



7363 - Web-basierte Anwendungen: **Übung 05**

Web Services, Teil 1:
XML-RPC
Monitoring & Debugging



- Teilziele
 - Praktische Erfahrungen mit XML-RPC: Ein Admin-Beispiel
 - Inbetriebnahme eines Werkzeugs für HTTP-Debugging
- Übungen
 - `tcpmon`: Ein Monitor für HTTP-basierte C/S-Anwendungen aus dem Apache Axis-Paket (V 1.x)
 - Aufbau eines XML-RPC Servers
 - Aufbau eines dazu passenden XML-RPC Clients
 - Anwendung: Überwachung einer Serverfarm
- Übergeordnetes Ziel
 - Generelles Verständnis für Web Services
 - Vorbereitung für die Projektarbeit mit SOAP



`tcpmon` :

Ein Werkzeug zum HTTP-Debugging



- Quelle
 - \$srcdir/05/axis.jar.
- Installation
 - Kopieren Sie "axis.jar" in Ihren Bereich oder nutzen Sie direkt die Datei im Dozentenverzeichnis.
 - Aufruf: Am besten per "alias" in ~/.bashrc, nach folgendem Muster:

```
alias tcpmon=  
    'java -cp /path/to/axis.jar org/apache/axis/utils/tcpmon'
```
 - Alternativ auf der Kommandozeile, mit optionalen Argumenten:

```
java -cp /path/to/axis.jar org/apache/axis/utils/tcpmon  
    [listenPort targetHost targetPort]
```
- Konfiguration
 - Entfällt. Alles was Sie benötigen ist ein 2. Port (siehe Funktionsweise)



- Funktionsweise
 - `tcpmon` arbeitet auf der TCP/IP-Ebene (daher der Name)
 - Das Tool wird zwischen HTTP-Client und -Server geschaltet:
 - Der Client baut eine Verbindung zu `tcpmon` statt zum Server auf
 - `tcpmon` reicht diese Daten an den Server weiter und umgekehrt.
 - Die durchgereichten Daten zeigt `tcpmon` an!
 - Wählen Sie dazu einen freien Port
 - Teilen Sie dem Client mit, den Server auf diesem Port zu erwarten.
 - Konfigurieren Sie `tcpmon` so, dass das Tool auf diesem Port lauscht.
 - Teilen Sie `tcpmon` den wahren Server-Port für die Weiterleitung mit.
- Nutzen
 - Auf HTTP beruhende höhere Protokolle wie XML-RPC und SOAP, insbesondere aber ihre APIs, verdecken die auf der HTTP-Ebene ausgetauschten Details.
 - Im Fehlerfall benötigt man Werkzeuge wie `tcpmon`. Es hilft z.B. sehr, die genauen Daten zu erfahren, die ein Web Server erhält.



- Alternativen?
 - FAQ 1: Warum nicht **ethereal/wireshark** nehmen?
 - A: Dieses Werkzeug zeigt (a) zu viel an und (b) unterstützt XML nicht, d.h. protokollierte Daten sind viel schwerer lesbar als mit **tcpmon**.
- Test
 - Verwenden Sie **tcpmon**, um den Datenverkehr zwischen Client und Server Ihrer XML-RPC Anwendung des nächsten Teils mitzulesen.
 - Entsprechen die Client-Daten genau Ihren Erwartungen?
 - Antwortet der Server ebenfalls wie erwartet?
 - Finden Sie Erklärungen für eventuelle Abweichungen!
 - Führen Sie diese Analyse auch aus, wenn keine Probleme mit XML-RPC auftreten, denn sie ist eine Vorbereitung auf die nächsten Praktikums- und Projektaufgaben!



Eine XML-RPC Anwendung



- Szenario
 - Sie sind Systemadministrator(in) einer heterogenen Serverfarm.
 - Sie benötigen eine zentrale Überwachung für bestimmte System-Ressourcen auf allen Servern.
- Der Ansatz
 - Sie besitzen eine zentrale Liste aller zu beobachtenden Server
 - Auf jedem Server betreiben Sie einen XML-RPC Server, der Methoden anbietet, welche die gewünschten Angaben lokal ermittelt.
 - Auf einem der Rechner betreiben Sie einen XML-RPC Client, der für einen beliebigen vorgegebenen Server die gewünschte Auskunft einholt.
 - Allgemeiner: Ein zentrales Admin-Werkzeug ermittelt die gewünschten Angaben für jeden Server per XML-RPC und gibt sie tabellarisch aus.



- Die gewünschten Angaben
 - Plattenplatz: Für jedes Dateisystem ist gewünscht:
 - Gesamtplatz in MB
 - Freier Platz in MB
 - Freier Platz in %
 - CPU-Last
 - Kurz-, mittel- und langfristiger Mittelwert
 - Angemeldete Benutzer
 - Benutzername und (Pseudo-)Terminal

- Quellen für diese Angaben
 - Unter Unix/Linux:
 - Plattenplatz: Systemkommando `df`
 - CPU-Last und Benutzer: Systemkommando `w`
 - [Optional: Unter Windows-2000 / XP: `(tbd)`]



Der XML-RPC Server

- Implementierungssprache:
 - Ihre Entscheidung!
 - Ihr Dozent empfiehlt: Ruby

Zu implementierende Methoden

a) Die Standard-Introspektionsmethoden:

- `array system.listMethods`
- `string system.methodHelp(string)`
- `array system.methodSignature(string)`

b) Die Anwendungsmethoden

- `array monitor.listActiveUsers`
- `boolean monitor.checkActiveUser(string)`
- `array monitor.getLoadAverages`
- `array monitor.getDiskSpace`
- `array monitor.getSelectedDiskSpace(array)`



