Thorsten Berger, Stefan Sosnicki, Michael Welt

1. Allgemeines

Unsere Gruppe hatte bei dieser Aufgabe die Zielstellung, einen lauffähigen Prototypen zu entwickeln, der als eine Art Konzept- und Designstudie mit minimaler Funktionalität ausgestattet ist. Dabei ging es konkret um die Implementierung eines Servlets mit Benutzerverwaltung, Sitzungsverfolgung (Session Tracking) und die Anzeige benutzerspezifischer Informationen mit dynamisch generierten HTML-Seiten. Die Benutzerdatenbank stellt dabei eine XML-Datei dar, welche mit DOM und SAX verarbeitet werden sollte.

Folgenden Ablauf einer Sitzung soll unser Prototyp gewährleisten:

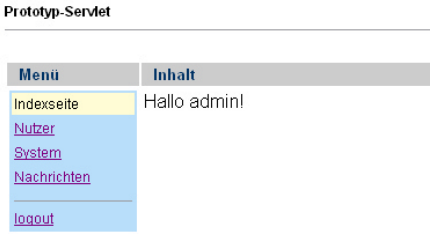
- der Benutzer loggt sich auf dem Prototypen ein
- der Benutzer kann über ein Menü zwischen verschiedenen (rollenbezogenen) Informationen navigieren
- der Benutzer hat die Möglichkeit, über eine minimale Chat-Funktion mit anderen eingeloggten Benutzern zu kommunizieren
- Ende der Sitzung durch Ausloggen des Benutzers

Im Gegensatz zu Anwendungsprogrammen mit eigener GUI basieren Servlets auf dem verbindungslosen HTTP-Protokoll, d.h. das Servlet kann nicht ohne weiteres einer Client-Anfrage einen eingeloggten Benutzer zuordnen. Dieses Problem beseitigen Sessions, indem jedem Client eine eindeutige Session ID zugeordnet wird und diese bei jeder Anfrage durch den Browser mitgeschickt wird. Die Session Tracking API liefert hier verschiedene komfortable Methoden zur Sitzungsverfolgung bei Java-Servlets.

2. Produktübersicht

Dem Benutzer präsentiert sich zu Beginn einer Sitzung die Login-Aufforderung:

Nach erfolgreicher Authentifizierung gegen die Benutzerdatenbank gelangt man in den internen Bereich:



Hier kann der Benutzer über ein dynamisch generiertes Menü zwischen folgenden Seiten navigieren:

- **Indexseite:** Begrüßung des Benutzers ;)
- **Nutzer:** Anzeige der Rollen des Benutzers (welche auch in demo-users.xml eingetragen sind)
- **System:** Anzeige verschiedener Systeminformationen, insofern der Benutzer durch seine Rolle dazu berechtigt ist
- **Nachrichten:** Anzeige eingegangener Nachrichten anderer Benutzer sowie eine Möglichkeit, selbst Nachrichten zu verschicken
- **Logout:** Ausloggen des Benutzers, d.h. Beenden seiner Sitzung

Außerdem existieren eine Kopf- und Fußzeile, die bei jeder Seite eingebunden werden und weitere Informationen enthalten können. Diese sind als separate HTML-Dateien ausgelegt (*header.html* bzw. *footer.html*) und können unabhängig vom Servlet bearbeitet werden.

3. Grundsätzliche Design-Entscheidungen

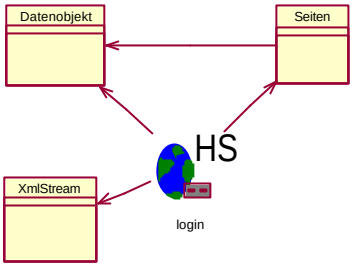
Aufgrund des relativ geringen Umfangs des Prototyps haben wir uns entschieden, nur ein Servlet zu erstellen, das die Grundfunktionalität (Authentifizierung, Sitzungsverwaltung) enthält. Um dieses Servlet herum sind im wesentlichen Klassen zur Haltung und Verwaltung von Daten implementiert. Zum einen zur dynamischen Haltung der Daten des online-Betriebes, und zum anderen zur persistenten Haltung aller rechterelevanten - und Authentifizierungs-Daten.
Weiterhin bleibt anzumerken, dass sich ein Benutzer nur einmal einloggen darf, da ansonsten die Methoden zum Nachrichtenaustausch evtl. keine eindeutige Ziel-Adresse hätten bzw. dann ein weiteres Unterscheidungskriterium hätte eingeführt werden müssen. Sollte sich ein Benutzer trotzdem zweimal einloggen wollen, so wird die ältere Session einfach gelöscht.

