Erfurt University of Applied Sciences

Fakultt Gebudetechnik und Informatik

Projektarbeit Programmierung 3

Fachrichtung Angewandte Informatik

Thema:	Teamprojekt Programmierung des Datenmodells und grundlegender Funktionen einer Serveranwendung in Java
eingereicht von:	Jrgen Arne juergen.arne@fh-erfurt.de Klemens Arndt klemens.arndt@fh-erfurt.de Richard Wieditz richard.wieditz@fh-erfurt.de Daniel Henneberg daniel.henneberg@fh-erfurt.de
eingereicht am:	28. Februar 2016
Betreuer:	Dr. Steffen Avemarg

Inhaltsverzeichnis

jStrg

Jürgen Arne, Klemens Arndt, Richard Wieditz, Daniel Henneberg

29. Februar 2016

Teil I.

Introduction

1. Goals

JStrg server is a backend for various applications. Applications like these need a server that is able to manage huge amounts of data.
Example for this are:

- Serverapplications which wants to deliver content, but not able to hold up the neccessary disk space.
- Backupapplications
- Cloud Storage - continuous synchronization to the cloud
- Filesharing
- Local applications which bind persistent data at useraccounts and want to serve this over the network.

If you want to run jStrg, there's no need for disk space. You can easily use Cloud Storage from established providers.

2. current status

At this moment only the core of the software is used.

- file handling: uploaded files get automatically forwarded to the destination location (at this stadium commonly a Cloud Storage Provider)
- configuration: jStrg can be configured by a jstrg-conf.conf file for global settings
- application configuration: applications can be registered to jstrg by providing a file named jstrg-[APPNAME].conf
- network communication: a socket is created for each application known to jstrg
- network communication: jstrg reacts to string based requests, allowing an application to send multiple request types to the server (including up/download requests)
- Example Client: additionally in simple_console an sync mechanism is implemented. This can be used to sync local folders to the server.

3. Roadmap

In the next semester the following improvements and extenions are planned:

- Multithreading: one server and many clients can work parallel.
- Network: an functionally interface to clients and other JStrg servers (jStrgConnector).
- Configuration: some improvments have to be done here
- Database: persistant data holding
- (Streaming: jStrg is used to be able to manage an application comparable to YouTube, what causes jStrg to provide a media file streaming solution)
- Exampleclients: SimpleConsole, some existing webapplications
- Interface for an operator
- Delete Files
- Internal Funktions: scrub mechanism
- Internal Funktions: change storage pools
- Internal Funktions: cleanup FileVersions
- Internal Funktions: improvement of deduplication and caching behavior

Teil II.

Construction source/packages

4. jStrg

main

Here is main programm is started. In attention to the settings the networserver or the console is starting.

Environment

global list Contains an global list which is use to hold up data for the main programm.Later the posts should be load dynamicly from the database

initial configuration DThe class Enviroment also contains the global initializations for the logfile and the settings.

seed In the development phase we're used an seed method to setup an initial position with testdata.

4.1. communication_management

This Package contains all known request and answer types that can be received or send by the server (as listed in the JavaDoc)

answers

This Package contains all answers

requests

This Package contains all requests

4.2. exceptions

Self defined expections are collected here

4.3. file_system

This important package implement the internal logical viewing of the filesystem.

User

Represent users with id, passwords and different storage-backends

File/FileFolder

This is main filesystem.Every user has a rootfolder, which can't be viewed from outside. He is used as an container for all top-level elements. Every element knows which filefolder is superordinated to him. This descript the meta data. In the class file are few implimentations which could be outsorced lately. They can easly noticed , because they have "test" in the method name.

FileVersion

Physical data which are stored in the backend. Every file has one fileversion. This fileversion knows the previous, if this is existed. Contains also e.g. size, checksum and timestamp.

Application

Responsible for different applications which are provided from the jStrg server. The applications have them own usebase. You can create one configuration file for one application

Settings

All settings can be done here.

