Universität Potsdam - HPI
Hasso Plattner Institut
Internet-Technologien und Systeme
Lehrveranstaltung *Grundlagen der Webprogrammierung*
Prof. Dr. sc. nat. Christoph Meinel

Kristian Eckstein
Dennis Gerike
Benjamin Lüpfert
Robert Schmidl

Chatfunktionalität für die Plattform ”tele-task”

Zwischenpräsentation

Potsdam, den 26.12.2005

Inhaltsverzeichnis

1 Einführung und Aufgabenstellung **2**

1.1 Aufgabenstellung 2

1.1.1 denkbare Erweiterungen 2

1.1.2 Anforderungen im Hinblick auf den Rest des Projekts 3

1.2 Analyse 3

1.2.1 Der Chat 3

1.2.2 Verwaltung 4

2 Umsetzung und Status Quo **5**

2.1 Grundgedanke 5

2.1.1 Backend 5

2.1.2 Frontend 6

2.2 Kommunikation 7

3 Ausblick **9**

3.1 Geplante Änderungen nach der Präsentation 9

3.2 Geplante Änderungen 9

3.3 Nächste Schritte 10

Kapitel 1

Einführung und Aufgabenstellung

In diesem Teil möchten wir zunächst die gestellte Aufgabe analysieren und unsere Herangehensweise sowie unsere Gedanken zur Lösung dieser Aufgabe darlegen.

Dazu stellen wir die getroffenen Entscheidungen hinsichtlich der geforderten Funktionalität dar und beleuchten diese ausführlich.

1.1 Aufgabenstellung

Es gilt ein System zu entwickeln, dass es ermöglicht auf der eLearning Plattform ”tele-task” neben den Video- bzw. Textstreams einen Chat anzubieten. Dieser Chat soll allen Benutzern zur Verfügung stehen, die sich vor Betrachten des Streams ausdrücklich für die Verwendung eines Chats entschieden haben.

Sollte zu der gewählten Veranstaltung bereits ein Stream mit Chat laufen, so soll dem Nutzer dies angezeigt werden und er soll vor die Wahl gestellt werden, dem laufenden Chat (bzw. einem der laufenden Chats) beizutreten oder einen neuen Chat zu eröffnen. Die Wahl, einem bestehenden Chat beizutreten, geht dann mit einer Synchronisierung des Streams einher. Diese findet anhand des aktuellen Zeitindex des Chats statt, dem der Nutzer beitreten möchte. Über diese Synchronisierung, die es dem Nutzer in diesem Fall unmöglich macht, den Stream von Anfang an zu sehen, wird der Nutzer umfassend informiert und es wird ihm der aktuelle Zeitindex der vorhandenen Chats mitgeteilt.

Sollte er dies nicht wünschen, startet er einen neuen Chat und sein Stream beginnt von vorne.

1.1.1 denkbare Erweiterungen

Über diese Anforderungen hinaus, ist es denkbar, dass mit dieser Funktionalität eLearning-Seminare abgehalten werden. Für diesen Fall könnte es ein Planungsmodul geben, das sowohl Benutzer einer bestimmten Zielgruppe (Anmeldung vorausgesetzt) einlädt, sowie den Stream solange anhält, bis

er freigegeben wurde. So könnte sichergestellt werden, dass alle Teilnehmer dieser Live-Sitzung den gleichen Stream von Anfang an sehen.

Ebenfalls denkbar wäre eine Speicherung des Chatprotokolls in bestimmten Chats, die von vornherein als solche ausgewiesen wurden. Möglich wäre dies u.a. in betreuten Tutorien, in denen ein Livetutor im Chat anwesend ist bzw. war. Um dieses Tutorium anschließend allen zur Verfügung zu stellen, könnte man das Chatprotokoll in entsprechende Formate exportieren (PDF).

