Kinderbetreuung
Stadt Bern

Ki-Tax-Installationshandbuch

Änderungen

Version	Datum	Autor(en)	Bemerkung
2.2.3	08.03.2018	Hofstetter Markus	Übernahme ins Bern-Layout
3.2.0	16.03.2018	Herger Franziska	Neue Einstellungen aus Version 3.2.0 ergänzt

Herausgeberin: Stadt Bern | Direktion für Bildung, Soziales und Sport | Jugendamt / Ki-Tax
Effingerstrasse 21 | 3008 Bern | Telefon 031 321 51 15
kinderbetreuung@bern.ch | www.bern.ch/ki-tax | www.bern.ch/kinderbetreuung
Bern, 19. März 2018

Inhalt

1 Zweck

Dieses Dokument beschreibt die Installation der Applikation Ki-Tax auf dem Test- und dem Produktionsserver inklusive der notwendigen initialen Konfigurationen. Mit fortschreitender Entwicklung werden weitere Konfigurationen vorgenommen werden müssen.

1.1 Allgemeine Informationen

Die Applikation E-BEGU wird auf der Basis von JEE (Java Enterprise Edition) Version 7 entwickelt.

Die Installation benötigt die folgenden Serverkomponenten:

- Java JRE ab Version 8 (Java Virtual Machine)
- Wildfly 10.0.0.Final mit integriertem Apache Tomcat
- Webserver (beliebig)
- Maria DB 5.5.44 +

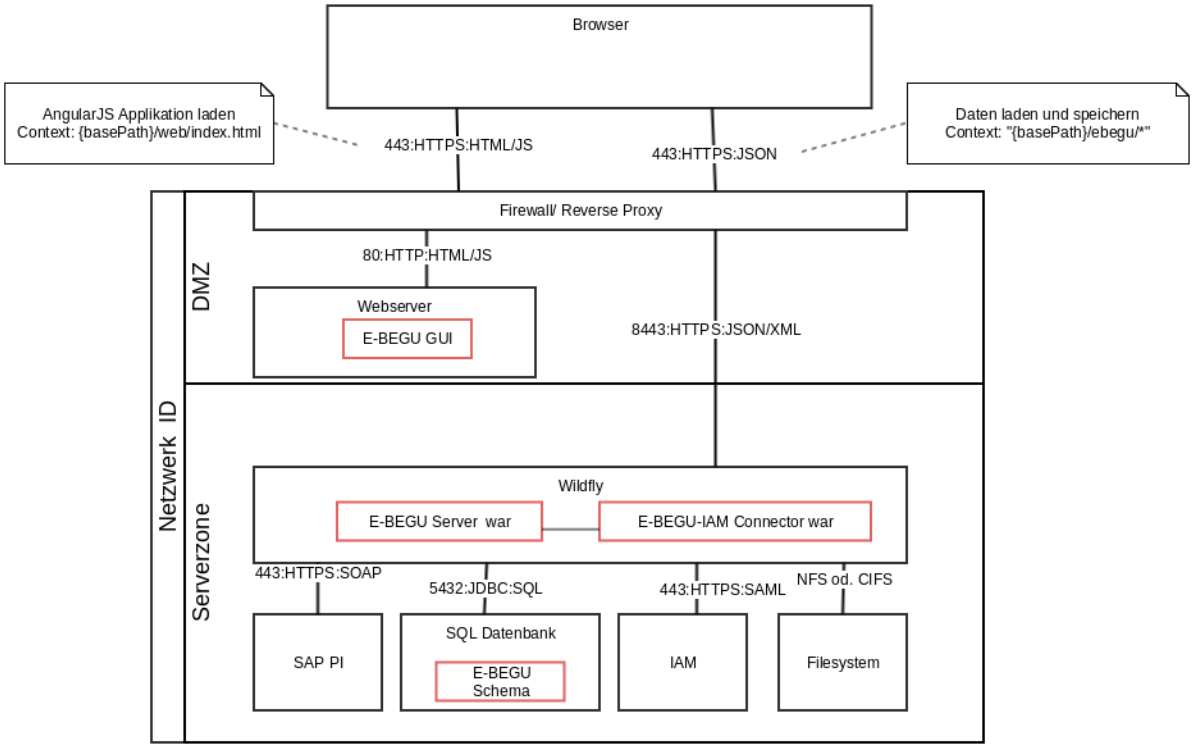
Weil das System auf der Basis von Java entwickelt wurde, besteht eine weitgehende Plattformunabhängigkeit (Hardware und Betriebssystem) für den eigentlichen Betrieb des Serversystems.

