FEED READER APIs

Author: Suneel Karnam

Table of Contents

1. Over view: 3

2. Data Structures 3

3. Unit Tests: 3

4. Tools/Libraries 4

5. Deployment: 4

6. APIs 5

6.1 Subscribe feed: 5

6.2 UnSubscribe feed: 6

6.3 Add article to a feed: 7

6.4 List all articles of a feed 8

6.5 List all the articles along with the feeds subscribed by the user 9

7 Improvements 10

8. Appendix 10

8.1 Install grails 3.0.0 10

8.2 Run Unit and integration Tests 10

8.3 Compile and start application in dev (local) 11

**Feed Reader APIs**

# 1. Over view:

The “**feedreader”** application provides the APIs to subscribe/unsubscribe a feed by the user. The feeds have articles and articles can be added to the feeds using the APIs defined below. The API also gets the list of articles for a feed and also gets the articles along with feeds subscribed by an user

# 2. Data Structures

The application uses **ConcurrentMap** available in **java.util.concurrent** package in Java to store all the data in the memory. The **ConcurrentMap** is **thread safe** and the data can be added/removed/updated by multiple concurrent users without compromising the state of the data. Also while adding the data the thread safe method **putIfAbsent** is used which is atomic. The get will be consistent most of the times with all the updates available at the start of iteration.

Also to save the application state i.e all the data (feeds, articles, users) I have chosen to write to a file for each transaction. First the items are pushed into a **BlockingQueue** and continue without waiting. A separate thread takes the item from the Blocking Queue and writes to a file. The data files are located in **data/userfeeds/ userfeed\_<userId>\_data.txt** and **data/feedarticles/feed\_articles\_<feedId>\_data.txt**. The data is partitioned and stored into a separate file in **append only mode** for each User and Feed for avoiding race conditions and efficiency.

While starting the application the data **gets bootstrap loaded** parsing the data files generated and populating the appropriate **ConcurrentMaps**. Each operation like SUBSCRIBE/UNSUBSCRIBE/ADD is applied from top to bottom on the data which results in adding or removing from the map. The bootstrapping helps to restore the data even if the application is shit down

Also while stating application few users, feeds and articles are loaded

# 3. Unit Tests:

* The unit tests are provided **build/reports/tests/index.html from the feedreader (root)**
* Open index.html in a browser
* The source code for the unit tests are found in the folders **src**/test and **src-integration-test** provided along the source code

