Server - Client Kommunikation

ÄNDERUNGEN:

Release 3.1:

- Drittes Argument zum Headless client Hinzugefügt

Release 3:

- Signature
- Testuser
- Nodename

Release 2:

- Usernamelänge
- Chatcommand
- Continuous Integration - Gradle und Jar
- Headless Communication
- Command Error Channel

ÜBERSICHT

Allgemeines

In diesem Release braucht ihr euch noch nicht am Server anmelden. Wenn ein neuer User das Modell anfordert wird automatisch ein neuer User erzeugt. Da wir keine Anmeldung vorgeschaltet haben, bitten wir euch um ein wenig Disziplin.

MQTT Broker

Der Broker ist unter tcp://se1.cs.uni-kassel.de:1883 mittels eines MQTT Clients zu erreichen.

Username

Bitte wählt euch einen eigenen eindeutigen. **Die maximale Länge beträgt 50 Zeichen.**

MQTT

In diesem Semester wird, wie in der Vorlesung angekündigt, MQTT zum Einsatz kommen. MQTT ist ein schlankes Protokoll für einfache Kommunikation zwischen verschiedenen Systemen. Nachrichten werden an einen Broker geschickt und in verschiedenen Channels veröffentlicht. Clients melden sich an Channels an, um Nachrichten zu erhalten.

Server

Der Spielserver ist als Client am SE1 MQTT Broker angemeldet und sendet Nachrichten in verschiedenen Channels. Beispielsweise kann das Datenmodell per Nachricht angefordert werden und wird anschließend in einem eigens dafür vorgesehenen Channel veröffentlicht. Der Server nimmt ebenfalls Befehle für Spielzüge entgegen. Der Server wird alle 3 Stunden neugestartet (0,3,6,9,12,15,18,21 Uhr).

Clients

Die von den Teams entwickelten Clients müssen sich am MQTT Broker anmelden und Nachrichten senden und empfangen, um mit dem Spielserver zu kommunizieren.

Channels

Für Anfragen an den Spielserver gibt es den Channel */ist/command* in den alle Clients ihre Nachrichten schicken.

Fordert ein Client das Datenmodell vom Spielserver an, so antwortet dieser im Channel */ist/model/<<username>>.* Im Channel */ist/model/all erhalten alle Clients* Updates des Datenmodells. Alle Nachrichten haben eine HistoryID die die Reihenfolge festlegt.

Befehle

Die für das erste Release benötigten Befehle befinden sich in der Tabelle im Anhang. In der ersten Spalte befindet sich jeweils ein Beispiel für eine entsprechende Nachricht. Die Befehle und Antworten werden im JSON Format ausgetauscht

Continuous Integration

Euer Grade Script muss einen Task headlessclientJar enthalten. Dieser soll nach dem Beispiel des fatJar Tasks ein Jar-File bauen, dass alle nötigen Abhängigkeiten enthält. Auf keinen Fall darf diese Jar Grafiken oder andere unnötige große Dateien enthalten, die nicht für den Betrieb notwendig sind.

Das erstellte Jar-File soll *teamXheadlessclient.jar* heißen und mit dem Befehl

java -jar teamXheadlessclient.jar host username ‚Pfad zum Privatekey'

Beispiel:

java -jar teamXheadlessclient.jar „tcp://se1.cs.uni-kassel.de:1883" se_ai /home/dude/se_ai.key

gestartet werden können. Unser Bildserver wird in bestimmten Zeitabständen eure Repositories prüfen, ob dort im Branch release eine neue Version vorliegt, wenn ja führt er den Task headlessclientJar aus und startet diese im Anschluss.

Headless Communication

Wenn ihr mit eurem headless Client kommunizieren wollt, könnt ihr dies über alle Channel tun, die im namespace /ist/headless/teamx/* liegen.

Chat

Für den Chat müsst ihr euch am Channel /ist/chat registrieren und auf die dort auflaufenden Nachrichten lauschen. Nachrichtenformat nach Commanddoku.

Command Error Channel

Wenn bei der Ausführung eines Commands ein Fehler auftritt wird im Channel /error eine Fehlermeldung erzeugt.

Signature

Ab dem 27.6. verarbeitet der Server nur noch signierte Nachrichten (Commands). Die Signierung muss auf die selbe Art erfolgen wie im umgekehrten fall beim model.