1.2 Anmerkung zum Namen

Das Projekt wurde unter dem Codenamen E-BEGU gestartet. Für das Marketing der Anwendung wurde der Name Ki-Tax verwendet. Deshalb referenziert die folgende Dokumentation häufig auf den Namen E-BEGU.

2 Komponentenübersicht

Bezeichnung	Funktion	Artefakt	Bemerkung
Webserver	Serven des web-basierten Clients	ebegu-web-client.tar.gz	Der statische (Client-)Teil der Applikation. Besteht aus HTML/JS/CSS. Beinhaltet die Darstellungslogik der Applikation. Das Bereitstellen dieses statischen Teils kann allenfalls auch von der WAF übernommen werden.
Wildfly Applikations-server	Applikationslogik, Kommunikation mit Schnittstellen	ebegu-rest.war	Deployment der Applikationslogik. Beinhaltet die eigentliche JEE Applikation. Bietet ein HTTP Interface an welches die Clients aufrufen um Daten auszutauschen oder Aktionen auszulösen. Beinhaltet die Businesslogik und übernimmt die Kommunikation mit sämtlichen Schnittstellen.
Maria Datenbank	Datenhaltung	(Schema ebegu)	Speichern der vom Benutzer eingegebenen Daten in einem Datenbankschema „ebegu“. Wird per JDBC angebunden.
Filesystem	Ablage der hochgeladenen Dokumente	(Ordner)	Speichern der vom Benutzer Hochgeladenen Dokumente.



2.1 Artefakte

Die Applikation wird in zwei Teilartefakten ausgeliefert (Client- und Serverteil).

Beim Client handelt es sich um eine mit AngularJS entwickelte Webapplikation. Der Client wird als Angular Applikation erstellt und in ein *.tar.gz File gepackt. Diese wird entpackt und auf einem Webserver bzw. der Web Application Firewall gehostet. Die Benutzer öffnen diese Website im Browser und sie kommuniziert über eine HTTP Schnittstelle mit dem Server.

Beim ebegu-rest.war handelt es sich um eine JEE Applikation welche die Businesslogik implementiert, die Daten über JPA in der Datenbank speichert sowie die Kommunikation mit allen Umsystemen abwickelt (SOAP, Mailing etc.). Für den Client stellt er mittels JAX-RS die Serviceschnittstelle zur Verfügung.

Dazu kommt optional ein Artefakt mit dem Login-Connector.

Um komplexere Login-Verfahren einzubinden, kommt dazu optional ein Artefakt mit dem Login-Connector. Modul verwendet die interne API von E-BEGU um die eingeloggten Benutzer an E-BEGU zu melden.

2.2 Java Virtual Machine

Der JEE Application Server benötigt eine Java 8 JVM. Diese muss gemäss der Anleitung von Oracle installiert werden.

https://docs.oracle.com/javase/8/docs/technotes/guides/install/

Hinweis: Aus Performance- und Konsistenzgründen sollte die JVM von Oracle und nicht OpenJDK verwendet werden.

2.3 Datenbank

De E-BEGU Applikation verwendet wo immer möglich Standard SQL. Als Datenbankserver empfehlen wir eigentlich grundsätzlich den Einsatz von PostgreSQL. Da aber die Informatik Dienste derzeit Maria DB bevorzugen wird fürs erste dieser Datenbankserver verwendet.

Auf dem Maria SQL Server muss eine Datenbank für die Applikation E-BEGU erstellt werden. Zudem wird ein technischer Benutzer mit Rechten für das erstellte Schema benötigt.

Das Anlegen der benötigten Tabellen und allfällige Schemaänderungen werden durch die Applikation selber beim deployen von E-BEGU auf den Wildfly vorgenommen.

Der Name des Datenbankschemas sollte „ebegu“ lauten.

```
# mariadb installieren
sudo apt-get install mariadb-server
# schema und Benutzer konfigurieren (in mysql command line):
create schema ebegu;
create user 'ebegu' identified by 'ebegu';
grant all on ebegu.* to 'ebegu';
```

2.4 Filesystem

Die Applikation wird später hochgeladene Dokumente im Filesystem ablegen. Dafür werden wir ein Netzwerkverzeichnis benötigen welches vom Wildfly aus erreicht werden kann.

