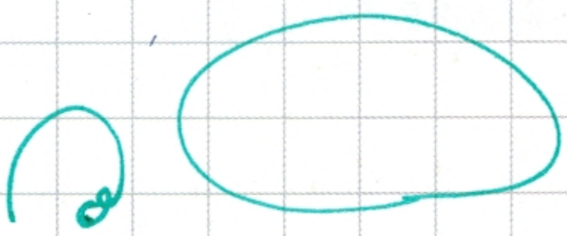
localhost:8080/MeetingScheduler2

↓

js file



Header:

GET /user → User

/location

POST /sessions

DAO

Status: '200'

username Password

- Design pattern for project:

① _ singleton: how many times you run it, give you the same result

↓

Constructor → private

function getinstance() → Call this one

② _ Dependency Injection:

(MVC) - Spring → in one file, we put all the files we need.

Jackson

↓

Convert to jason

- seperates our SW layers.

- Com.sandbox.com.....model.data

↳ SessionTbl.java

↳ UserTbl.java

- Controller:

@RestController → says it is a rest service

@RequestMapping → /users

@RequestHeader

↳ we just say we need this

↳ Spring would build it for us.

@Autowired → says what we need & inject there.

↳ exm: sessionService: all funcs with session

↳ any one who log in to system → he/she has a session.

Logfactor → log :→ in ① Tomcat folder → log file.

has different levels

debug → to see if request worked

info → we don't see this

message

give us → http request

② Hibernate: load obj

↳ obj relational mapping

↳ Data typ → map to Relational DB

Don't need to call DB every time

↓

get Data from DB, cach it, & till it Updates.

↳ tables row → convert to obj & cache.

Save the whole log of Program that shows which program did what & saw our activities too.

↓

beginning of Proj ↓ reading war, : transactions

{ Auth-token → except login

get All users

Ctrl+click → func definition

UserController.java —impl→ Get/users

↳ check token first
↳ validate() → boolean

sessionDAO
↳ Data Access Obj → read from session table & convert it to obj
↓ (sessionTbl) → in SQL
SessionDAO.java
func ↳ getSessionByToken()

{ userId : int
expires: Long
token: String

hibernate ← Session
do this.
↳ with caching → speed up
↳ we make a criteria on SessionTbl
↓
search among row obj ← find everyone with token == token
that cached

↳ Can return one row or zero row of SQL.

{ service
↳ think about Logic of Program
DAO → Provide Data
Model.Data —sql→ tables
↳
the format that we want for Data.

UserTbl.java
@Id → primary key
show
@GeneratedValue
↳ Auto InCrement
↳ Identity
↳ hibernate is responsible to generate this Id with SQL method (Auto InCrement)
@Column → tell program to read which column

if (currentSessions.size() > 0) {

Check, if return →
sth (max 1)
↳ then user was valid → extend expiry → user is active

}

if nothing with that token matched
↳ killSession(Y) token
↳ SessionDao.should Delete this token (fake)!
↓
deleteSession(Session 761 s)
a row obj

@Transactional:
↳ nothing should happen to forget what we want to do
Lesson: when we want to set / Delete sth from SQL

ask Hibernate to delete session

Criteria for search in Table

① CreateSession()
② DeleteSession()
③ UpdateSession()
④ getSessionById() ↳ user who owns the Id
⑤ getSessionByToken()

↓
SessionDAO

Hibernate Control these

{ Sessions expire in 45Min
- inactive for 10min

the whole process should finish then DB would be changed.

Model.DAO →

UserDao:

↳ Repository → Common among all threads

functions { ~~UserDaoClass~~
getAllUsers()
getUserById()
getUserByCredentials(Username, Password)

Logic of our Program → in Service

Controller: { SessionController
UserController
ApplicationController { if /echo get a request then:
Print: I'm running!
↓
to make sure our App is Running.

↓
interface of App for answering requests
/Get
/PUT
/Delete
/Post

web.Contr

SessionController.java → responsible: Authenticate Login / logout!

↳ RequestMapping("sessions")

① { checkToken() ← .Get ↵ ↳ All Requests to /sessions
↳ if False → RestUnauthorized!

· web.messages { ① RESTResponse.java :

↳ All rest service have } statuse

↓

Header & has statuse } response

{ body : we put statuse in here too.

