   ```

   bereit. Nach dem Öffnen dieser Seite erscheint eine Aufforderung zur Anmeldung. Nach der Anmeldung wird die Applikation dann mit dem "Take Off" Button gestartet.

   Für das weitere Arbeiten mit der Oberfläche sei nun auf die dazugehörende Dokumentation verwiesen.

Installation der HTTPS Erweiterungen

Die Installation von der Zope HTTPS Erweiterung erfordert einige Arbeit. Es ist nicht besonders schwierig. Es ist aber wichtig die Anweisungen genau zu befolgen und idealerweise auch zu verstehen.

1. Abholen des `M2Crypto` Moduls

   Das M2Crypto Modul findet man derzeit unter

   `http://pypi.python.org/packages/source/M/M2Crypto/M2Crypto-0.21.1.tar.gz`

   Packen Sie es unter `$HOME` aus.

2. Installieren von M2Crypto in Ihr Virtual Environment

   ```
   $ cd  $HOME/M2Crypto-0.21.1
   $ $HOME/software/Zope/bin/python setup.py install
   ```

   Zum Testen, ob die Installation erfolgreich war, kann schlichweg versucht werden das Modul in Python zu laden:

```
$ $HOME/software/Zope/bin/python
Python 2.7.1+ (r271:86832, Apr 11 2011, 18:05:24)
[GCC 4.5.2] on linux2
Type "help","copyright","credits" or "license" for more information.
>>> import M2Crypto
>>>
```

Bei einer Fehlermeldung ist evtl. ein `ldconfig` oder das Schließen und Neuöffnen des Terminals notwendig.

3. Zope Installation patchen

   Der nächste Schritt sorgt dafür, dass Zope, je nach Konfiguration, auch das nun installierte M2Crypto Modul benutzen wird.

   ```
   $ $BICSUITEHOME/zope/https/patch.sh
   ```

4. Generieren der von Zope HTTPS benötigten Dateien

   Die von Zope benötigten Dateien können, müssen aber nicht, ins "etc"-Verzeichnis der Zope-Instanz abgelegt werden. Im Folgenden wird davon ausgegangen, dass dies auch passiert. Eine Modifizierung des Vorgehens setzt voraus, dass ein solides Verständnis von dem Funktionieren von TLS/SSL sowie HTTPS vorhanden ist.

   Zuerst wird zum "etc"-Verzeichnis gewechselt:

   ```
   $ cd $HOME/bicsuiteweb/etc
   ```

   Dann wird ein SSL Certificate Authority erstellt:

   ```
   $ openssl req -new -x509 -newkey rsa:2048 -keyout cakey.pem \
   > -out cacert.pem -days 3650
   ```

   Als Beispiel können dabei folgende Eingaben gemacht werden:

   ```
   Enter PEM pass phrase: super_geheim
   Verifying - Enter PEM pass phrase: super_geheim
   Country Name (2 letter code) [AU]:DE
   State or Province Name (full name) [Some-State]:Bayern
   Locality Name (eg, city) []:Schrobenhausen
   Organization Name (eg, company) [Internet Widgits Pty]:independIT
   Organizational Unit Name (eg, section) []:Development
   Common Name (eg, YOUR name) []:Dieter Stubler
   Email Address []:dieter.stubler@independit.de
   ```

   Nun kann ein SSL Server Certificate erstellt werden

   - Erzeugen des Keys für Server Certificate:

     ```
     $ openssl genrsa -out serverkey.pem -aes128 2048 -days 3650

     Enter pass phrase for serverkey.pem: dummy
     Verifying - Enter pass phrase for serverkey.pem: dummy
     ```

   - Entfernen des Passworts aus serverkey.pem

```
$ openssl rsa -in serverkey.pem -out serverkey.pem

Enter pass phrase for serverkey.pem: dummy
```

- Certificate Signing Request erzeugen

  ```
  $ openssl req -new -key serverkey.pem -out req.pem -nodes
  ```

  Als Beispiel können dabei folgende Eingaben gemacht werden:

  ```
  Country Name (2 letter code) [AU]:DE
  State or Province Name (full name) [Some-State]:Bayern
  Locality Name (eg, city) []:Schrobenhausen
  Organization Name (eg, company) [Internet Widgits Pty]:independIT
  Organizational Unit Name (eg, section) []:Development
  Common Name (eg, YOUR name) []:my_hostname
  Email Address []:dieter.stubler@independit.de
  A challenge password []:
  An optional company name []:
  ```

  Achtung !!!

  `my_hostname` muss durch den Namen ersetzt werden, unter dem der Browser den HTTPS Host anspricht !!!

- Certificate signieren

  a) Sichern openssl.cnf (benötigt meistens root-Rechte)

  ```
  # cp /etc/ssl/openssl.cnf /etc/ssl/openssl.cnf_save
  ```

  b) Editieren openssl.cnf

  ```
  # sudo vi /etc/ssl/openssl.cnf
  ```

  Dabei werden folgende Einträge geändert:

  ```
  dir             = .
  private_key     = $dir/cakey.pem
  RANDFILE        = $dir/.rand
  default_days    = 3650
  new_certs_dir   = $dir
  ```

  c) Prepare files

  ```
  $ touch index.txt
  $ echo 01 > serial
  ```

  d) Signieren

  ```
  $ openssl ca -in req.pem -notext -out servercert.pem
  ```

  Beispiel:

  ```
  Enter pass phrase for ./cakey.pem: super_geheim
  Sign the certificate? [y/n]:y
  1 out of 1 certificate requests certified, commit? [y/n]y
  ```

  e) Wiederherstellen der alten openssl.cnf

```
# mv /etc/ssl/openssl.cnf_save /etc/ssl/openssl.cnf
```

- SSL Server Certificate schreiben

  ```
  $ cat servercert.pem serverkey.pem > server_cert.pem
  ```

- Erzeugen des SSL Entropy Pools

  ```
  $ dd if=/dev/random of=ssl_entropy_pool.dat bs=1024 count=1
  ```

- Erzeugen des SSL DH Init:

  ```
  $ openssl dhparam -out ssl_dh_init.pem 1024
  ```

5. Zope HTTPS konfigurieren

   Falls HTTPS auf Port 8085 hören soll und

   ```
   $HOME == /home/bicsuite
   ```

   ist, soll in die Datei `$HOME/bicsuiteweb/etc/zope.conf` folgendes nach dem Endtag `</http-server>` eingefügt werden:

```
<https-server>
 # valid keys are "address", "force-connection-close",
 # required keys are
 #        "x509_remote_users",
 #        "ssl_certificate_authority",
 #        "ssl_server_certificate"
 address 8085
 x509_remote_users off
 ssl_certificate_authority /home/bicsuite/bicsuiteweb/etc/cacert.pem
 ssl_server_certificate /home/bicsuite/bicsuiteweb/etc/server_cert.pem
 ssl_dh_init /home/bicsuite/bicsuiteweb/etc/ssl_dh_init.pem
 ssl_entropy_pool /home/bicsuite/bicsuiteweb/etc/ssl_entropy_pool.dat
 # force-connection-close off
</https-server>
```

   Soll nur über HTTPS zugegriffen werden können, so ist der `<http-server>` Bereich bis `</http-server>` auszukommentieren.

6. Zope neu starten

   ```
   $ $HOME/bicsuiteweb/bin/zopectl stop
   ```

   Wir benutzen runzope, damit wir Fehler gleich sehen.

   ```
   $ $HOME/bicsuiteweb/bin/runzope
   ```

   Für Problemanaylse ist auch `$HOME/bicsuiteweb/log/event.log` hilfreich.

7. Nachdem es funktioniert hat, runzope abbrechen (Ctrl-C) und Zope neu starten.

   ```
   $ $HOME/bicsuiteweb/bin/zopectl start
   ```

Aufruf von BICsuite!Web:
Im Browser
http://hostname:8080/SDMS
oder
https://hostname:8085/SDMS

Beim ersten Aufruf muss das Server Certificate dem Browser als vertrauenswürdig bestätigt werden. Bei Firefox reicht dies aus. Unter Internet Explorer muss das

/home/bicsuite/bicsuiteweb/etc/cacert.pem

unter Internetoptionen → Inhalte → Herausgeber → Vertrauenswürdige Stammzertifizierungsstellen importiert werden, damit ohne andauernde Warnungen gearbeitet werden kann.