Im einzelnen :

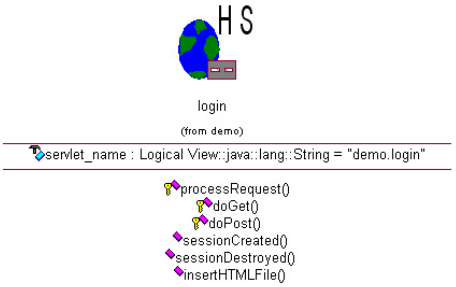
`login.class` `Autor: Thorsten Berger`	Das eigentliche Servlet.
`XmlStream.class` `Autor: Michael Welt`	Implementierung eines Interface zum Einlesen und Auswerten der XML-Datei *demo-users.xml* die persistente Daten zur Benutzerauthentifizierung enthält.
`Datenobjekt.class` `Autor: Stefan Sosnicki`	Enthält die Benutzerverwaltung im online Betrieb, ist somit dynamisch, ergo eine Liste zur Speicherung von Informationen über eingeloggte Benutzer sowie Methoden zum Nachrichtenaustausch.
`Seiten.class` `Autor: Stefan Sosnicki`	Verschiedene Methoden, die das HTML-Frontend generieren.

Es gab bei der Planung des Prototyps auch Überlegungen, die Aufgaben über mehrere Servlets zu verteilen, also für jede Seite ein Servlet zu schreiben. Dies hätte allerdings die Implementierung weiterführender Konzepte, insbesondere der Inter-Servlet-Kommunikation erfordert und wäre insgesamt fehleranfälliger.
Stattdessen entscheidet das Servlet mit Hilfe eines an die URL angehängten Parameters, welche Seite angezeigt werden soll und ruft die entsprechende Methode aus der Klasse *Seiten* auf.

4. Paket- und Klassenstruktur

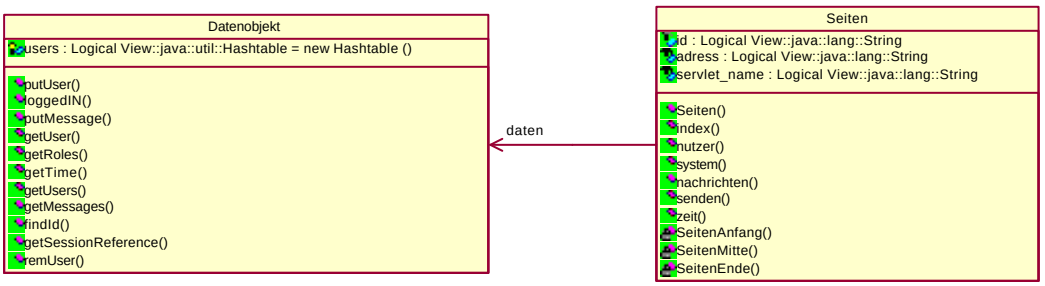


login.class:



Die Funktionsweise bei Aufruf des Servlets ist folgende:
Eine neue Session wird angelegt bzw. eine bereits bestehende weitergeführt.
Als nächstes erfolgt eine Überprüfung, ob ein Benutzer mit dieser ID bereits im *Datenobjekt* registriert ist. Ist dies der Fall, so werden die Methoden der Klasse *Seiten* ausgeführt, welche je nach Wert des URL-Parameters *?page* die entsprechende HTML-Seite liefert. Anderenfalls erscheint ein Login-Formular (wenn keine Login-Informationen im Request gepostet wurden) bzw. der Benutzer wird mit Hilfe der Klasse *XmlStream* authentifiziert und mittels *Datenobjekt.putUser()* eingeloggt.
Zum Ausloggen ist der URL-Parameter *?page=logout* vorgesehen, der ein Löschen der Session veranlasst.
Weiterhin implementiert das Servlet die *HttpSessionListener* Schnittstelle, um bei Verfall einer Session (entweder durch Ausloggen oder Überschreitung eines Timeouts) den entsprechenden Benutzer aus dem Datenobjekt zu entfernen.

