CF13.4 Session

Konfiguration und Setup

Vor dem Start muss das Plugin konfiguriert werden. Dazu muss im Verzeichnis "`plugins/sessionPlugin/src/main/resources`" die Datei "`settings.conf.dist`" als "`settings.conf`" kopiert und diese Datei angepasst werden.

Das Session Plugin verwendet eine eigene MySQL-Datenbank, um die Daten der Sessions zu speichern. Obwohl es prinzipiell möglich ist, das selbe Datenbank-Schema zu verwenden wie der Simplifier selbst, wird stark empfohlen ein eigenes Datenbankschema für das Plugin anzulegen, um Namenskonflikte zu vermeiden. Dieses kann allerding ohne Probleme auf dem selben DB-Server liegen, sofern dies die Netzwerktopologie zulässt.

Zusätzlich kann ein "expire" Timeout (in Minuten) definiert werden, welches die maximale Zeit seit dem letzten Zugriff ist, nach dem eine Session automatisch invalidiert wird.

settings.conf

```
database_mysql {
  user: "simplifier"
  pass: "simplifier"
  host: "localhost"
  port: 3306
  database: "simplifier_session"
}

sessionPlugin {
  interactive: false
  expireTimeout: 60 # Minutes
}

...
```

Nachdem die Datenbank-Verbindung konfiguriert wurde, muss das Datenbankschema für das Plugin angelegt werden.

Dazu kann einfach im Plugin-Verzeichnis der SBT/Activator Befehl "`run admin schemify`" ausgeführt werden.

Die benötigten Tabellen und Indizes sollten nun in der konfigurierten MySQL Tabelle angelegt worden sein.

Plugin-Ausführung

Das Plugin befindet sich im Verzeichnis `plugins/sessionPlugin` und kann dort mit SBT/Activator über den "`run`" Befehl gestartet werden. Wird über STDIN der Befehl "`stop`" eingegeben, wird die Ausführung des Plugins regulär beendet (und vom AppServer abgemeldet), sofern der Konfigurationsparameter `interactive = true` angegeben wurde.

Um die Log-Ausgabe zu konfigurieren (oder in eine Datei auszugeben), kann die Logback-Konfigurationsdatei "`plugins/sessionPlugin/src/main/resources/logback.xml` angepasst werden.

Berechtigungen und Benutzer

Das Session Plugin stellt keine Berechtigungsobjekte bereit und kann von jedem gültigen Simplifier-User benutzt werden.

Jede erstellte Session wird an den User gebunden, mit dem die Session erzeugt wurde und es kann nur von diesem User aus auf die Session zugegriffen werden. Das transferieren einer Session auf einen anderen Benutzer ist nicht möglich.

Datenmodell

Das Session Plugin verwaltet "Session" Objekte, in denen Daten zu einzelnen User Sessions gespeichert werden.

Datenbank Tabelle **session**:

Feld	Typ	Bedeutung
id	INT	Primärschlüssel
sessionid	VARCHAR	Zufällig erstellte UUID, die eine Session eindeutig identifiziert.
user	INT	Zu der Session zugeordnete UserID, der die Session gehört
expires	DATETIME	Zeitpunkt, ab dem die Session "abläuft", also nicht mehr zugänglich ist. Jeder Zugriff auf die Session verlängert diesen Wert
data	BLOB	JSON Daten der Session

REST Schnittstellen

Es sind REST-Schnittstellen für CRUD Operationen auf den Session Objekten vorhanden. Außerdem gibt es Operationen zum Lesen, Schreiben und Löschen von einzelnen Keys in den Daten einer Session, so dass nicht jedes mal das Komplette JSON Objekte mit allen Daten übertragen werden muss.

Alle Operationen sind sowohl als Slot und als HttpSlot vorhanden.

Slot	HttpSlot	Operation
sessionCreate	sessionCreateHttp	Session Erstellen
sessionWrite	sessionWriteHttp	Daten einer Session schreiben/überschreiben
sessionFetch	sessionFetchHttp	Daten einer Session abfragen
sessionDelete	sessionDeleteHttp	Session löschen
sessionKeyWrite	sessionKeyWriteHttp	Einzelnen Key im JSON-Objekt der Daten einer Session schreiben/überschreiben
sessionKeyFetch	sessionKeyFetchHttp	Einzelnen Key im JSON-Objekt der Daten einer Session abfragen
sessionKeyDelete	sessionKeyDeleteHttp	Einzelnen Key im JSON-Objekt der Daten einer Session löschen

Jede Rückgabe einer Slot-/HttpSlot-Operation enthält immer mindestens die folgenen Werte:

JSON Key	Type	Bedeutung
`success`	Boolean	Erfolg der Operation; `true` = erfolgreich, `false` = fehlerhaft
`message`	String	Erfolgsnachricht oder Fehlernachricht mit Fehlerbeschreibung

Hinweis:

Wird auf eine Session zugegriffen, die es nicht (mehr) gibt, die bereits abgelaufen ist, oder die nicht dem aktuellen User zugeordnet ist, so wird die Anfrage mit einer jeweiligen Fehlermeldung und `success=false` beendet.

