Übungsaufgabe #6: Library - JINI

18.1.2004

In dieser Aufgabe soll die Client/Server-Anwendung aus Aufgabe 4 mit Hilfe von JINI verbessert werden. Als Ausgangspunkt kann die Bibliotheksverwaltung aus Aufgabe 1 oder Aufgabe 4 dienen.

a) Wir beginnen mit der `LibraryDB`. Diese soll wie in Aufgabe 4 die Datenbank darstellen und von Clients über Veränderungen informiert werden. Die Clients sollten demzufolge auf ein Objekt dieses Types zugreifen können, d.h. es soll ein Remote-Objekt sein und, wie bisher auch, das Interface `LibraryDB` implementieren. Um Bibliotheksdienste anderer Gruppen nutzen zu können müssen die Schnittstellen exakt gleich sein. Im Verzeichnis */proj/i4mw/pub/aufgabe6* ist daher das Interface `LibraryDB` nochmals vorgegeben. Schreiben Sie nun eine Klasse `library.server.Server`, welche ein Objekt vom Typ `LibraryDBImpl` erzeugt und es bei allen Lookupdiensten registriert. Die Lookupdienste sollen zuvor mittels Multicastdiscovery gesucht werden; als Proxy kann ein RMI-Stub dienen.
   Ändern Sie anschließend die Klasse `library.client.LibraryFrontend`, so dass diese eine Referenz auf die Datenbank im Konstruktor übergeben bekommt. Anschließend erstellen sie die Klasse `library.client.Client`, welche einen Lookupdienst per Multicast sucht und dort anschließend einen Proxy für eine `LibraryDB` herunterlädt.
   Zum Testen können Sie den Lookupdienst *reggie* auf einem Rechner im CIP-Pool starten.

b) Um den eigenen `LibraryDB` Service zu identifizieren soll nun ein `IDEntry` erstellt werden, welcher mindestens einen Namensstring enthält. Sowohl dem Server als auch dem Client soll ein Name auf Kommandozeile übergeben werden. Beim Registrieren soll der Dienst mit Hilfe eines `IDEntry` Objekts markiert werden. Der Client soll nun den eigenen Service suchen. Außerdem soll die Methode `getName` in der Klasse `LibraryDBImpl` diesen Namen zurückgeben.

c) In diesem Schritt soll der Client um den Befehl "list" (wie in Aufgabe 2) erweitert werden. Dieser soll bei jedem gefundenen LookupService nach allen `LibraryDB` Diensten suchen und die dort vorhandenen Bücher ausgeben. Die Methode `getName` der `LibraryDB` Objekte kann dazu genutzt werden die Ausgabe zu strukturieren.

d) Nun sollen Clients nach Bibliotheken suchen können, welche ein bestimmtes Buch anbieten. Erstellen Sie dazu das Objekt `ItemEntry`, welches mindestens den Titel eines Items enthalten soll. Der Server muss nun dafür sorgen, dass die `Entry` Objekte, welche bei den Lookupservicen hinterlegt sind, im Falle eines "register"-Aufrufes aktualisiert werden.
   Ein Client soll nun bei einem "borrow" Befehl eine Bibliothek suchen, die das gewünschte Item enthält. Der Befehle "register" soll weiterhin auf den beim Starten spezifizierten Bibliotheksdienst angewendet werden. Bei "return" soll ebenfalls eine Bibliothek gesucht werden, bei der das Item zurückgegeben werden kann.

e) Nun sollen die Clients wie in Aufgabe 4 eine kurze Meldung ausgeben, wenn ein neues `Item` in einer Datenbank eingetragen wird. Ein Client soll sich dazu bei jedem gefundenen Bibliotheksdienst registrieren. Ein Dienst soll alle registrierten Clients über Zustandsänderungen (bei einem `register`-Aufruf) informieren. Verwenden sie bei der Implementierung die `RemoteEvent` Schnittstellen von JINI.

Bearbeitung: **bis zum 29.1.2004/18:00 Uhr**
Alle Dateien sollen im Verzeichnis `/proj/i4mw/`*`loginname`*`/aufgabe6/` abgelegt und mit dem `abgabe`-Programm abgegeben werden.
Die Bearbeitung ist in 2er Gruppen möglich.

Übungen zu MW