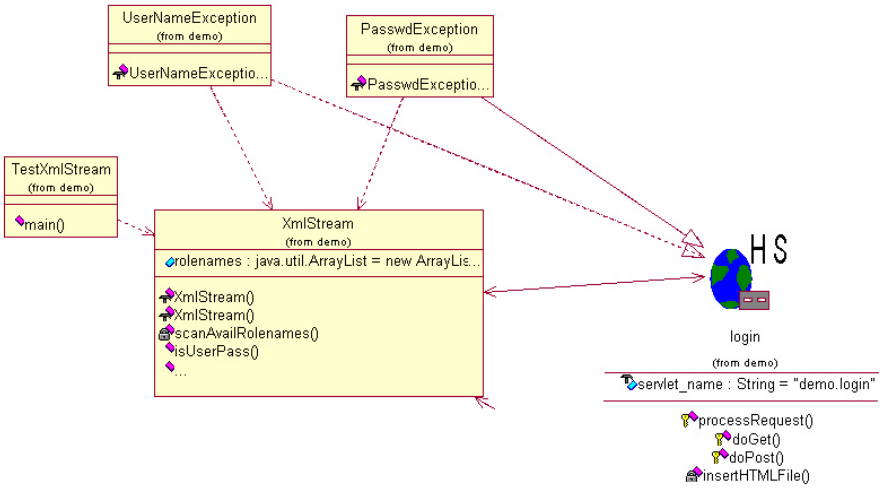
Datenobjekt.class, Seiten.class:



Die Klasse *Datenobjekt* speichert die Daten sämtlicher angemeldeter Nutzer und stellt Methoden zum Auslesen und Manipulieren dieser Daten zur Verfügung. Die Klasse *Seiten* verwendet eben diese Methoden um eine Art Schnittstelle zwischen Datenobjekt und Servlet zu bilden.

Die Speicherung der Nutzerdaten erfolgt in einer *Hashtable*. Den Primärschlüssel bildet dabei die SessionID da diese eindeutig einem Benutzer zugeordnet werden kann. Der Wert setzt sich aus einem *Vector* zusammen, bestehend aus dem Benutzernamen, seinen Rollen, dem Zeitpunkt seiner Anmeldung, den ihm geschickten Nachrichten und einer Referenz auf das Sessionobjekt des Benutzers. Es werden Methoden bereitgestellt zum Eintragen und Löschen von Benutzern, zum Auslesen der einzelnen Daten und zum Eintragen eines Nachrichtenstrings bzw. zur Ausgabe aller Nachrichten eines Benutzers. Ferner gibt es noch Methoden um zu ermitteln ob ein bestimmter Benutzer zur Zeit angemeldet ist oder nicht.
Die Klasse Seiten erzeugt im Wesentlichen die Ausgabe des Servlets unter Zuhilfenahme der vom *Datenobjekt* bereitgestellten Funktionalitäten. Die Klasse ist unterteilt in verschiedene Methoden, die jeweils für einen bestimmten Bereich des Servlets verantwortlich sind. Das sind die Indexseite, die Seite mit den nutzerrelevanten Daten, die (rollenabhängigen) systembezogenen Daten, und die Nachrichtenverwaltung. Die Formatierung und Anzeige des Menüs erfolgt wiederum über gesonderte Methoden.

XmlStream.class:



Wie oben beschrieben implementiert diese Klasse eine Schnittstelle zu dem XML-File „demo-users.xml“. Es stellte sich die Aufgabe, die darin gegebenen Informationen zu extrahieren, und in entsprechender Form aufzubereiten. Dies ist mit Hilfe der SAX – Umgebung und der dynamischen Datenstruktur DOM umgesetzt worden.
Im Einzelnen stellt die Klasse Funktionen bereit, um entsprechend eine solche Datei einzulesen, und intern in einem DOM zu verwalten, und die üblichen Query Funktionen, wie z.B. eine Anfrage nach Richtigkeit von Benutzernamen – Passwort Kombinationen, sowie eine Abfrage der zu einem Benutzer zugehörigen Rollen.

Angewandte OO-Konzepte sind:
- Klassenkonzept von Java
- Dynamische Datenerfassung in Form von z.T. generischen ArrayListen
- Datenkapselung in der Form:
  - Zustand des Objektes kann nur in der spezifizierten und angegebenen Form abgefragt werden Anwendung von sogn. „gettern“
  - Interface-Modellierung, der Art, dass die Methoden zur Abfrage fest vorgegeben sind, und unabhängig (nach außen hin) von meiner internen Implementierung
- Erweiterbarkeit in der Form, dass ich neu aufgenommene Benutzer so wie auch Benutzerklassen ohne weiteres abfragen kann, ohne den Quelltext überarbeiten zu müssen.