4.4. network_communication

This important package makes the server listen on a port and handling the requests and answers defined in the communication_management package

4.5. simple_console

This is an example for one client. In cause to the development position there at this time no external connections and multithreating. So we implemented this directly in to the project.

commands

Some comandos are filed here.

functions You can find detailed informations in the manual

4.6. storage_management

Implements the total physical administration for the uploaded files.

Location

Interface for other methodes. The only special feature is still the CacheFileLock. Otherwise, only the interface is used.

CacheFileLock

This class implements the locks to CacheFiles. Now we can use the available resources efficiently. With this lock on an file we can setup an need automatically.If you are the first one, the file must be served. An addionally flag mark this and now you can read the file. So several Clients can download one file, without to turnout multiple times from the backend. This class is in the early stage.

DiskConnector, GoogleConnector, S3Connector

Backends which are supported at this moment.It can be all which is understanding java. E.G: Microsoft Azure, Ceph, Tape, NoSQL, Hadoop.

4.7. tests

All test are collected in this package

integration_tests

Tests for the Environment of jStrg

server_tests

this tests pings all known jStrg Clusters and all Webservers registered to jStrg and returns false if a single instance is not reachable

unit_tests

Tests for importants classes.

console_tests

Test for individually console commands. Because many methods throw a chain of actions is here much more tested than it appears.

4.8. lib

External Libraries.The most of them a compiled with marve. others come already pre-compiled from various java repositories.Since only widespread and well-known libraries are used , here has been no detailed indication of the source.

4.9. testdata

These data are used in the unit tests.Due to time constraints here was used in a solution , the only source waived. Instead,real test data are included.

Teil III.

Manual

5. Conditions

5.1. Connectors

Access details

If the connectors for example, Amazon S3 or Google Cloud Storage to be used must be valid credentials available. Due to the nature of these data (If the test quota is exceeded , this produces an invoice for the stored credit card) no test connections are supplied.

Buckets

In the cloud storage providers corresponding buckets and / or project id 's must be created. The exact procedure depends on the provider. The configuration should be largely self-explanatory.

6. Developers Cookbook

6.1. IntelliJ

Instructions for a developer working environment

1. Use as project-root the github root directory.
2. The Java SDK specify. Note the language level 8 is posted.
3. Under File -> Projectstructure -> Modules delete all and add the project-root. Now would IntelliJ have recognized the code / src folder correctly and colorize it blue.Now the import is working again.
4. Under File -> Projectstructure -> Global Libraries add the code/lib folder. Add IntelliJ libs to serve junit.
5. Under "Artifacts" add an new entry .Add "jstrg compile output".Add META-INF with the main class "jStrg.Main". Or use existing.
6. To compile succesful go to "Modules" -> "Paths" add custom Path. IntelliJ not created this automatically. Here is uses "sonstiges/".

Now, the work environment is ready. You can run tests that compile and produce an artifact .

7. Usermanual

7.1. Installation

Artifact

With a functioning development environment Artifact can be generated.

program start

start the program with

```
java -cp "<path to .jar >:<path to code/lib folder>/*" jStrg.Main

```

Alternatively, in the folder ”sonstiges” run ”start.sh” . This script contains all to execute jStrg .

7.2. Configuration

Global Configuration

There is a possibility jStrg a global configuration file to control. This must be in the user’s home directory that runs the Java process . The name is ” .jstrg.conf ”

Example

```
#
# default config file, with default values.
# put this file in your user home and rename it to ".jstrg.conf"
#
logfile=/tmp/jstrg.log
loglevel=finest
console=yes
networkserver=no
user_folder=/tmp/nutzer6/

```

Konfiguration pro Anwendung

Applications are addressed via a separate configuration file . These must be in the home directory of the executing user . The name follows the pattern :

```
$HOME/.jstrg-<Applikationsname>.conf

```

Example of an application configuration file

```
s3=yes
s3_default=jstrg-development
s3_aws_access_key_id=<id>
s3_aws_access_key=<key>
bytes_per_s3_bucket=5368709120
disk=yes
disk_default=/home/henne/test/
gcloud=yes
gcloud_bucket=jstrg-bucket
gcloud_projectid=jstrg-development
gcloud_json_credentials=/home/henne/.gcloud-key.json
bytes_per_google_bucket=5368709120
cache_default=/home/henne/test/cache/

```

7.3. Console

Um die Konsole zu benutzen, muss in der globalen Konfiguration

```
console=yes

```

gesetzt sein.