Testuser

Mittels des Testuser-Commands können Testuser angelegt werden, max 2 Pro User. Diese müssen dann die selben Keys wie der ursprüngliche User verwenden. Das Testufer command benötigt jetzt schon eine Signatur.

Nodename

Die Nodes haben nun eindeutige Namen.

Command - Beispiel	Properties	Parameter	Beschreibung
{"class":"de.uniks.se1.ist.model.Command","id":".C1","prop":{"name":"getModel","user":"honk"}}	"name"="getModel" , "user"=$username	keine	Mit dem getModel-Command fordert man den Server auf im Topic /model/$username einmal das gesamte model zu publizieren
{"class":"de.uniks.se1.ist.model.Command","id":".C2","prop":{"name":"bid","user":"honk","commandParameters":[{"class":"de.uniks.se1.ist.model.CommandParameter","id":".C3","prop":{"name":"orderID","value":"1234","command":{"id":".C2","class":"de.uniks.se1.ist.model.Command"}}},{"class":"de.uniks.se1.ist.model.CommandParameter","id":".C4","prop":{"name":"value","value":"42","command":{"id":".C2","class":"de.uniks.se1.ist.model.Command"}}}]}}	"name"="bid" , "user"=$username	"orderID"="1234" ,"value"="42"	Mit dem bid-Command bietet man auf Aufträge
{"class":"de.uniks.se1.ist.model.Command","id":".C5","prop":{"name":"shuttleGoto","user":"processing4thewin","commandParameters":[{"class":"de.uniks.se1.ist.model.CommandParameter","id":".C6","prop":{"name":"shuttleID","value":"ist-model.S1","command":{"id":".C5","class":"de.uniks.se1.ist.model.Command"}}},{"class":"de.uniks.se1.ist.model.CommandParameter","id":".C7","prop":{"name":"destinationNodeID","value":"ist-model.N56","command":{"id":".C5","class":"de.uniks.se1.ist.model.Command"}}}]}}	"name"="shuttleGoto" , "user"=$username	"shuttleID"="ist-model.S1" ,"destinationNodeID"="ist-model.N56"	Das goto Command beordert ein shuttle zum nächsten Node
{"class":"de.uniks.se1.ist.model.Command","id":".C8","prop":{"name":"shuttleLoad","user":"honk","commandParameters":[{"class":"de.uniks.se1.ist.model.CommandParameter","id":".C9","prop":{"name":"shuttleID","value":"ist-model.S1","command":{"id":".C8","class":"de.uniks.se1.ist.model.Command"}}},{"class":"de.uniks.se1.ist.model.CommandParameter","id":".C10","prop":{"name":"goodID","value":"ist-model.G16","command":{"id":".C8","class":"de.uniks.se1.ist.model.Command"}}}]}}	"name"="shuttleLoad" , "user"=$username	"shuttleID"="ist-model.S1" ,"goodID"="ist-model.G56"	Ware einladen
{"class":"de.uniks.se1.ist.model.Command","id":".C11","prop":{"name":"shuttleUnload","user":"honk","commandParameters":[{"class":"de.uniks.se1.ist.model.CommandParameter","id":".C12","prop":{"name":"shuttleID","value":"ist-model.S1","command":{"id":".C11","class":"de.uniks.se1.ist.model.Command"}}}]}}	"name"="shuttleUnload" , "user"=$username	"shuttleID"="ist-model.S1"	Ware ausladen
{"class":"de.uniks.se1.ist.model.Command","id":".C13","prop":{"name":"chat","user":"honk","commandParameters":[{"class":"de.uniks.se1.ist.model.CommandParameter","id":".C14","prop":{"name":"text","value":"Bing mal Kaffee mit","command":{"class":"de.uniks.se1.ist.model.Command","id":".C13"}}},{"class":"de.uniks.se1.ist.model.CommandParameter","id":".C15","prop":{"name":"toUser","value":"lennert","command":{"class":"de.uniks.se1.ist.model.Command","id":".C13"}}}]}}	"name"="shuttleUnload" , "user"=$username	"text"=$derZuSendeneText, "toUser"=$Username für Privatechat	Chat im Channel /ist/chat
{"class":"de.uniks.se1.ist.model.Command","id":".C1","prop":{"name":"testuser","user":"honk","commandParameters":[]}}	"name"="testuser" , "user"=$username	keine	

Links

MQTT: http://mqtt.org

JSON: http://www.json.org

SE1 MQTT Broker: tcp://se1.cs.uni-kassel.de:1883