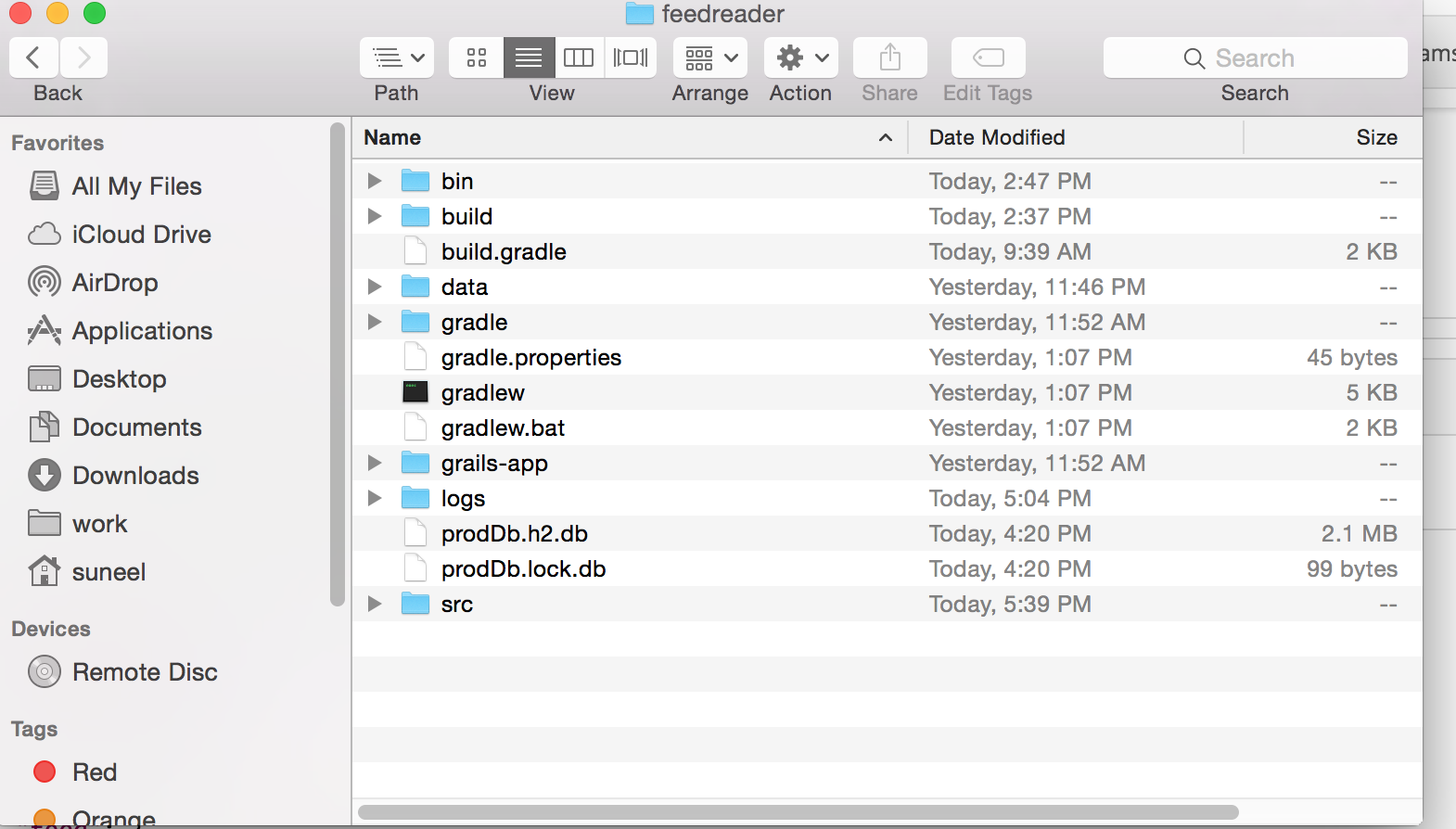
# 4. Tools/Libraries

* Java: 1.8.0\_45
* Grails: 3.0.0
* IDE: STS
* Container: Tomcat

Note: No need to install any tools except java 1.8 and please follow the instructions below to start the application.

# 5. Deployment:

1. Clone the repository from <https://github.com/suneelscanadu/feedreader.git>
2. From the command prompt cd to project root folder which is **feedreader**
3. The directory structure will be



1. From command prompt type **java -Dgrails.env=prod -jar build/libs/feedreader-0.1.war**
2. The **tomcat** starts automatically in port **8080**
3. The application url is **http://localhost:8080**
4. Open any Rest client from the chrome plugin and start using the following urls
5. Please refer appendix for compiling the code locally and running the unit tests

# 6. APIs

## 6.1 Subscribe feed:

The user subscribes to a feed. The success response returns all the feeds subscribed by the user

**URL: http://localhost:8080/userfeed/v1/subscribe**

Http Method: **POST**

Parameters

| **Field** | **Type** | **Description** |
| --- | --- | --- |
| userId | Long | \* User ID created during application start. Default are from 1 to 100 |
| feedId | Ling | \* Feed ID created during application start. Default are from 1 to 100 |

Sample request

{

"userId": 23,

"feedId": 45

}

success message:

HTTP/1.1 200 OK

[{

"feed": {

"description": "This is feed# 45",

"feedId": 45,

"name": "feed\_45"

}

}]

Error-Response when the UserId doesn’t exist

HTTP/1.1 400 Bad request

{

"message": "User not found"

}

Error-Response when the FeedId doesn’t exist

HTTP/1.1 400 Bad request

{

"message": "Feed not found"

}

## 6.2 UnSubscribe feed:

The user unsubscribes to a feed. The success response returns all the feeds subscribed by the user. If no feeds an empty response is sent as shown

url: **http://localhost:8080/userfeed/v1/unsubscribe**

Http Method: **POST**

Parameters

| **Field** | **Type** | **Description** |
| --- | --- | --- |
| userId | Long | \* User ID created during application start. Default are from 1 to 100 |
| feedId | Ling | \* Feed ID created during application start. Default are from 1 to 100 |

