GOVERNIKUS

GOVERNIKUS
DATA BOREUM®
Web Edition

Samples
Governikus DATA Boreum WebEdition

Governikus DATA Boreum WebEdition, Version 10.9.0

Dokumentenversion: 10.9.0_0

Inhaltsverzeichnis

1 Einführung

Das Handbuch Samples der Governikus DATA Boreum WebEdition bietet Beispiele zur einfacheren Integration in eine Fachanwendung. Die Beispiele werden in dieser ZIP-Datei ausgeliefert:

`GovernikusDataBoreumWebEditionSample-<Releasenummer>.zip`

GovernikusDataBoreumWebEditionSample

Die WAR-Datei `GovernikusDataBoreumWebEditionSample.war` stellt für die Governikus DATA Boreum WebEdition eine Aufrufseite zur Verfügung. Um die WebEdition zu testen, sollten Sie die War-Datei `GovernikusDataBoreumWebEditionSample.war` und die WAR-Datei `GovernikusDataBoreumWebEdition.war` auf demselben Rechner installieren und ausführen. Die URLs für diese Tests sind mit der Serveradresse `localhost` vorbelegt. Die URLs können angepasst werden, siehe Kapitel 2.2.

Hinweis: Im Folgenden wird die Governikus DATA Boreum WebEdition für die bessere Lesbarkeit mit WebEdition abgekürzt.

2 Beispiele

Inhalt

Die bereitgestellte Archivdatei hat den Namen:

`GovernikusDataBoreumWebEditionSample-<Releasenummer>.zip`

Entpacken Sie die Archivdatei. Sie enthält den Ordner Samples, in dem Sie diese Dateien finden:

- `GovernikusDataBoreumWebEditionSample_Sources.zip`
- `GovernikusDataBoreumWebEditionSample.war`

Der Ordner Samples

In diesem Ordner sind vorbereitete Beispiele mit Pflichtparametern enthalten, um das Ausprobieren und die Einbindung der WebEdition zu erleichtern. Diese Beispiele sind im nachfolgenden Abschnitt 2.1 erläutert.

2.1 Samples

GovernikusDataBoreumWebEditionSample.war

Die WAR-Datei `GovernikusDataBoreumWebEditionSample.war` enthält vorbereitete Beispiele (Servlets), um das Ausprobieren und die Einbindung der WebEdition zu erleichtern. Die Servlets sind auch als Quellcode in dieser Datei beigelegt:

`GovernikusDataBoreumWebEditionSample_Sources.zip`.

Startseite

Auf der Startseite der `GovernikusDataBoreumWebEditionSample` stehen links die einzelnen Funktionalitäten `Signieren`, `Verschlüsseln` und `Entschlüsseln` bereit. Bitte beachten Sie, dass die URL per Default auf `localhost` eingestellt ist. Das Ändern des Servernamens `localhost` ist im Kapitel 2.2 beschrieben.

Wenn Sie auf dieser HTML-Seite auf einen der Links Signieren, Ver- oder Entschlüsseln klicken, wird ein Browser-spezifisches Dialogfenster geöffnet, in dem Sie entscheiden können, ob Sie die WebEdition aufrufen wollen. Bitte beachten Sie, dass der WebEdition bei diesem Test keine optionalen Parameter übergeben werden (beispielsweise ohne Datei und ohne Schlüssel).

RCServlet

Dieses Servlet (Quelldatei `RCServlet.java`) ist eine beispielhafte Implementierung zur Annahme von Return-Codes. Dieses Servlet nimmt die folgenden Parameter entgegen:

- `returnCode`: Angabe des Return-Codes
- `returnCodeDescription`: Beschreibung des angegebenen Return-Codes.
- `transactionID`: Zusätzlicher `GET`-Parameter aus der `RC_URL`, zur Übermittlung einer eindeutigen Transaktionsnummer. Dieser Parameter wird nur angezeigt, aber nicht ausgewertet.
- `param0 … param<n>`: Parameter, die zur Erstellung der RC-Description verwendet wurden (z.B. Dateiname), sofern vorhanden.

Das Servlet stellt die 20 letzten übermittelten Return-Codes auf einer HTML-Seite dar, die automatisch alle 10 Sekunden aktualisiert wird.