1.1.2 Anforderungen im Hinblick auf den Rest des Projekts

Nach Analyse der Aufgabenstellung sind wir zu dem Schluss gekommen, dass wir unsere Ziele weitestgehend autark von den restlichen Gruppen des Projekts umsetzen können. Die einzigen Gruppen, mit denen eine genaue Koordinierung unabdingbar ist, sind die Gruppe, die das Benutzerobjekt erstellt und die Gruppe, die für die Implementierung des Streamplayers verantwortlich ist.

Zu diesen Koordinierungsanforderungen schreiben wir mehr in Kapitel 3.

1.2 Analyse

Die Anforderungen der Aufgabe lassen sich auf zwei Teilbereiche herunterbrechen: Zum Einen muss ein Chatsystem entwickelt werden, das übersichtlich unter dem gewählten Stream dargestellt werden kann und zum Anderen muss eine Verwaltung der geöffneten Chats und Streams implementiert werden, die den Nutzer vor die in Abschnitt 1.1 beschriebene Wahl stellen kann.

1.2.1 Der Chat

Da der Chat unter dem gewählten Stream angezeigt werden soll und die Verwendung des HTML Frames nicht in Frage kommt, bietet sich hier die Nutzung der AJAX-Technologie[1] an. Diese ermöglicht es, den Inhalt eines bestimmten Bereichs einer Webseite zu manipulieren, ohne die gesamte Seite neu aufzubauen.

Dies ist insbesondere deshalb förderlich, da das Hauptaugenmerk des Nutzers ja weiterhin auf dem Stream liegen soll und nicht auf dem Chat, der hier als Zusatzfeature (vgl. ”convenience-feature”) zum Einsatz kommt. Daher wäre es unmöglich, die gesamte Webseite neu aufzubauen. Durch die Nutzung von AJAX auf der Seite des Frontends wird dies vermieden.

Die Frage des Backend auf Serverseite klärte sich mehr oder weniger von selber, da auf dem Webserver, der dem Team zur Verfügung steht, nur PHP und MySQL installiert sind. Eine genauere Betrachtung der Realisierung des Chats ist in Kapitel 2 zu finden.

[1] http://de.wikipedia.org/wiki/Ajax_(Programmierung)

1.2.2 Verwaltung

Der Überbau des eigentlichen Chats, die ”Verwaltung” ist in unseren Planungen stets der letzte Teil gewesen, den es umzusetzen gilt, da wir zunächst den Chat an sich realisieren wollten, damit es etwas gibt, was verwaltet werden kann. Die Gedanken, die wir uns um diese Verwaltung gemacht haben, sollen hier trotzdem mit einfließen.

Wie schon in Abschnitt 1.1 beschrieben, muss der Nutzer eine Wahlmöglichkeit bekommen, ob er einem bestehenden Chat zu seinem gewählten Stream beitreten möchte, oder ob er einen neuen Chat eröffnen möchte.

Hier ist ein Einsatz der AJAX Technologie nicht unbedingt erforderlich, da hier keine Live Manipulation der Inhalte einer Webseite nötig ist.

Mit Hilfe des Verwaltungsmoduls soll es ebenfalls möglich sein, die eingangs angesprochenen orgnisierten Chats zu planen und zu administrieren. Mit diesen sind dann die erwähnten Live-Tutorien denkbar.

Kapitel 2

Umsetzung und Status Quo

Nun möchten wir einen kurzen Überblick über die Umsetzung der genannten Ziele geben und inwieweit uns diese bereits gelungen ist. Es sei angemerkt, dass wir hier nur die Umsetzung des Chats darlegen können, da wir, wie beschrieben, die Verwaltung bisher noch nicht angegangen sind.

2.1 Grundgedanke

Der Chat an sich ist ein Austausch von Strings zwischen den Chatteilnehmern, die ihrerseits an ihrem Benutznamen zu erkennen sind (auch wieder ein String) und deren Nachricht zu einer bestimmten Zeit geschickt wurde. Dies sind die drei Hauptdaten, die im Verlauf eines Chats periodisch ausgetauscht werden.