2.5 Wildfly Application Server

Der (JBoss) Wildfly Application Server ist für den Betrieb der E-BEGU Applikation erforderlich. Er stellt eine JEE Betriebsumgebung zur Verfügung.

Wir empfehlen einen Wildfly User und Gruppe zu erstellen:

```
    # groupadd -r wildfly
    # useradd -r -g wildfly -d /opt/wildfly -s /sbin/nologin wildfly
```

Zur Installation muss der Wildfly 10.0.0.Final von der Seite http://wildfly.org/downloads/ heruntergeladen werden (direkter Downloadlink). Das Heruntergeladene Filearchiv muss entpackt werden.

```
    # tar xvzf wildfly-10.0.0.Final.tar.gz -C /opt
    # ln -s /opt/wildfly-10.0.0.Final /opt/wildfly
    # chown -R wildfly:wildfly /opt/wildfly
```

Das Vorgehen zur Installation des Wildfly als Service hängt stark vom Betriebssystem ab.

Im Ordner <wildfly-home>/docs/contrib/scripts/ gibt es einige Beispiele für init-scripts, systemd usw.

Das Installationsverzeichnis des Wildfly wird im Folgenden als <wildfly-home> bezeichnet.

Fast alle Konfigurationseinstellungen werden zentral in einem einzigen File vorgenommen

```
<wildfly-home>/standalone/configuration/standalone.xml
```

Dieses File wird im folgenden standalone.xml genannt.

2.5.1 Konfiguration der Datenbankverbindung

Im standalone.xml müssen folgende Einträge gemacht werden. (Fett = hinzufügen, gelb = auf Systemumgebung anpassen)

```
[File: <wildfly-home>/standalone/configuration/standalone.xml]


[…]
<datasources>
    <datasource jndi-name="java:jboss/datasources/ExampleDS" pool-name="ExampleDS" enabled="true"
                use-java-context="true">
        <connection-url>jdbc:h2:mem:test;DB_CLOSE_DELAY=-1;DB_CLOSE_ON_EXIT=FALSE</connection-url>
        <driver>h2</driver>
        <security>
            <user-name>sa</user-name>
            <password>sa</password>
        </security>
    </datasource>
    <xa-datasource jndi-name="java:/jdbc/ebegu" pool-name="ebegu" use-ccm="true">
        <xa-datasource-property name="url">
            ${ebegu.db.jdbcurl}
        </xa-datasource-property>
        <xa-datasource-class>org.mariadb.jdbc.MariaDbDataSource</xa-datasource-class>
        <driver>mariadbsql</driver>
        <transaction-isolation>TRANSACTION_READ_COMMITTED</transaction-isolation>
        <xa-pool>
            <min-pool-size>5</min-pool-size>
            <initial-pool-size>1</initial-pool-size>
            <max-pool-size>50</max-pool-size>
            <prefill>true</prefill>
            <use-strict-min>true</use-strict-min>
        </xa-pool>
        <security>
            <user-name>${ebegu.db.username}</user-name>
            <password>${ebegu.db.password}</password>
        </security>
        <validation>
```

```
            <valid-connection-checker
                    class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker"/>
            <validate-on-match>false</validate-on-match>
            <background-validation>true</background-validation>
            <background-validation-millis>10000</background-validation-millis>
            <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter"/>
        </validation>
        <timeout>
            <set-tx-query-timeout>true</set-tx-query-timeout>
            <idle-timeout-minutes>5</idle-timeout-minutes>
            <xa-resource-timeout>60</xa-resource-timeout>
        </timeout>
        <statement>
            <track-statements>true</track-statements>
            <prepared-statement-cache-size>500</prepared-statement-cache-size>
            <share-prepared-statements>true</share-prepared-statements>
        </statement>
    </xa-datasource>
    <drivers>
        <driver name="h2" module="com.h2database.h2">
            <xa-datasource-class>org.h2.jdbcx.JdbcDataSource</xa-datasource-class>
        </driver>
        <driver name="mariadbsql" module="org.mariadb">
            <driver-class>org.mariadb.jdbc.Driver</driver-class>
        </driver>
    </drivers>
</datasources>
[…]
```

Die gelb markierten Stellen müssen mit den für die Umgebung korrekten Werten ersetzt werden. Beispiel: ${ebegu.db.jdbcurl} ersetzen mit jdbc:mysql://meinDatenbankserver:3306/ebegu wobei meinDatenbankserver der Datenbankserver ist und ebegu der Name des Datenbankschemas.