FileSaveServlet

Dieses Servlet (Quelldatei `FileSaveServlet.java`) beinhaltet eine beispielhafte Implementierung zur Annahme der signierten Datei durch die WebEdition. Dieses Servlet nimmt ein `http POST form-Data/Multipart Request` mit der signierten Datei entgegen und speichert diese serverseitig. Der serverseitige Speicherort der übergebenen Dateien ergibt sich wie folgt:

Das `FileSaveServlet` prüft ob es in dem `WEB-INF/classes` Ordner eine Konfigurationsdatei `FileSaveServlet.properties` gibt und liest die Einstellung `root.folder` ein. Des Weiteren wird der `POST`-Parameter `storage_dir` ausgewertet. Die nachfolgende Tabelle erläutert die Kombinationsmöglichkeiten:

root.folder (aus FileSaveServlet .properties)	**storage_dir** (POST-Parameter)	**Anmerkungen**	**Speicherort**
Vorhanden	Vorhanden	`storage_dir` muss als relativer Pfad angegeben sein. Es werden keine "../" o. ä. im Pfad akzeptiert.	`<root.folder>/<storage_dir>`
Vorhanden	Nicht vorhanden		`<root.folder>/`
Nicht Vorhanden	Vorhanden	`storage_dir` sollte als absoluter Pfad angegeben sein, da ansonsten dieser Pfad als relativ zum Arbeitsverzeichnis des Webservers interpretiert wird.	`<storage_dir>/`
Nicht Vorhanden	Nicht vorhanden	Es wird das aktuelle Java-Temp Verzeichnis genommen.	`<java.io.tmpdir>`

Hinweis: Auch wenn auf dem Zielsystem als Betriebssystem Microsoft Windows eingesetzt wird, muss der Pfad unter `root.folder` und `storage_dir` mit "/" als Trennzeichen angegeben werden (z.B. d:/input/signed).

Als Antwort wird eine HTML-Seite mit einigen Logging-Ausgaben und folgenden XML-Elementen erstellt:

- `<saved_file>`: Angabe des voll qualifizierten Dateinamens auf dem Server.
- `<error>`: Beschreibung des Fehlers beim serverseitigen Speichern der Datei.

- `<fileexist/>`: Wird innerhalb des `<error>`-Elements gesendet, wenn die Datei serverseitig schon existiert.

Die WebEdition wertet auch diese XML-Elemente aus und gibt ggf. den ReturnCode `CANNOT_SAVE_OUTPUT` zurück.

Hinweis: Die Kodierung einer Datei, die über einen `http POST form-Data/Multipart Request` gesendet wird, ist in der RFC 1867 nachzulesen. Die Kodierung des Dateinamens, der mit gesendet wird, ist in der RFC 1522 nachzulesen.

Auszug aus der RFC 1522:

```
encoded-word = "=?" charset "?" encoding "?" encoded-text "?="
```

Hinweis: Das aktuelle Java-Temp Verzeichnis wird durch `System.getProperty(java.io.tmpdir)` ermittelt.

Für den Download und das Speichern der Datei wird eine Funktionsbibliothek `HttpReceive` mitgeliefert, die diese Kernfunktionalität auch für Fremdanwendungen zur Verfügung stellt.

Hinweis: Die Klasse `HttpReceive` wird nur als `.class` Datei im `commons_http_post_request JAR` mitgeliefert.

Da sowohl `jsp` als auch `.java` als Quellcode zur Verfügung gestellt werden, können die Dateien an die jeweiligen Anforderungen angepasst werden und ggf. auch in unterschiedlichen Ausprägungen, mit vom obigem Vorschlag abweichenden Vorbelegungsanforderungen, implementiert werden.

Hinweis: Die Beispielklassen `FileSaveServlet.java` und `RCServlet.java`, welche im Quellcode mitgeliefert werden, können mit dem mitgelieferten `compile.bat skript` kompiliert werden.

2.2 Default-Einstellung localhost ändern

Wenn Sie die Tests mit der Datei auf einem anderen Server durchführen wollen, müssen Sie die Datei `GovernikusDataBoreumWebEditionSample.war` entpacken.

index.html

In entpackten Verzeichnis finden Sie die Datei `index.html`. Hier können Sie die URLs der Funktionsaufrufe Signieren, Verschlüsseln und Entschlüssen mit einem Editor ändern.

testdata

Im Unterverzeichnis `testdata` finden Sie die Dateien

- `decrypt.gov-ws`
- `encrypt.gov-ws` und

- `sign.gov-ws`

Editieren Sie die Dateien und ändern Sie die dort hinterlegten Beispiel-URLs.

3 Administration

Systemanforderungen

Für den Application-Server, der die WebEdition Samples bereitstellt, müssen folgende Punkte beachtet werden:

- Die WebEdition muss lokal auf demselben Application-Server installiert sein, da die URLs die Default-Einstellung `localhost` haben. Sie können die URLs anpassen, wie in Kapitel 2.2 erklärt.
- Das Dateisystem muss mit dem Zeichensatz "ISO-8859-1" eingestellt sein, damit das `FileSaveServlet` aus `GovernikusDataBoreumWebEditionSample.war` die Dateien mit Umlauten richtig speichern kann.