Dazu müssen zwei Komponenten realisiert werden: Das Backend auf dem Server, das die Verteilung der Nachrichten in die einzelnen Chats und die Zuordnung der Nutzer übernimmt. Und das Frontend, was einen aufgeräumten Chat darstellt, den der Nutzer effizient gebrauchen kann.

Anzumerken sei hier noch, dass in diesem Kapitel von uns mehrmals von einem ”Timestamp” gesprochen wird. Gemeint ist hierbei jeweils der aktuelle Unix-Timestamp, an dem die Nachricht beim Server eingetroffen ist.

2.1.1 Backend

Das Backend haben wir mit PHP realisiert. Da die erforderlichen Daten sehr gut strukuriert werden können, haben wir uns zunächst dafür entschieden serverseitig eine XML Datei zum Speichern der Daten zu benutzen, was man in Abbildung 2.1 sehr schön sehen kann.

Ebenso sollte die Liste der aktuell angemeldeten User als XML Datei auf dem Server vorgehalten werden. Diese wird aktuell zum Aufbau der Liste der aktiven Chats genutzt und ist in Abbildung 2.2 zu sehen.

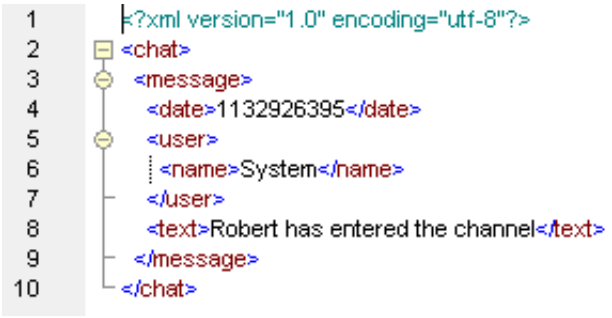

Abbildung 2.1: Chatlog als XML Datei

Abbildung 2.2: Userliste als XML Datei

Wie ebenfalls in den Abbildungen 2.1 und 2.2 zu sehen, werden sowohl der Timestamp der gesendeten Nachrichten, als auch der Raumname in den XML Dateien abgebildet.

Dies ist aus zweierlei Gründen notwendig bzw. vorteilhaft. Einerseits ist es so möglich nur die aktuellsten Nachrichten des Chats zu übertragen, was verhindert das jeder User den gesamten Chat anfordert, andererseits ist es so möglich komfortable Logs zu erstellen, die mit einer genauen Uhrzeit und einem Raumnamen gekennzeichnet werden können.

Unser Chat besteht bisher aus einer PHP Klasse, wie in Abbildung 2.3 zu sehen. Diese nutzt Funktionen der PEAR Klasse XML_Serializer, welche zum konvertieren der Daten in und aus XML Formaten zuständig ist.

2.1.2 Frontend

Das Frontend besteht aus einem Eingabefeld und dem eigentlichen Chatfenster, gepaart mit der Anzeige der aktuellen Chats. Dieser Chatbereich wird auf der Stream Seite unter dem Streamplayer dargestellt, so dass das Hauptaugenmerk, wie gefordert, weiterhin auf dem eigentlichen Inhalt, dem Stream, liegt.

Das Frontend wird mittels JavaScript und AJAX realisiert, wobei hier nur Strings an den Server

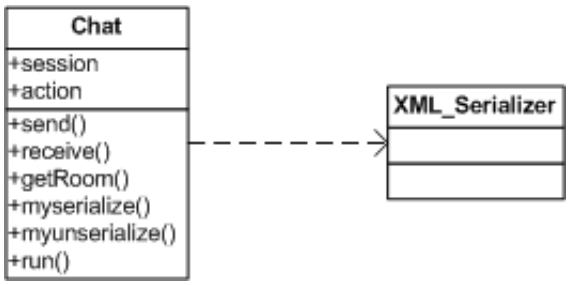

Abbildung 2.3: UML Diagramm des Chat-Backends

gesendet werden, während der Server XML zurückgibt. Diese XML Fragemente werden auf dem Client ausgewertet und der jeweils aktuelle Chat wird aufgebaut.

