Übung 2: Chatroom mit *servlet*

In dieser Übung soll das Chatroom auf der Basis der Servlet-Technologie implementiert werden. Das Servlet soll HTTP Mitteilungen über verschiedene Themen entgegen nehmen, speichern und bearbeiten. Das Servlet soll die bekannten Schnittstellen `IChatRoom` und `IChatDriver` benutzen.

In dieser Aufgabe verwenden Sie das gegebene Java GUI und setzen aus der Klientenklasse heraus HTTP Anfragen ab, die dann vom Servlet bearbeitet werden.

Benützen Sie als Web-Container Tomcat. Tomcat kann als *stand-alone* servlet container verwendet werden. Tomcat (aktuell 2015: 8.0.21) finden Sie unter http://tomcat.apache.org/ oder in Ihrer Linux Installation.
Geben Sie neben dem Code der Servlet-Implementierung auch das Konfigurationsfile `web.xml` ab.

Bemerkungen zur Implementation

Der Server, abgeleitet von der Klasse `HTTPServlet`, erstellt eine Instanz der Klasse `ChatRoomDriver` und ruft ihre Methode `connect()` auf, die ihrerseits eine Instanz der `ChatRoom`-Klasse erzeugt. Es gibt nur eine Instanz von `ChatRoom`, die alle Klienten, die möglichen Gesprächsthemen, und die eingehenden Nachrichten verwaltet. `ChatRoom` delegiert alle Anfragen an je eine seiner inneren Klassen, `Participants` oder `Chats`. Die Klasse `Participants` verwaltet die Namen der aktiven Mitglieder und die Klasse Chats die Themen und Nachrichten. Dieser Aufbau erlaubt eine komplette Trennung der beiden, voneinander unabhängigen Funktionalitäten, was den Zugriff auf die gespeicherten Daten effizienter macht. Danach bearbeitet der Server die Meldungen seiner Klienten auf eine ähnliche Weise wie in der Socket-Applikation. Bemerken Sie aber, dass die Kommunikation mittels des HTTP Protokolls via Anfrage/Antwort verläuft. Um genau die gleiche Funktionalität wie im Fall der Socket-Applikation anzubieten, bedarf es eines speziellen Designs. Sie können diese Applikation aber ganz einfach spezifizieren: Der Server verschickt nicht automatisch an alle Beteiligten jede Mitteilung, sobald er diese empfängt, sondern er speichert sie. Der Klient muss die Mitteilungen beim Server selber verlangen.
Sie können sich dazu der folgenden zwei Knöpfe im ClientGUI bedienen:

- <Get Msgs> Wird dieser Knopf gedrückt, werden die letzten zehn gespeicherten Mitteilungen zu dem Thema, das gerade im Themencombo ausgewählt ist, beim Server verlangt und dann in der Textanzeige dargestellt.
- <Refresh> Wird dieser Knopf gedrückt, werden nicht nur die letzten zehn Mitteilungen verlangt und angezeigt, sondern es werden auch die beiden Themen- und Teilnehmer-Combos aufgefrischt.

Es ist Ihnen aber frei gestellt, auf welche Weise Sie dieses Problem lösen möchten.

Wie im Fall der Socket-Applikation, kann der Client mittels der Klasse `RunClient` im Paket `ch.fhnw.kvan.chat` gestartet werden. Die Datei `Clients.txt` enthält eine Vorgabe mit den nötigen Argumenten.
Die `main()`-Methode des Client benötigt folgende Argumente:
<Name > <Pfad> wobei Pfad hier den spezifischen Teil der URL-Adresse angibt, der den Server definiert, wie z.B.: `localhost:8080/chat/Server`.
Der Pfad wird dann von der Methode `connect(…)` als Argument gebraucht, damit sich der Klient beim Server anmelden kann.

Bei allen Anfragen und Mitteilungen wird der Pfad mit der Protokoll-Bezeichnung (http://) und dem eigentlichen Anfrageteil (siehe unten) ergänzt; die so entstandene HTTP Abfrage verschickt; die Antwort des Server bearbeitet und die erhaltenen Daten angezeigt. Die Kommunikation ist hier so einfach wie möglich zu halten. Darum wird die Standard-Abfragemethode `GET` benutzt. (Beachten Sie aber, dass nur kurze Nachrichten mit Hilfe dieser Methode verschickt werden können.)

Die Kommunikation

Klient-Anfragen:

"?action=addParticipant&name=<Name >"	der Klient meldet sich an
"?action=removeParticipant&name=<Name >"	der Klient meldet sich ab
"?action=addTopic&topic=<Thema>"	Thema einfügen
"?action=removeTopic&topic=<Thema>"	Thema entfernen
"?action=postMessage&message=< Nachricht>&topic=<Thema>"	eine Nachricht zum gegebenen

	Thema wird gesendet; diese muss vor dem Verschicken mit dem Namen des Klienten ergänzt werden
"?action=getMessages&topic=<Thema>""	die zehn letzten Nachrichten zum gegebenen Thema werden verlangt
"?action=refresh&topic=<Thema>"	der Klient verlangt die zehn letzten Nachrichten zum gegebenen Thema und die aktuelle Liste der Themen und der Teilnehmer

Server-Antworten:

Wie im Fall der Socket-Applikation, sendet der Server Daten an den Client in Form von Strings. Wenn mehrere Meldungen auf einmal vom Server an den Client geschickt werden, wird jede Meldung als eine separate Zeile gesendet. Am Anfang jeder Nachrichtenzeile steht ein Schlüsselwort, der die Art der Meldung und deren Inhalt bezeichnet:

"topics=<Thema1;Thema2;Thema3;….;> "	der Server schickt die Liste aller aktueller Themen
"participants=<Name1;Name2;Name3;….;> "	der Server schickt die Liste aller aktiven Teilnehmer
"messages= Nachricht 1;; Nachricht 2;; Nachricht 3;;…;;"	der Server schickt die Liste der letzten zehn Nachrichten zum gewählten Thema

Freilich können Sie die ganze Kommunikation anders gestalten. Beachten Sie, dass Sie je nach Ihrer Implementationsart unter Umständen noch andere Anfragen/Antworten definieren müssen.

Aufgabe

Sie sollen die beiden Klassen `Server.java` und `Client.java` implementieren. Beachten Sie, dass der Chatroom als Singleton zu definieren ist.
Alle anderen Klassen haben Sie schon im früheren `Chat.zip`-File auf der Webseite erhalten.

Vergessen Sie nicht, dass der Server einem neuen Chat-Teilnehmer bei seiner Anmeldung sowohl die aktuelle Liste aller aktiven Mitglieder, als auch die Themenliste zuschicken sollte. Beachten Sie dabei, dass die Klasse `Participants` den String `participantString` enthält, welcher die aktuelle Liste in der folgenden Form enthält: "participants=<Name1;Name2;Name3;….;>". Dasselbe gilt für die Themenliste, die in der Klasse `Chats` im topicsString ("topics=<Thema1;Thema2;Thema3;….;>") gespeichert wird.

Lieferung

Geben Sie neben dem Code eine Beschreibung Ihrer Lösung ab. Daraus soll hervorgehen, wie Sie die Kommunikation Client-Server realisiert haben. Bitte beachten Sie, dass alle von Ihnen implementierten Files in einem einzigen File (zusammen mit der kurzen Beschreibung des Lösungswegs) in **pdf-Format** an carlo.nicola@fhnw.ch senden sollen.

Abgabe: 20. Mai 2015