WebEdition Samples deployen

Die WAR-Datei

- `GovernikusDataBoreumWebEditionSample.war,`

muss in das Deploy-Verzeichnis des Webservers kopiert werden. Das Deploy-Verzeichnis ist bei den unterstützten Web-Servern in den folgenden Unterverzeichnissen zu finden.

JBoss

`server/standalone/deployments/`

Tomcat

`webapps/`

Sobald der Webserver gestartet wurde, werden die Beispiele der WebEdition unter der Adresse `http://<hostname>:<Portnummer>/ GovernikusDataBoreumWebEditionSample` zur Verfügung gestellt.

4 Anhang Beispieldateien

Im Verzeichnis `./Samples/testdata` finden Sie die Dateien, mit denen Sie die WebEdition beispielhaft aufrufen können.

4.1 Beispielaufrufe nur mit Pflichtparametern

sign.gov-ws

Die Datei `sign.gov-ws` enthält diese Zeilen als Pflichtparameter:

```
signer.process=sign
```

```
signer.rcURL=http://localhost:8080/GovernikusDataBoreumWebEditionSample/rcCode?transactionID=xx.xx.xx.xx
```

Die Beispielaufrufe in den Dateien `encrypt.gov-ws` und `decrypt.gov-ws` sind analog aufgebaut.

4.2 Beispielaufrufe mit allen Parametern

Es folgen die Dateien für Beispielaufrufe mit allen Parametern. Bitte beachten Sie, dass Sie diese Dateien editieren müssen, damit Sie die Parameter an Ihre Anforderungen anpassen können.

4.2.1 sign.gov-ws mit allen Parametern

Die folgende Datei zeigt ein ausführliches Beispiel.

```
#================================================
signer.process=sign
signer.rcURL=http://localhost:8080/GovernikusDataBoreumWebEditionSample/rcCode?transactionID=xx.xx.xx.xx

# =========
# === 0 ===
# =========

# skipCompletedStep - ist default | firstStep | lastStep
signer.processPolicy = firstStep

signer.configurationFile =
http://localhost:8080/GovernikusDataBoreumWebEditionSample/configuration/configuration.xml

signer.licenceFile =
http://localhost:8080/GovernikusDataBoreumWebEditionSample/licence/dynamic_SignLicence.xml

# ======================
# === 1. Dateiauswahl ===
# ======================

# Verzeichnis
```

```
#signer.source = D:/test/boreum_we/test

# url-datei#datei-Pfad
signer.source =
http://localhost:8080/GovernikusDataBoreumWebEditionSample/test/Bild.TIFF#D
:/test/boreum_we/test/PDF.pdf

# url-dateiname-auf-dem-server|dateiname-lokal
# signer.source =
http://localhost:8080/GovernikusDataBoreumWebEditionSample/test/PDF.pdf|lok
ale_PDF.pdf

# url-dateiname-auf-dem-server|dateiname-lokal|sha256
# signer.source =
http://localhost:8080/GovernikusDataBoreumWebEditionSample/test/PDF.pdf|lok
ale_PDF.pdf|9d002a17a06c26f14eef890381d35edea6ce339b6a736e0d12bb24a8fe61ba8
3

# ====================
# === 2. Optionen ===
# ====================

# === Folgende Parameter überschreiben entsprechende <options> in
configuration.xml ===

# == Parameter überschreibt <signatureType> in configuration.xml
# == detachedPKCS7 - ist default | envelopedPKCS7
signer.signformat = envelopedPKCS7

# == Parameter überschreibt <pdfSignatureType> in configuration.xml
# == noPdfInline | pdfInline - ist default
signer.pdfSignatureType = noPdfInline

# == Parameter überschreibt <pdfSignatureReason> in configuration.xml
signer.advancedPdfReason = Test Boreum WE

# == Parameter überschreibt <xmlSignatureType> in configuration.xml
# == noXmlInline - default | xmlInline
signer.xmlSignatureType = xmlInline

# == Parameter überschreibt <isExternalTimeStamp> in configuration.xml
# == yes | no - ist default
signer.createtimestamp = yes



# ==================================
# === 4. Zielverzeichnis wählen ===
# ==================================

# == Parameter "signer.targetFolderType" überschreibt
<sign><selecttargetfolder><targetfolder.type> in configuration.xml
# == "signer.targetFolderType = sameAsSourceFolder" - ist default, oder
"signer.dest=<Pfad>"
```

```
signer.dest = D:/test/signer_we/dest
#================================================
```