Dabei wird der Server periodisch nach neuen Inhalten abgefragt. Die Abstände hierzu liegen momentan bei zwei Sekunden für den eigentlichen Chat und zehn Sekunden für die Chatliste.

Das Zusammenspiel von Backend und Frontend ist in Abbildung 2.4 nocheinmal zusammengefasst.

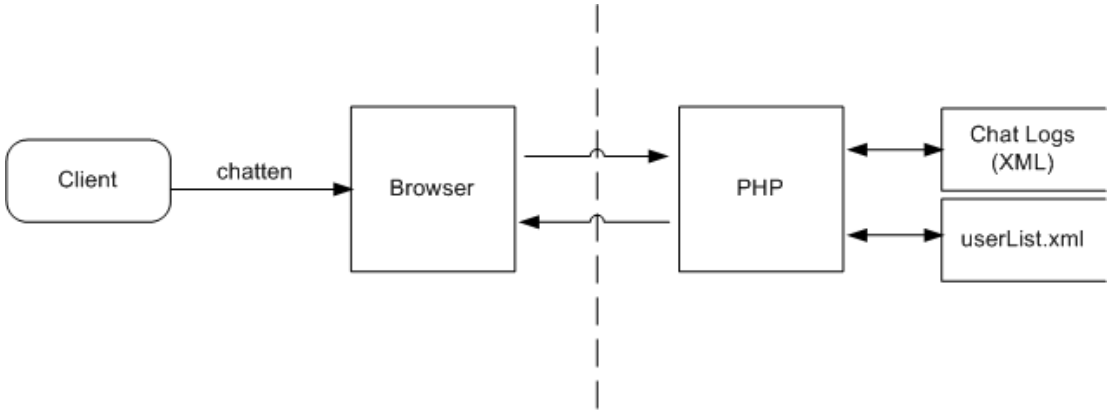

Abbildung 2.4: kompletter Datenfluss

Da auf dem Server pure XML Dateien gespeichert werden, können hier mittels ”Formatting Objects” auf bequemen Wege PDF Exporte der einzelnen Chats erstellt werden.

2.2 Kommunikation

Wie schon in 2.1.1 und 2.1.2 erwähnt, sendet der Server komplette XML Fragemente, während er von den Clients nur Strings bekommt, aus denen er die beschriebenen XML Dateien zusammenfügt (vgl. Abbildung 2.5)

Diese Art der Übertragung wurde von uns gewählt, da vom Nutzer auch nur einzelne Strings eingegeben werden, während die Darstellung nach recht komplexen Datenstrukturen verlangt. Wir haben also die Anforderungen, die man aus der Aufgabenstellung modellieren kann, recht genau in der Umsetzung abgebildet.

