Übungsaufgabe #5: Library - JINI

24.11.2005

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 `SimpleDB`. 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 `SimpleDB` implementieren. Um Bibliotheksdienste anderer Gruppen nutzen zu können müssen die Schnittstellen exakt gleich sein. Im Verzeichnis */proj/i4mw/pub/aufgabe5* ist daher das Interface `SimpleDB` nochmals vorgegeben. Schreiben Sie nun eine Klasse `<loginname>.mwlibrary.server.Server`, welche ein Objekt vom Typ `SimpleDBImpl` 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 `<loginname>.mwlibrary.client.LibraryFrontend`, so dass diese eine Referenz auf die Datenbank im Konstruktor übergeben bekommt. Anschließend erstellen sie die Klasse `<loginname>.mwlibrary.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 `SimpleDB` Service zu identifizieren soll nun ein `IDEntry` verwendet werden, welches einen Namensstring enthält. Die Klasse `IDEntry` ist vorgegeben, um die Interoperabilität zwischen den entstehenden Lösungen zu erhöhen. 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 `SimpleDB` und `SimpleDBImpl` um die Methode `getName` erweitert werden, welche diesen Namen zurückgibt.

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

d) Nun sollen Clients nach Bibliotheken suchen können, welche ein bestimmtes Buch anbieten. Verwenden sie dazu das vorgegebene `ItemEntry`, welches den Titel eines Items enthält. 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 15.12.2005/20:00 Uhr
(abgeben nicht vergessen, 2er Gruppen erlaubt)

Übungen zu MW