Einzelheiten zu den Anwendungsmethoden

- `array monitor.listActiveUsers`
 - Ergebnis ist ein Array von Structs
 - Jedes struct besteht aus folgenden name/value-Paaren (value-Typ in Klammern) :
 - `username (string)`
 - `from (string)`
 - `loginTime (DateTime.iso6905)`
- `boolean monitor.checkActiveUser(string)`
 - Zur vereinfachten Überprüfung, ob der angegebene Benutzer auf diesem Rechner angemeldet ist oder nicht.
- `array monitor.getLoadAverages`
 - Ergebnis ist ein Array aus drei double-Werten, vgl. "w"



Einzelheiten zu den Anwendungsmethoden

- `array monitor.getDiskSpace`
 - Ergebnis ist ein Array aus structs
 - Jedes struct besteht aus folgenden name/value-Paaren:
 - `filesystem (string)`, `mountpoint (string)`, `totalSize (int)`, `freeSize(int)`, `freePercent (double)`
- `array monitor.getSelectedDiskSpace (array)`
 - Das Übergabe-Array enthält strings mit den Namen der zu prüfenden Dateisysteme, z.B. "C:" (Windows) oder "/dev/hda1" (Linux)
 - Ergebnis analog zu `monitor.getDiskSpace`
 - Zur Überwachung bestimmter (kritischer) Dateisysteme, auch einzelner.



Der XML-RPC Client

- Implementierungssprache:
 - Ihre Entscheidung!
 - Ihr Dozent empfiehlt: Ruby

Methoden bzw. Funktionen: Analog zu den Server-Methoden

- Aufgabe des Clients ist die Bereitstellung eines API für die Anwendungslogik für die Server-Methoden.
- Je nach Toolkit und Implementierungssprache ist eine Trennung zwischen XML-RPC Client und der eigentlichen Anwendung erforderlich oder entbehrlich.



Die XML-RPC Anwendung (CLI-Variante)

- Name: **sysreport**
- Synopsis:
 - `sysreport -f hostlist | -h hostname -D | -L | -A [params]`
- Optionen
 - `-f hostlist`
hostlist: Datei mit Hostnamen wie "lx2-05", ein Name pro Zeile
 - `-h hostname`
 - Alternativ zu `-f`, für die direkte Übergabe eines einzigen Hostnamens
 - `-D` "diskfree"- Modus
Tabellarische Anzeige, eine Zeile pro hostname
 - `-L` "LoadAverage"-Modus
Tabellarische Anzeige, eine Zeile pro hostname, 1+3 Spalten
 - `-A` "Active Users"-Modus
Tabellarische Anzeige, eine Zeile pro struct, 1+3 Spalten
- params:
 - Optionale Angaben z.B. zur Auswahl von Teilfunktionen. Freiwillig!



Die XML-RPC Anwendung (Client)

- CLI oder WUI?
Wählen Sie, ob Sie die Anwendung
 - a) als Kommandozeilen-Version wie umseitig beschrieben, oder
 - b) als Web-Anwendung mit gleicher Funktionalität

schreiben wollen.

Bei einer Web-Anwendung geben Sie ggf. die Optionen per Formularseite ein und stellen die Ergebnisse als HTML-Tabelle(n) dar.



Die XML-RPC Anwendung

- Fehlerbehandlung
 - Dieses Thema steht hier zwar nicht im Vordergrund, aber je nach Situation muss eine Anwendung geeignet reagieren.
 - Unterscheiden Sie daher
 - Server-Probleme (Fehler auf HTTP-Ebene)
 - RPC-Probleme, die z.B. eine Folge falscher Übergabeparameter oder fehlender Voraussetzungen auf dem Server sein können.
 - Beispiele für RPC-Fehler
 - *Filesystem not mounted*
 - *No such user*
 - *Wrong parameter*
 - Konsequenzen für die Ausgabe:
 - Z.B. Tabellenzeilen oder -zellen mit Fehlertext oder farbl. Kennzeichng.



Die XML-RPC Anwendung

- Hinweise zur Durchführung
 - **Entwickeln Sie zunächst eine einzige Betriebsart.**
 - **Stellen Sie sicher, dass der XML-RPC Mechanismus funktioniert.**
 - **Entwickeln Sie die Benutzerschnittstelle erst, wenn der "Kern" getestet und ausbaufähig ist.**
 - **Nutzen Sie `tcpmon` zum Debugging auf HTTP-Ebene.**
 - Gehen Sie arbeitsteilig vor!
 - Greifen Sie auf das XML-RPC Tutorial und seine Code-Beispiele für diverse Sprachen zurück.
 - Wer in Ruby implementiert, kann die Vorlesungsbeispiele relativ leicht erweitern zur hier erwarteten Funktionalität.



- **Abgabe?**
 - Abnahmegespräch während des nächsten Praktikums!
 - Testen Sie Ihr Client- bzw. Server-Programm auch mit einem Server bzw. Client einer anderen Gruppe, die in einer anderen Sprache implementiert.
 - Führen Sie die Aufgabe auf jeden Fall zu Ende, denn sie ist eine gute Vorübung für spätere SOAP-basierte Aufgaben.
 - Bei Zeitproblemen:
 - Konzentration auf vollständige Implementierung einer der Teilfunktionen, z.B. „diskfree“ oder „Active Users“
 - Interessante Lösungen können auf Wunsch der Gruppe vorgestellt werden.