Alternativ können diese Platzhalter auch in einem Property -File definiert werden welches beim Starten des Wildfly als Parameter (--properties=ebegu.properties) mitgegeben wird. Dadurch kann die gleiche Basiskonfiguration zum Beispiel auf Produktion und Test verwendet werden, lediglich das Property-File muss angepasst werden.

Beispiel für ein solches Propertyfile:

```
[File: ebegu.properties]


ebegu.db.jdbcurl=jdbc:mysql://meinDatenbankserver:3306/ebegu
ebegu.db.username=mein-begu-datenbankschema-username
```

```
ebegu.db.password=mein-ebegu-db-passwd
```

Der oben konfigurierte JDBC Treiber muss im Wildfly als Modul hinzugefügt werden. Dazu legt man ein neues Verzeichnis:

```
<jboss-home>/modules/org/mariadb/main
```

an und kopiert das Treiber JAR (in unserem Fall mariadb-java-client-1.3.7.jar) in dieses Verzeichnis.

Weiterhin muss man im selben Verzeichnis die Modul-Definitionsdatei module.xml mit folgendem Inhalt anlegen.

```
[File: <wildfly-home>/modules/org/mariadb/main/module.xml]

<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="org.mariadb">

    <resources>
        <resource-root path="mariadb-java-client-1.3.7.jar"/>
    </resources>

    <dependencies>
        <module name="javax.api"/>
        <module name="javax.transaction.api"/>
    </dependencies>
</module>
```

2.5.2 Konfiguration des Login-Moduls

Um ohne konkretes LoginModul testen zu können existiert ein Dummy Modus. Mit dem über die Seite #locallogin eingelogged werden kann. Damit dies klappt, muss das folgende Property aktiviert werden. ACHTUNG nicht für Produktion aktiviert lassen!

```
<property name="ebegu.dummy.login.enabled" value="true"
```

Um ein versehentliches Einschalten dieser Funktion in einer produktiven Umgebung zu erschweren, gibt es eine zusätzliche Einstellung in der Datenbank. Diese ist standardmässig auf false gesetzt und muss explizit geändert werden:

```
update application_property set value = 'true' where name = 'DUMMY_LOGIN_ENABLED';
```

Es müssen beide Einstellungen (Systemproperty und DB-Einstellung) auf true gesetzt werden, damit das Dummy Login aktiviert wird.

Um komplexere Login-Verfahren einzubinden, muss eine extra Applikation deployed werden, welche das Login-Verfahren abwickelt. Das Login Modul verwendet die interne API von E-BEGU um die eingeloggten Benutzer an E-BEGU zu melden.

Neue LoginConnectoren müssen über ein REST Schnittstelle mit E-BEGU kommunizieren. Der Connector muss dabei ein Service Interface implementieren über welches E-BEGU die URL abholt an die unauthentifizierten Benutzer zwecks Login weitergeleitet werden (ILoginProviderInfoResource).

Wenn ein Connector einen Benutzer als erfolgreich eingeloggt taxiert hat, muss er dies wiederum E-BEGU mitteilen. Dies erfolgt über eine von E-BEGU zur Verfügung gestellte REST Schnittstelle (ILoginConnectorResource)

Das API ist als Javadoc in folgenden Interfaces beschrieben:

- ch.dvbern.ebegu.api.connector.clientinfo.ILoginProviderInfoResource (muss implementiert werden)
- ch.dvbern.ebegu.api.connector.ILoginConnectorResource (muss konsumiert werden)

2.5.3 Konfiguration des cache-containers

E-BEGU verwendet einen cache-container um die Anzahl der nötigen Datenbankzugriffe zu reduzieren. Aus diesem Grund muss folgende Konfiguration eingefügt werden

```
[File: <wildfly-home>/standalone/configuration/standalone.xml]

<subsystem xmlns="urn:jboss:domain:infinispan:4.0">
[…]

        <cache-container name="ebeguCache" default-cache="ebeguAuthorizationCache">
            <local-cache name="ebeguAuthorizationCache">
                <transaction mode="FULL_XA"/>
                <eviction strategy="LRU" max-entries="10000"/>
                <expiration lifespan="1800000"/>
            </local-cache>
        </cache-container>
[…]

</subsystem>
```