② RESTOK: } Message for statuse 200

↓

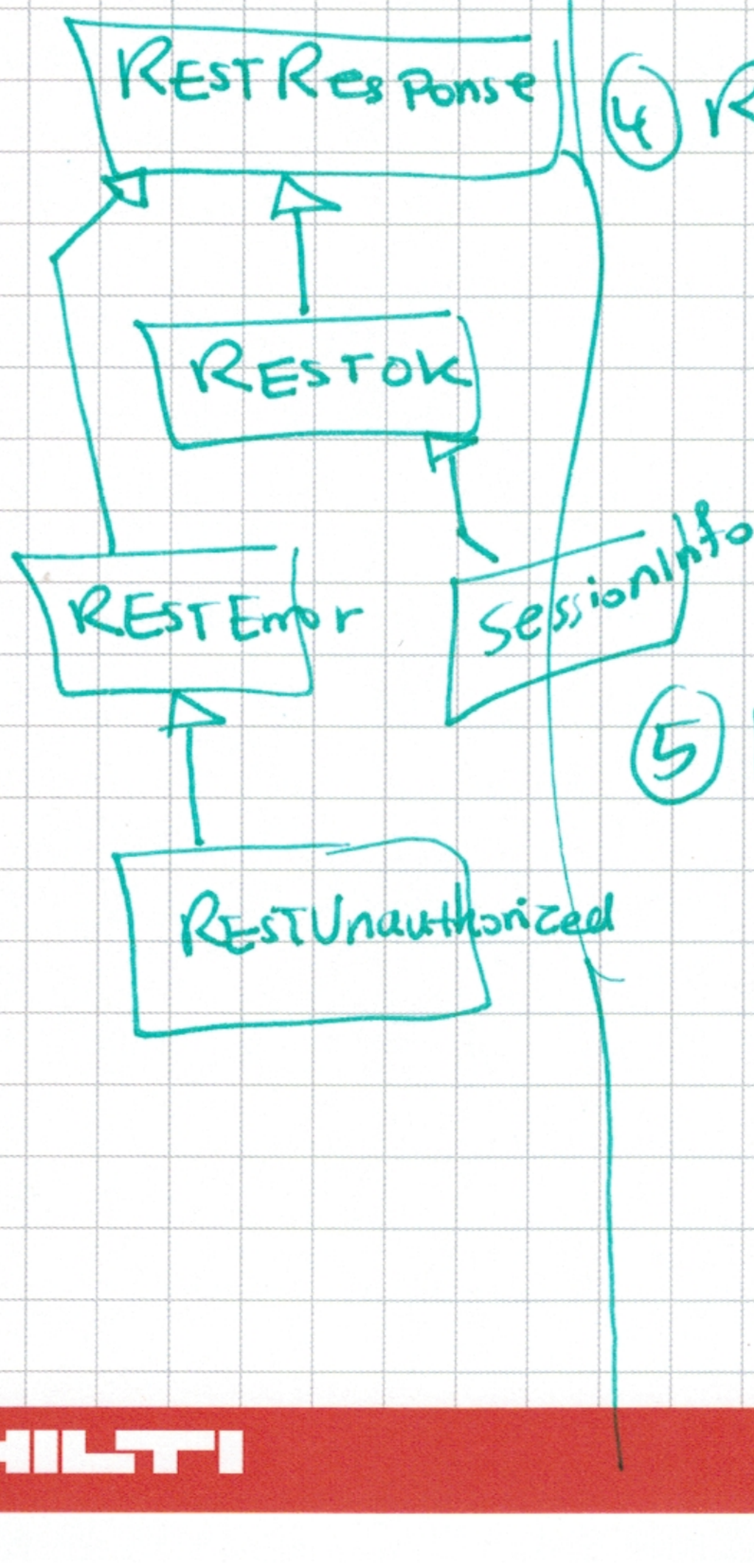
. SC_OK

show Response. message to user. ←

③ RESTError } message String

debug s

for developer



④ RESTUnauthorized.java } status=401;

↓

for anything without login, User wanted to do.

⑤ RESTCredentials.java } Username Password

↓

for log in Users

↓

just an empty form that spring would fill it.

⑥

⑥ SessionInformation.java } UserId, token, expires

for cookie ←

if login was successful → server send token, ... to make a json.

⑦ RESTUser.java:

adaptor → obj is not in the form we want →
we define another obj with desire attributes.

} UserId
UserName
email

⑧ RESTUserList: } list of users

RestOK

↑

⑨ RestUserList Response: } it has a message.
list of users.

whatever extends from RestOK

Statuse 200;

⑩ REST Forbiden } statuse. 403

Predefined message.

Rest error

SessionService.java

① validate(token): T/F

valid / expired / invalid

② isValidSession( )

S.getExpiry().compareTo(systemTime) > 0

↓ true

↓ bca TypeLong

③ killSession()

④ registerSession(userId)

↳ if user was log in before → previous session

↓

getSession

↳ if new user:

token = UUID

↓

uniqueID

expire = systemtime + 6min ← 360000L / ms = session_TimeOut;

⑤ extendExpiry:

add session-TimeOut + expiry.

Password

↓

MySQL

↓

VARBinary

↓

Case Sensitive

① Dependency Injection

↳ Aspect oriented

↳ Data

↳ Access Data

Session Controller
- ② Post /Sessions
  login( )
- ③ /Delete
  logout( )

UserService
- userDao
- setUserDAO( )
- getAllUsers()

Java <bean> in application-Config.xml
- for spring settings
  - → explains Hibernate settings
  - ↳ transactionManager
  - ↳ dataSource → our Real DB

Command log
Console log
- logback.xml → anything you want to log
  - ← log generator
  - ← appender

Pom.xml: has all Dependencies
- artifacts → libraries
- war → web archive → need to be run on web_server ⑨
- jar → java archive
  - ↳ web.xml

Hilroy

Web.xml → servelets for server → deployment descriptor

① dispatcher servelet:
↳ class: Call this class

② server mapping → listen to '/'
↳ respond to this

MVC.Config → define our messages (HTTP) to json

annotation RequestMapping → Handle annotations

json message convertor
↳ obj to json

Component-Scan:

aspects

Model

Web

MVC

jsp: Html .${} → to get html Pages from server