Wird eine Key-Operation (überschreiben, lesen, löschen) auf einen nicht (mehr) zugewiesenen Key ausgeführt, so wird implizit der JSON-Wert `null` für den Key angenommen, so dass Lese- und Lösch-Operationen auf nicht existierende Keys trotzdem erfolgreich ausgeführt werden (vorrausgesetzt es handlet sich um eine gültige Session, siehe Hinweis darüber).

Session erstellen: http://localhost:8080/client/1.0/PLUGIN/sessionPlugin/**sessionCreateHttp**

Input:

Nichts
Output:

Key	Typ	Bedeutung
`sessionid`	String	SessionId der erstellten Session

Beispiel:

```
{
    "sessionid":
"7e7fcc7e-5528-4e44-9190-7f511130355d"
}
```

Session-Daten schreiben: http://localhost:8080/client/1.0/PLUGIN/sessionPlugin/**sessionWriteHttp**

Input:

Key	Typ	Bedeutung
`sessionid`	String	SessionId der Session
`sessionData`	JSON Object	Daten der Session (Muss ein JSON-**Object** sein, kein String oder Array o.ä.), die Values dürfen selbst wieder beliebige JSON Daten enthalten

Beispiel:

```
{
    "sessionid":
"7e7fcc7e-5528-4e44-9190-7f511130355d",
    "sessionData": {
        "key": "value",
        "intkey": 123,
        "objectkey": {
            "foo": "bar"
        }
    }
}
```

Output:

Nichts

Session-Daten lesen: http://localhost:8080/client/1.0/PLUGIN/sessionPlugin/**sessionFetchHttp**

Input:

Key	Typ	Bedeutung
`sessionid`	String	SessionId der Session

Beispiel:

```
{
    "sessionid":
"7e7fcc7e-5528-4e44-9190-7f511130355d"
}
```

Output:

Key	Typ	Bedeutung
result	JSON Object	Vollständige Daten der Session

Beispiel:

```
{
    "result": {
        "key": "value",
        "intkey": 123,
        "objectkey": {
            "foo": "bar"
        }
    }
}
```

Session löschen: http://localhost:8080/client/1.0/PLUGIN/sessionPlugin/**sessionDeleteHttp**

Input:

Key	Typ	Bedeutung
`sessionid`	String	SessionId der Session

Beispiel:

```
{
    "sessionid":
"7e7fcc7e-5528-4e44-9190-7f511130355d"
}
```

Output:

Nichts

Session-Daten Key schreiben: http://localhost:8080/client/1.0/PLUGIN/sessionPlugin/**sessionKeyWriteHttp**

Input:

Key	Typ	Bedeutung
`sessionid`	String	SessionId der Session
`key`	String	Key innerhalb des JSON Objects der Session Daten
`sessionData`	JSON Value	Neuer Wert für den Key in den Daten der Session (kann belibieger JSON Wert sein, also String, Array, Int, Object, ...)

Hinweis:

Existiert bereits ein Wert für den gegebenen Key, so wird dieser überschrieben. Existier noch kein Wert, so wird dieser hinzugefügt.

Beispiel:

```
{
    "sessionid":
"7e7fcc7e-5528-4e44-9190-7f511130355d",
    "key": "newkey",
    "sessionData": {
        "testkey": "value",
        "intkey": 123,
        "objectkey": {
            "foo": "bar"
        }
    }
}
```

Output:

Nichts

Session-Daten Key lesen: http://localhost:8080/client/1.0/PLUGIN/sessionPlugin/**sessionKeyFetchHttp**

Input:

Key	Typ	Bedeutung
`sessionid`	String	SessionId der Session
`key`	String	Key innerhalb des JSON Objects der Session Daten

Beispiel:

```
{
    "sessionid":
"7e7fcc7e-5528-4e44-9190-7f511130355d",
    "key": "testkey"
}
```

Output:

Key	Typ	Bedeutung
`result`	JSON Value	JSON-Daten des Keys innerhalb des JSON Objects der Session. Ja nach Struktur der Sessiondaten kann hier jeder gültige JSON Wert zurückgegeben werden (String, Int, Object, Array)

Hinweis:

Wurde dem Key kein Wert zugeordnet, so wird der JSON Typ "null" zurückgegeben.

Beispiel:

```
{
    "result": "value"
}
```

Session-Daten Key löschen: http://localhost:8080/client/1.0/PLUGIN/sessionPlugin/**sessionKeyDeleteHttp**

Input:

Key	Typ	Bedeutung
`sessionid`	String	SessionId der Session
`key`	String	Key innerhalb des JSON Objects der Session Daten

Beispiel:

```
{
    "sessionid":
"7e7fcc7e-5528-4e44-9190-7f511130355d",
    "key": "keytodelete"
}
```

Output:

Nichts

Hinweis:

Wurde dem Key kein Wert zugeordnet, so ändert sich nichts und die Operation wird als erfolgreich angezeigt.