2.5.4 Konfiguration der Anzahl Batch-Threads

In E-BEGU werden z.T. umfangreiche Statistiken erstellt. Diese werden asynchron ausgeführt, der Benutzer wird per Mail informiert, wenn die Auswertung fertig ist. Damit aber nicht zu viele Benutzer gleichzeitig eine Auswertung starten (Speicherverbrauch), kann die Anzahl Threads für die BatchJobs limitiert werden. Die Standardeinstellung ist 10.

```
[File: <wildfly-home>/standalone/configuration/standalone.xml]

<subsystem xmlns="urn:jboss:domain:batch-jberet:1.0">
    <default-job-repository name="in-memory"/>
    <default-thread-pool name="batch"/>
```

```
    <job-repository name="in-memory">
        <in-memory/>
    </job-repository>
    <thread-pool name="batch">
        <max-threads count="4"/>
        <keepalive-time time="30" unit="seconds"/>
    </thread-pool>
</subsystem>
```

2.5.5 Applikationsproperties

Im standalone.xml sollten folgende Properties eingefügt werden. Dazu kann der unten Fett markierte Abschnitt nach dem <extensions> Element eingefügt werden.

```
[File:<wildfly-home>/standalone/configuration/standalone.xml]
<extensions>
…
</extensions>

<system-properties>
    <property name="org.apache.coyote.http11.Http11Protocol.COMPRESSION" value="on"/>
    <property name="org.apache.coyote.http11.Http11Protocol.COMPRESSION_MIME_TYPES"
value="text/javascript,text/css,text/html"/>
    <property name="ebegu.development.mode" value="false"/>

</system-properties>
```

Im Folgenden werden die wichtigsten Configurationproperties für die E-BEGU Applikation aufgelistet und erklärt:

Name	Zweck und mögliche Values	Default
ebegu.client.using.https	Gibt an ob das Cookie in der Fedletantwort nur für https Verbindungen gesetzt werden soll. {true,false}	false
ebegu.document.file.path	Pfad zum Verzeichnis unter dem hochgeladene Dokumente abgelegt werden Zum Beispiel /media	Jboss data dir also jbosshome/standalone/data
ebegu.dummy.login.enabled	Gibt an ob die Seite mit den Dummy-Logins aktiviert ist. **Darf auf keinen Fall in der Produktion gesetzt werden!**	False
ebegu.mail.disabled	Schaltet das Versenden von Mails durch E-BEGU ein oder aus	Wert von ebegu.development.mode
ebegu.mail.smtp.from	Emailadresse die im VON Feld der versendeten Mails erscheint	-

ebegu.mail.smtp.host	Emailserver über den die Mails verschickt werden	-
ebegu.mail.smtp.port	Port des Emailservers	-
ebegu.hostname	Hostname der E-BEGU Installation. Wird verwendet um in den versendeten Mails Links einzubauen die zurück zum System führen.	-
ebegu.development.mode	Lässt die Applikation wissen ob sie sich im Entwicklungsmodus befindet. **Darf auf keinen Fall in der Produktion gesetzt werden!**	true
ebegu.openidm.url	URL des openidm Systems mit dem synchronisiert werden soll.	-
ebegu.openidm.user	User fuer die Anmeldung an OpenIDM (bzw OpenAM well loginwithtoken eingeschaltet ist)	-
ebegu.openidm.passwd	PW fuer die Anmeldung an OpenIDM	-
ebegu.openidm.enabled	Schalter mit dem die OpenIDM Synchronisierung an und ausgeschaltet werden kann	false
ebegu.openidm.loginwithtoken.enabled	Schalter mit dem definiert wird ob für die Konfiguration mit IDM zuerst ein logintoken von OpenAM System abgeholt wird welches dann im Header mitgeschickt wird	true
ebegu.openam.url	URL des Openam Systems bei dem das Token für die Kommunikation mit dem IDM System abgeholt wird.	-
ebegu.personensuche.disabled	Gibt an ob der Personensuchservice (EWK) enabled ist oder nicht. Wenn false wird für gewisse Namen ein Dummywert zurückgegeben (zu Testzwecken)	true
ebegu.personensuche.endpoint	Endpunkt-url unter der der SOAP Service läuft (ohne ?wsdl parameter).	-
ebegu.personensuche.wsdl	URL des WSDLs des SOAP Services	-
ebegu.personensuche.username	Benutzername für den EWK Service	-
ebegu.personensuche.password	Passwort für den EWK Service	-
ebegu.zahlungen.test.mode	Schalter, mit dem das Feld „Datum Generiert“ bei den Zahlungen manuell gesetzt werden kann. Dies ist notwendig, um die Zahlungen in die Zukunft zu testen.	False