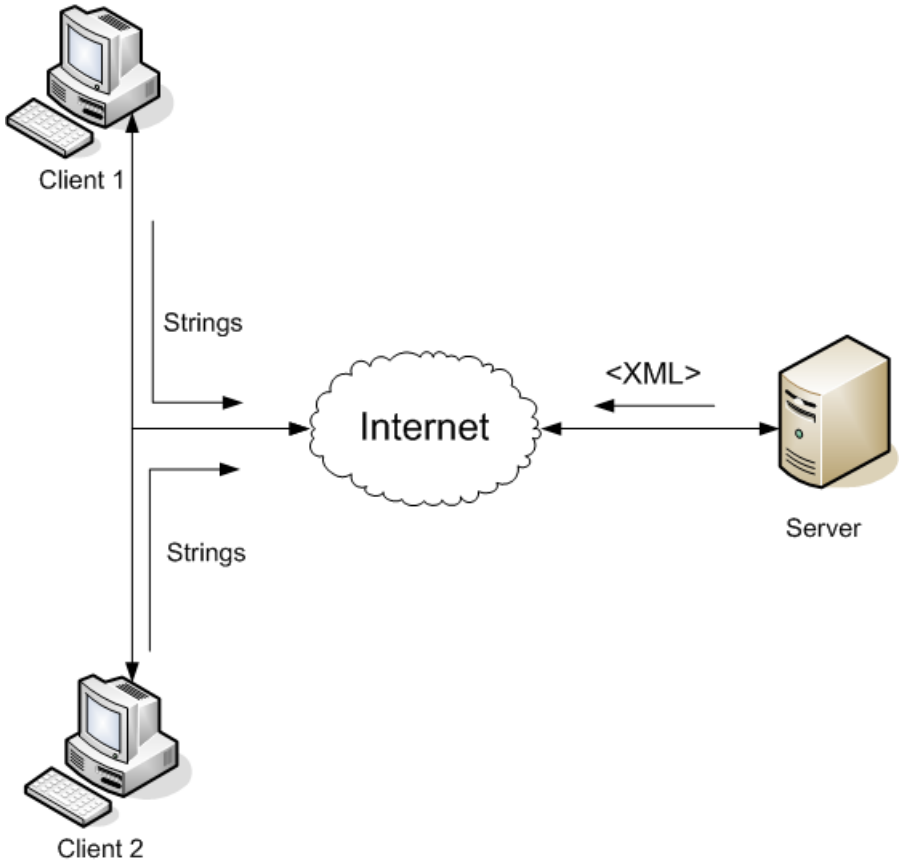

Abbildung 2.5: Übertragung der relevanten Daten

Kapitel 3

Ausblick

In diesem Abschnitt wollen wir nun unsere Erfahrungen nach der Präsentation unserer Ergebnisse darlegen, sowie einen Ausblick geben, was unsere nächsten Schritte sein werden.

3.1 Geplante Änderungen nach der Präsentation

Während der Vorstellung unserer Ergebnisse wurde an uns herangetragen, dass wir die Datenhaltung auf dem Server nicht mittels XML Dateien realisieren sollten, sondern die MySQL Datenbank benutzen sollten. Dies sei sowohl sicherer, was die Zugriffe angeht, als auch performanter. Während unsere XML Strategie aus unserer Sicht sehr wohl, in dem gegebenen Rahmen, mit der Performanz einer Datenbank mithalten könnte, fällt das Argument der Zugriffssicherheit bei uns auf fruchtbaren Boden. Genaueres beschreiben wir in Abschnitt 3.2.

Vor allem die Anbindung an die Nutzerrechte muß noch mit den entsprechenden Projektgruppen ausgearbeitet werden. Auch die Synchronisation mit dem Stream über das zu entwickelnde Playerobjekt muss in diesem Rahmen noch beleuchtet werden.

3.2 Geplante Änderungen

Zunächst werden wir das, was bisher im Chat mit XML Dateien geregelt wird, komplett auf MySQL umgestellt. Dazu wird der Chatverlauf in eine Tabelle geschrieben und eine aktuelle Chatliste, in der alle Nutzer eingetragen werden, findet ebenfalls in einer Tabelle Platz.

Grundsätzlich wird dann der geplante PDF Export der Chatlogs direkt aus der Datenbank geschehen und ohne ”Formatting Objects” auskommen.

Aufbauend darauf wird dann auch die Verwaltung auf diesen Tabellen beruhen, was wahrscheinlich die Implementierung im Gegensatz zur Arbeit mit versch. XML Dateien vereinfacht.

3.3 Nächste Schritte

Unsere nächsten Schritte werden die besprochenen Änderungen am Persistenzsystem sein, wie wir sie in 3.2 aufgezeigt haben.

Danach werden wir die Anbindung an einen Stream anhand eines Dummy-Objektes testen, damit wir die Synchronisation mit den Real-Streams rechtzeitig implementieren können.

Ferner werden wir nach Absprache mit den entsprechenden Projektgruppen eine Anbindung an das reale Playerobjekt und das Benutzerobjekt implementieren.

Parallel dazu werden wir die Verwaltung der Chats aufbauen und diese dann am Ende an alle anderen Teile anknüpfen.