Allgemeine Funktionen

- Tab Completion: Kommandos und Pfade können mithilfe von <tab> komplettiert werden. Doppelt <tab> zeigt verfügbare Optionen an.
- Hilfe: unbekannte Kommandos rufen automatisch eine Kurzbeschreibung der Kommandos auf
- Kommandohistorie: mit den Pfeiltasten auf und ab kann die Historie durchgeblättert werden.

wichtige Befehle

context

Der Kontext kann gewechselt werden. ”admin” oder <userid> sind als Argument akzeptiert. Zum testen kann Nutzer ID 6 verwendet werden. Dies ist ein leerer Testnutzer.

prop

Nutzerspezifische Einstellungen. Momentan kann nur der user_folder eingestellt werden. Dieser muss vorher existieren. Dient als Wurzelverzeichnis für das sync Kommando.

put

Lädt eine Datei auf den Server hoch und platziert es im aktuellen Verzeichnis. Nach einem sync würde es auch im Wurzelverzeichnis liegen.

jstrg sync

Synchronisiert das Wurzelverzeichnis mit dem Server.

rollback

Frühere Versionen einer Datei können mit ”rollback” angezeigt werden. In der Ausgabe erscheinen Nummern an den Zeilen. Mit ”rollback <datei> <nr>” kann zu einer bestimmten Version zurückgekehrt werden. Der ”mofification-timestamp” der Datei wird auf die Zeit des rollbacks gesetzt.

mkdir

Ein Verzeichnis auf dem Server erstellen.

cd

In ein Verzeichnis wechseln. Es kann immer nur ein Level angeben werden. Zusätzlich gibt es noch ”cd ..” für das Übergeordnete Verzeichnis. Ein ”cd” ohne Argumente wechselt in das Wurzelverzeichnis.

7.4. Beispiel Benutzung

```
henne@xmg ~ $ mkdir /tmp/nutzer6
henne@xmg ~ $ projekte/jstrg/sonstiges/start.sh
...
> context switch 6
switched to user
> prop get user_folder
/tmp/nutzer6/
> prop set user_folder /tmp
new root directory: /tmp
> prop get user_folder
/tmp
> prop set user_folder /tmp/nutzer6
new root directory: /tmp/nutzer6
> ls
found: 0
> mkdir testfolder
created new folder: /testfolder
> put /tmp/jstrg.log
jstrg.log        jstrg.log.lck
> put /tmp/jstrg.log
uploading file: jstrg.log
upload successful
> ls
testfolder/
jstrg.log
found: 1
> cd testfolder
current directory: /testfolder
> put /tmp/jstrg.log
uploading file: jstrg.log
upload successful
> ls
jstrg.log
found: 1
> jstrg sync
completed without errors
> put /tmp/jstrg.log
```

```
uploading file: jstrg.log
upload successful
> jstrg sync
completed without errors
> ls
jstrg.log
found: 1
> rollback jstrg.log
1 -> Version vom Sat Feb 27 13:16:15 CET 2016, size: 6378
found 1 versions
> rollback jstrg.log 1
rollback file: success
> ls
jstrg.log
found: 1
> rollback jstrg.log
1 -> Version vom Sat Feb 27 13:16:37 CET 2016, size: 11073
2 -> Version vom Sat Feb 27 13:16:15 CET 2016, size: 6378
found 2 versions
> rollback jstrg.log 1
rollback file: success
> jstrg sync
completed without errors
> exit
henne@xmg ~ $ ls -lR /tmp/nutzer6/
/tmp/nutzer6/:
insgesamt 4
-rw-r--r-- 1 henne users 4007 27. Feb 13:14 jstrg.log
drwxr-xr-x 1 henne users   18 27. Feb 13:16 testfolder

/tmp/nutzer6/testfolder:
insgesamt 12
-rw-r--r-- 1 henne users 11171 27. Feb 13:18 jstrg.log

```