Praktikum 5 zur OOS SS16 G1: 17.06. & G2: 10.06.

Aufgabe 1: (Multiuserfähigkeit)

Ändern Sie die Implementierung Ihres Programms aus Praktikum Versuch 4 Aufgabe 2 so ab, dass es multi-user-fähig wird. Überlegen Sie sich hierzu, wie Sie das Konzept der Threads einsetzen, so dass der Server auch mehrere Clients nebenläufig bedienen kann.

Aufgabe 2: (Webservice)

Ziel dieser Aufgabe ist es, den bisherigen Server aus Versuch 4 als *WebService* zu implementieren. Ein *WebService* nutzt hierfür in der Regel standardisierte Protokolle, um mit Clients zu kommunizieren. Das bietet vor allem 3 Vorteile:

- Man muss sich nicht um die Einzelheiten des Nachrichtenversandes kümmern.
- Ein *WebService* kann von allen Programmiersprachen verwendet werden, die Webservices unterstützen.
- Webservices bieten in der Regel ihre Dienste über HTTP an.

In diesem Versuch werden wir die Kommunikation mit sog. *SOAP-WebServices* realisieren. Informieren Sie sich hierfür zusätzlich über die Funktionsweise von *SOAP*-basierten *WebServices*.

Vorbereitungen:

Zur Lösung der Teile 1 und 3 sollten Sie statt eines einfachen Projekts ein „Dynamic Web Project" erstellen.

WICHTIG: Zur Bearbeitung der Teile benötigen Sie die **Webservice-Anleitung** und die vorbereiteten Klassen, die Sie auf der Homepage finden. Dort erfahren Sie ebenfalls, wie Sie ihren eigenen Rechner für Webservices mit Eclipse einrichten können.

Die vorbereiteten Klassen haben die gleiche Funktionalität wie Ihre Klassen aus Versuch 4. Sie sind allerdings auf das Notwendige beschränkt. Jedoch sind die Einschränkungen bei der Verwendung von *WebService*, die in den folgenden Hinweisen aufgeführt sind, darin bereits berücksichtigt. Deshalb wird empfohlen, die vorbereiteten Klassen zu verwenden.

Allgemeine Hinweise:

Bei der Erstellung von *WebServices* müssen folgende Beschränkungen beachtet werden:

- Keine Verwendung von Umlauten.
- Manche Datentypen können nicht verwendet werden (Bsp: char-Array).

Praktikum 5 zur OOS SS16 G1: 17.06. & G2: 10.06.

- Methoden müssen mit einem Kleinbuchstaben anfangen; Klassennamen mit einem Großbuchstaben.
- Alle verfügbaren *WebService*-Methoden müssen als *public* deklariert worden sein.

Teil 1: Service erstellen

Erstellen Sie ein neues leeres "Dynamic Web Project" und importieren Sie die von uns bereitgestellte Server-Implementierung. Importieren Sie die Dateien von „\praktikum5\Server\" in ihr Projekt unter "\src".

Welche Klasse implementiert den Server?
Erzeugen Sie nun (mit Hilfe der Anleitung) einen Webservice aus dieser Klasse.

Teil 2: Client erstellen

Legen Sie ein neues leeres "Java Projekt" an. Generieren Sie nun mit Hilfe der .wsdl Datei die Client Proxy-Klassen.

Welche Klassen werden außer den Proxy-Klassen noch erzeugt?
Kommen Ihnen diese Klassen bekannt vor?
Schauen Sie sich die Implementierungen dieser Klassen an. Was fällt Ihnen auf?

Analog zum Server importieren Sie alles aus "\praktikum5\Client\" in das Clientprojekt unter "\src".

Teil 3: Webservice Aufruf

Wie ihnen vielleicht schon aufgefallen ist, existiert ein Fehler in der Client.java. Der Client muss nun so abgeändert werden, dass er den Webservice benutzt. Der im Client definierte ClientOrb ist im Moment falsch deklariert. Welche Klasse kann den ClientOrb vollkommen ersetzen?

Machen Sie das Programm lauffähig! Überprüfen Sie in der Service-Console, ob die Aktionen vom Client auch wirklich ausgeführt werden.