	Darf auf keinen Fall in der Produktion gesetzt werden!	
hibernate.search.default.indexBase	Verzeichnis, in welches der Lucene Such-Index generiert werden soll	
ebegu.login.provider.api.url	URL unter der E-BEGU das API des Login Provider sucht	
ebegu.login.api.allow.remote	Kann das interne Login API auch remote aufgerufen werden	false
ebegu.login.api.internal.user	Username fuer das BasicLogin der internen API von E-BEGU	
ebegu.login.api.internal.password	Passwort für das BasicLogin der internen API von E-BEGU	
ebegu.login.connector.rest.api.url	URL unter der die LoginConnector Applikation das interne API von E-BEGU sucht	
ebegu.login.connector.rest.api.user	User mit dem der LoginConnector auf das interne E-BEGU API verbinden wird	
ebegu.login.connector.rest.api.password	Passwort mit dem der LoginConnector auf die interne E-BEGU API einloggt	
ebegu.login.connector.development.mode	Wenn true warden die header der REST calls ins Logfile geschrieben	
ebegu.testfaelle.enabled	Wenn true können in der Applikation vordefinierte Testfälle eingefügt werden	false

2.5.6 JVM Einstellungen

Für den optimalen Betrieb des Wildfly sollten noch einige Java Virtual Machine Einstellungen wie zum Beispiel der zur Verfügung stehende Speicher angepasst werden. Dazu werden die JAVA-OPTS im File

```
<wildfly-home>/bin/standalone.conf
```

Angepasst (Fett=hinzufügen, Gelb=anpassen).

```
[File:<wildfly-home>/bin/standalone.conf]


[…]


if [ "x$JAVA_OPTS" = "x" ]; then
   JAVA_OPTS="-Xms64m -Xmx4096m -XX:MetaspaceSize=96M -XX:MaxMetaspaceSize=256m -
Djava.net.preferIPv4Stack=true"
   JAVA_OPTS="$JAVA_OPTS -Djboss.modules.system.pkgs=$JBOSS_MODULES_SYSTEM_PKGS -
Djava.awt.headless=true"

#Since JBoss is generally a long running process, using -server is recommended
   JAVA_OPTS="$JAVA_OPTS -server"

#Enables the use of compressed pointers (object references represented as 32 bit offsets instead of 64-bit pointers) for optimized 64-bit performance with Java heap sizes less than 32gb.
```

```
   JAVA_OPTS="$JAVA_OPTS -XX:+UseCompressedOops"


#whole-heap operations, such as global marking, are performed concurrently with the application threads. This prevents interruptions proportional to heap
   JAVA_OPTS="$JAVA_OPTS -XX:+UseG1GC"


#Sets a target for the maximum GC pause time. This is a soft goal, and the JVM will make its best effort to achieve it.
   JAVA_OPTS="$JAVA_OPTS  -XX:MaxGCPauseMillis=500"


#GC Logging is recommended to be enabled because it is light weight
   JAVA_OPTS="$JAVA_OPTS  -verbose:gc -Xloggc:gc.log.`date +%Y%m%d%H%M%S` -XX:+PrintGCDetails -XX:+PrintGCTimeStamps"
else
   echo "JAVA OPTS already set in environment; overriding default settings with values: $JAVA_OPTS"
fi
[…]
```

2.5.7 Applikation installieren

Das ausgelieferte Artefakt (*.war File) muss in den Ordner <wildfly-home>/standalone/deployments kopiert werden. Dieses Verzeichnis wird vom Wildfly automatisch gescannt und darin enthaltene *.war Files werden deployed.