Sample request

{

"userId": 23,

"feedId": 45

}

success message which shows that the feed is deleted and displays all other feeds

HTTP/1.1 200 OK

[{

"feed": {

"description": "This is feed# 66",

"feedId": 66,

"name": "feed\_66"

}

}]

Success message for user where none of the feeds are available other feeds

HTTP/1.1 200 OK

{}

Error-Response when the UserId doesn’t exist

HTTP/1.1 400 Bad request

{

"message": "User not found"

}

Error-Response when the FeedId doesn’t exist

HTTP/1.1 400 Bad request

{

"message": "Feed not found"

## 6.3 Add article to a feed:

The success response returns all the articles for the feed

url: **http://localhost:8080/userfeed/v1/article**

Http Method: **POST**

Parameters

| **Field** | **Type** | **Description** |
| --- | --- | --- |
| feedId | Long | \* feed ID created during application start. Default are from 1 to 100 |
| articleId | Long | \* article ID created during application start. Default are from 1 to 100 |

Sample request

{

"feedId": 23,

"articleId": 45

}

Sample response showing the article # 45 subscribed by the user # 23

HTTP/1.1 200 OK

[{

"article": {

"articleId": 45,

"description": "This is article# 45",

"name": "article\_45"

}

}]

Error-Response when the feed doesn’t exist

HTTP/1.1 400 Bad request

{

"message": "feed not found"

}

Error-Response when the article doesn’t exist

HTTP/1.1 400 Bad request

{

"message": "article not found"

}

## 6.4 List all articles of a feed

url: **http://localhost:8080/userfeed/v1/articles?feedId={feedId}**

Http Method: **GET**

Success message showing all the articles

HTTP/1.1 200 OK

[{

"article": {

"articleId": 71,

"description": "This is article# 71",

"name": "article\_71"

}

}]

Success message when no articles are added to the feed

HTTP/1.1 200 OK

{}

## 6.5 List all the articles along with the feeds subscribed by the user

The success response returns all the articles and the feeds subscribed by the user. If no feeds are subscribed then an empty response is sent as shown

url: **http://localhost:8080/userfeed/v1/user/articles?userId={userId}**

Http Method: **GET**

Success message showing all the feeds and articles for an user Id= 99

HTTP/1.1 200 OK

{

"userId": 99,

"data": [

{

"feedId": 62,

"articles": [

{

"articleId": 78,

"name": "article\_78",

"description": "This is article# 78"

}

]

},

{

"feedId": 64,

"articles": []

}

]

}

Success message when no feeds are subscribed

{

"userId": 13,

"data": []

}

# 7 Improvements

1. Plan to use Jmeter for load testing and to do the benchmarking and capacity planning.
2. Security provisioning for authentication and authorization
3. Metering and versioning
4. Would like to persist the to a storage engine to mitigate some of the key things like ACID, high availability, Partition/replication
5. Micro services might also be an option depending upon the business model and requirements
6. Open for any critics and enhancements

# 8. Appendix

## 8.1 Install grails 3.0.0

1. go to <https://grails.org/download.html>
2. select grails 3.0.0
3. unzip to any location on your computer
4. on Mac open the file ~/.bash\_profile
5. export GRAILS\_HOME=/path/to/grails
6. export PATH=$PATH:$GRAILS\_HOME/bin
7. close the command window and open gain and type grails –version and you will see the following info

| Grails Version: 3.0.0

| Groovy Version: 2.4.3

| JVM Version: 1.8.0\_45

## 8.2 Run Unit and integration Tests

1. Install grails (Please refer 8.1 above)
2. Make sure you java 1.8
3. cd to feedreader (the root of the projet)
4. Type grails test-app

## 8.3 Compile and start application in dev (local)

1. If grails is not installed please install grails 3.0.0 (please refer 8.1)
2. Make sure you have java 1.8
3. cd to feedreader (root of the project)
4. grails compile
5. grails run-app which starts the built in tomcat in 8080
6. type ctrl-c to stop the server