4.2.2 decrypt.gov-ws mit allen Parametern

Die folgende Datei zeigt ein ausführliches Beispiel.

```
#================================================
signer.process=decrypt
signer.rcurl=http://localhost:8080/GovernikusDataBoreumWebEditionSample/rcC
ode?transactionID=xx.xx.xx.xx

# =========
# === 0 ===
# =========

# === skipCompletedStep - ist default | firstStep | lastStep
signer.processPolicy = firstStep

signer.configurationFile =
http://localhost:8080/GovernikusDataBoreumWebEditionSample/configuration/co
nfiguration_crypt.xml

signer.licenceFile =
http://localhost:8080/GovernikusDataBoreumWebEditionSample/licence/dynamicD
ecryptLicence_erweitert.xml

# ========================
# === 1. Dateiauswahl ===
# ========================

# === Verzeichnis bzw. Datei angeben:
# signer.source = <Pfad>
signer.source = D:/test/boreum_we/encrypt_src/Quelle 1/Test-
1.zip.p7m#D:/test/boreum_we/encrypt_src/Quelle_2/Test.3.TIFF.p7m

# ============================
# === 2. Schlüssel wählen ===
# ============================

# === yes | no - default
# signer.usepassword = yes

# ================================
# === 3. Zielverzeichnis wählen ===
# ================================

# === yes | no - default
# signer.extractziparchiv = no

# === "signer.targetFolderType = sameAsSourceFolder" - ist default, oder
"signer.dest = <Pfad>"

# signer.dest = <Pfad_zur_dest>
```

```
# signer.dest =
http://localhost:8080/GovernikusDataBoreumWebEditionSample/fileSave?storage
_dir=D:/test/boreum_we/encrypt_dest_server
# signer.dest = D:/test/boreum_we/encrypt_dest

#=============================================
```

4.2.3 encrypt.gov-ws mit allen Parametern

Die folgende Datei zeigt ein ausführliches Beispiel.

```
#=============================================

signer.process=encrypt
signer.rcurl=http://localhost:8080/GovernikusDataBoreumWebEditionSample/rcC
ode?transactionID=xx.xx.xx.xx

# =========
# === 0 ===
# =========

# skipCompletedStep - ist default | firstStep | lastStep
signer.processPolicy = firstStep

signer.configurationFile =
http://localhost:8080/GovernikusDataBoreumWebEditionSample/configuration/co
nfiguration_crypt.xml

signer.licenceFile =
http://localhost:8080/GovernikusDataBoreumWebEditionSample/licence/dynamicE
ncryptLicence_erweitert.xml

# ======================
# === 1. Dateiauswahl ===
# ======================

# <Datei-URL>#<Datei-Pfad>#<Verzeichnis-Pfad>
signer.source =
http://localhost:8080/GovernikusDataBoreumWebEditionSample/test/PDF.pdf#htt
p://localhost:8080/GovernikusDataBoreumWebEditionSample/test/Bild.TIFF

# ==========================
# === 2. Schlüssel wählen ===
# ==========================

# === yes | no - default
# signer.usepassword = yes



# ================================
# === 3. Zielverzeichnis wählen ===
# ================================

# === yes | no - default
```

```
# signer.createziparchivbeforeencrypt= yes

# == "signer.targetFolderType = sameAsSourceFolder" - ist default, oder
"signer.dest = <Pfad>"

# signer.dest = <Pfad_zur_dest>/dest
signer.dest =
http://localhost:8080/GovernikusDataBoreumWebEditionSample/fileSave?storage
_dir=D:/test/boreum_we/encrypt_dest_server
#================================================
```

4.3 Tabelle der Parameter für das Signieren

In dieser Tabelle sind die zusätzlichen Parameter für das Signieren erklärt. Wird keine Konfiguration mit Parameter angeben, wird der Default-Wert genutzt. Wird eine Konfiguration mit Parameter angegeben, überschreibt dies den Default-Wert.

Dialog-Schritte	Default-Wert	Konfiguration	Parameter
-	skipCompletedStep	-	signer.processPolicy
-	-	-	signer.configurationFile
			signer.licenceFile
-	-	-	signer.source
Optionen	detachedPKCS7	<signatureType>	signer.signformat
	pdfInline	<pdfSignatureType>	signer.pdfSignatureType
	-	<pdfSignatureReason>	signer.advancedPdfReason
	noXmlInline	<xmlSignatureType>	signer.xmlSignatureType
	no	<isExternalTimeStamp>	signer.createtimestamp
Zielverzeichnis wählen	sameAsSourceFolder	<sign> <selecttargetfolder> <targetfolder.type>	signer.targetFolderType
	-	<sign> <selecttargetfolder> <targetfolder.path/>	signer.dest

Tabelle 1: Default-Werte, Konfigurationsnamen und Parameter für das Signieren