2.5.8 Wildfly starten

Wenn der Wildfly Applikationsserver nicht als Service installiert wurde muss er manuell gestartet werden. Dazu führt man den folgenden Befehl aus

```
<wildfly-home>/bin/standalone.sh -b=0.0.0.0 -bmanagement=0.0.0.0
```

Ansonsten muss der entsprechende Service gestartet werden. Dabei muss beachtet werden, dass Wildfly standardmässig nur auf Requests hört die von 127.0.0.1 ausgehen. Für den Serverbetrieb muss daher die Option –b=0.0.0.0 mitgegeben werden damit Wildfly auch auf Requests reagiert die nicht von localhost ausgehen. Statt 0.0.0.0 kann auch die genaue IP des hosts gebunden werden.

Sobald der Wildfly gestartet ist, kann die Standardseite unter http://127.0.0.1:8080 erreicht werden.

Hinweis: Wenn in Schritt 2.4.1 ein ebegu.properties File erstellt wurde mit dem Wildfly die benötigten Angaben für das Verbinden auf die Datenbank mitgegeben werden, so muss zudem der Parameter --properties=ebegu.properties mitgegeben werden.

```
<wildfly-home>/bin/standalone.sh --properties=ebegu.properties etc…
```

Um zu prüfen ob die Applikation ohne Fehler deployed werden konnte, kann das Logfile geöffnet werden. Dieses findet sich in folgendem Ordner:

```
<wildfly-home>/standalone/log/server.log
```

2.5.9 Standard-Ports

Standardmässig hört Wildfly auf dem Port 8080 für deployte Applikation. Port 9990 ist der Management-Port unter dem die Administration Console verfügbar ist. Wenn TLS konfiguriert wird sollten Applikationen zudem auch auf dem Port 8443 verfügbar sein.

2.5.10 Header Filtering sowie Response Status Code Filter

Unsere Applikation setzt für die Kommunikation zwischen dem Client und dem Serverteil einige Headers im http Request. Diese dürfen von der WAF nicht verändert oder ausgefiltert werden.

Name	Zweck	Beispielwert
`x-filename`	Bei Fileupload Filename übermitteln	„testfile.pdf“
`x-gesuchID`	Gesuch ID angeben	UUID, 072b3d31-b715-4d80-98f3-852d802fc1b
`x-forwarded-for`	Dokumentdownload nur für IP des ursprünglichen Requests zulassen	56.186.10.123
`x-xsrf-token`	um cross site request forgery Attacken zu unterbinden	uuid
`x-vorlagekey`	Vorlagenkey bei Fileupload	
`x-gesuchsperiode`	ID der Gesuchsperiode	
`x-progesuchsperiode`	Gibt an ob Upload pro Gesuchsperiode gilt	Boolean (true,false)
`x-ebegu-build-time`	Build Time des Servers	
`x-ebegu-version`	Version des Serverteils	1.0.5

Das E-BEGU Restinterface gibt teilweise bei Fehlern Informationen zurück welche im Client angezeigt werden müssen. Daher sollten 500 Statuscodes nicht mit einer generischen Fehlerseitenantwort ersetzt werden. Zumindest nicht wenn es sich um einen AJAX Request handelt.

Da die Applikation Filedownload/Upload ermöglicht, ist die maximale Requestgrösse auf 10 MB einzustellen.

2.5.11 Administration Console

Wildfly bietet ein Webinterface an über welches zum Einen der aktuelle Status des Wildfly eingesehen werden kann und zum anderen Anpassungen vorgenommen werden können. Damit diese Konsole von ausserhalb localhost erreichbar ist, muss beim Start -bmanagement=0.0.0.0 mitgegeben werden.

Die Admin Console kann dann unter http://hostname:9990 erreicht werden.

Für den Zugriff auf das Webinterface wird ein Wildfly-User Login benötigt. Um Wildfly-User interaktiv anzulegen wird folgendes Skript gestartet

```
$ <wildfl-home>/bin/add-user.sh
```

Alternativ kann der User auch nicht-interaktiv angelegt werden.

```
$ <wildfl-home>/bin/add-user.sh meinUserName meinPasswort –silent
```

Für Problemanalyse etc. sollte ein solcher Administrationsuser erzeugt werden.

2.5.12 Einsatz von HTTPS

Standardmässig laufen Applikation auf Wildfly über http Port 8080. Wenn der Datenverkehr über https laufen soll muss dies konfiguriert werden.

Dazu sind folgende Schritte nötig:

1. Erstellen eines Java Key Store Files mit dem benötigen Key:

```
$ keytool -genkeypair -alias serverkey -keyalg RSA -keysize 2048 -validity 7360
-keystore server.jks -keypass meinKeystorePasswort
-storepass meinKeystorePasswort
```

2. Das generierte server.jks File in den Ordner <wildfly-home>/standalone/configuration verschrieben.
3. Das stanalone.XML wie folgt anpassen (Fett = hinzufügen, Gelb = anpassen)

```
[File: <wildfly-home>/standalone/configuration/standalone.xml]


<security-realm name="ApplicationRealm">
    <server-identities>
        <ssl>
            <keystore path="server.jks" relative-to="jboss.server.config.dir" keystore-
password="meinKeystorePasswort" alias="serverkey"/>
        </ssl>
    </server-identities>
    <authentication>
        <local default-user="$local" allowed-users="*" skip-group-loading="true"/>
        <properties path="application-users.properties" relative-to="jboss.server.config.dir"/>
    </authentication>
    <authorization>
        <properties path="application-roles.properties" relative-to="jboss.server.config.dir"/>
    </authorization>
</security-realm>



[…]



<subsystem xmlns="urn:jboss:domain:undertow:3.0">
    <buffer-cache name="default"/>
    <server name="default-server">
        <http-listener name="default" socket-binding="http" redirect-socket="https"/>
```

```
        <https-listener name="default-https" security-realm="ApplicationRealm" socket-
binding="https"/>
        <host name="default-host" alias="localhost">
            <location name="/" handler="welcome-content"/>
            <filter-ref name="server-header"/>
            <filter-ref name="x-powered-by-header"/>
        </host>
    </server>
    <servlet-container name="default">
        <jsp-config/>
        <websockets/>
    </servlet-container>
    <handlers>
        <file name="welcome-content" path="${jboss.home.dir}/welcome-content"/>
    </handlers>
    <filters>
        <response-header name="server-header" header-name="Server" header-value="WildFly/10"/>
        <response-header name="x-powered-by-header" header-name="X-Powered-By" header-
value="Undertow/1"/>
    </filters>
</subsystem>

[…]
```

Nach einem Neustart sollte Wildfly nun die Standardseite unter https://localhost:8443/ anzeigen.

Mit dieser Konfiguration sollten Applikationen sowohl unter dem Port 8443 über https und unter dem Port 8080 über http erreicht werden können.

2.6 Webserver

Das ausgelieferte Artefakt ebegu-web-client.tar.gz muss auf dem Webserver entpackt und unter dem definierten Context verfügbar gemacht werden. Es kann ein beliebiger Webserver zum Einsatz kommen. Möglicherweise kann der Client sogar von der WAF zur Verfügung gestellt werden.

3 Applikationskontext und Pfade

Die Applikation besteht wie oben beschrieben aus zwei verschiedenen Artefakten. Der statische Webclient wird über einen Webserver zur Verfügung gestellt. Als Context-Pfad schlagen wir folgende Struktur vor.

https://{basisurl}/*

Alle Zugriffe auf diesen Context werden vom Webserver unter http://{webserverhost}:80/ behandelt. Alternativ kann dies direkt von der WAF übernommen werden.

Die Daten die der Client mit dem Server austauschen will und Aktionen die er durchführen muss werden vom Wildfly behandelt. Wir schlagen vor diesen unter dem Context

https://{basisurl}/ebegu/*

zur Verfügung zu stellen.

Zugriffe auf diesen Context werden vom Wildfly unter http://{wildflyhost}:8080/ebegu (bzw wenn SSL Konfiguriert wurde unter https://{wildflyhost}:8443/ebegu) behandelt.

Für die Anbindung an die Datenbank brauchen wir wie erwähnt ein Schema welches vom Wildfly unter jdbc:mysql://meinDatenbankserver:3306/ebegu erreicht werden kann.

Für die Anbindung an das Filesystem zur Ablage der hochgeladenen Dokumente muss ein von Wildfly zu erreichender Verzeichnispfad definiert werden.

4 Anhang

4.1 Informationen zum Builden des Projekts

Das Projekt ist als Standard-Maven-Projekt aufgesetzt und kann mittels mvn install gebuildet werden.

Da die Libraries der DV Bern (https://github.com/dvbern/date-helper und weitere) aktuell in keinem öffentlichen Maven Repository verfügbar sind, müssen diese vorgängig lokal ausgecheckt und gebuildet werden.