Lehrstuhl für Systeme und Informationsverwaltung

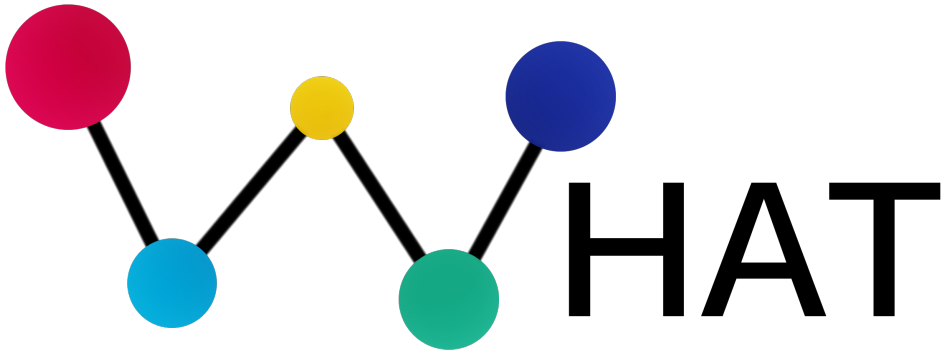
Prof.Dr.Ing. Klemens Böhm
Hoang Vu Nguyen
Marco Neumann

Karlsruher Institut für Technologie

Praxis der Softwareentwicklung (PSE)
WS 2012/2013

Visualizing and Statistically Analyzing Access Behavior to Scientific Databases

Design



WareHouse Analyzing Tool

December 13, 2012

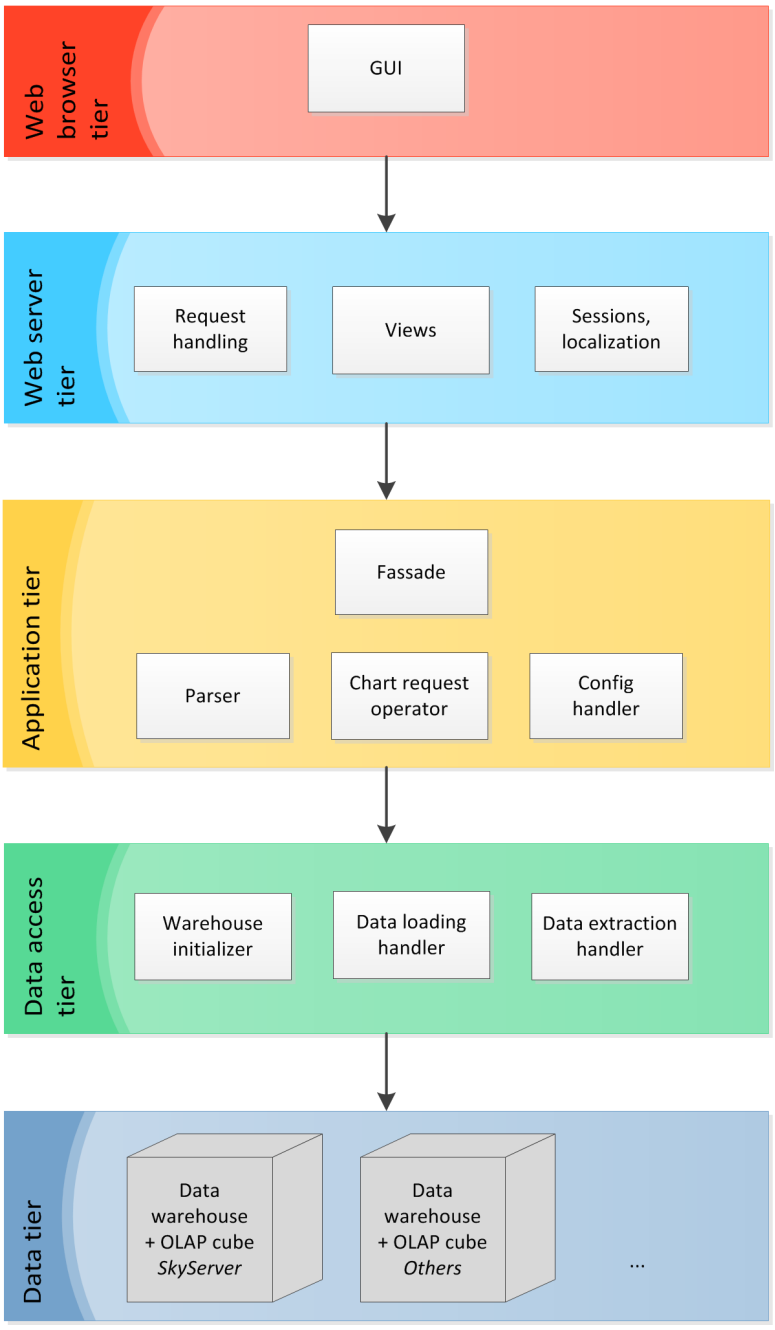
Functional Specification **Alexander Noe**
Design **Jonathan Klawitter**
Implementation **Anas Saber**
QA / Testing **Nikolaos Alexandros Kurt Moraitakis**
Final **Lukas Ehnle**

E-M@il: pse10-group14-ws12@ira.uni-karlsruhe.de

Contents

1 Architecture

WHAT follows an intransparent multitier architecture. The GUI, presentation logic, application processing, data accessing and storing data are logically and also partly locally separated. Intransparent means that communication between tiers just happens between adjacent ones. The different tiers are described below.



1.1 Web browser tier

The web browser tier represents the GUI of the client, which will be displayed as web page. In the following context GUI and web page will be used interchangeably, as they are they same thing. The webpage utilizes javascript. It handles part of the user interaction with the program.

1.2 Web server tier

The web server provides the presentation logic. This includes both the serving of static files as well as the dynamic integration of content in the html pages. Other tasks are session handling, language localization and most importantly request handling.

It handles the rest of the user interaction with the webpage, while invoking the application tier when needed. This tier uses and is tightly integrated with java Play Framework.

1.3 Application tier

The application tier's function is twofold. On the one hand, it handles the parsing process and the management of configuration files. On the other hand, it acts as a sort of middleman between the web server tier and the data access tier.

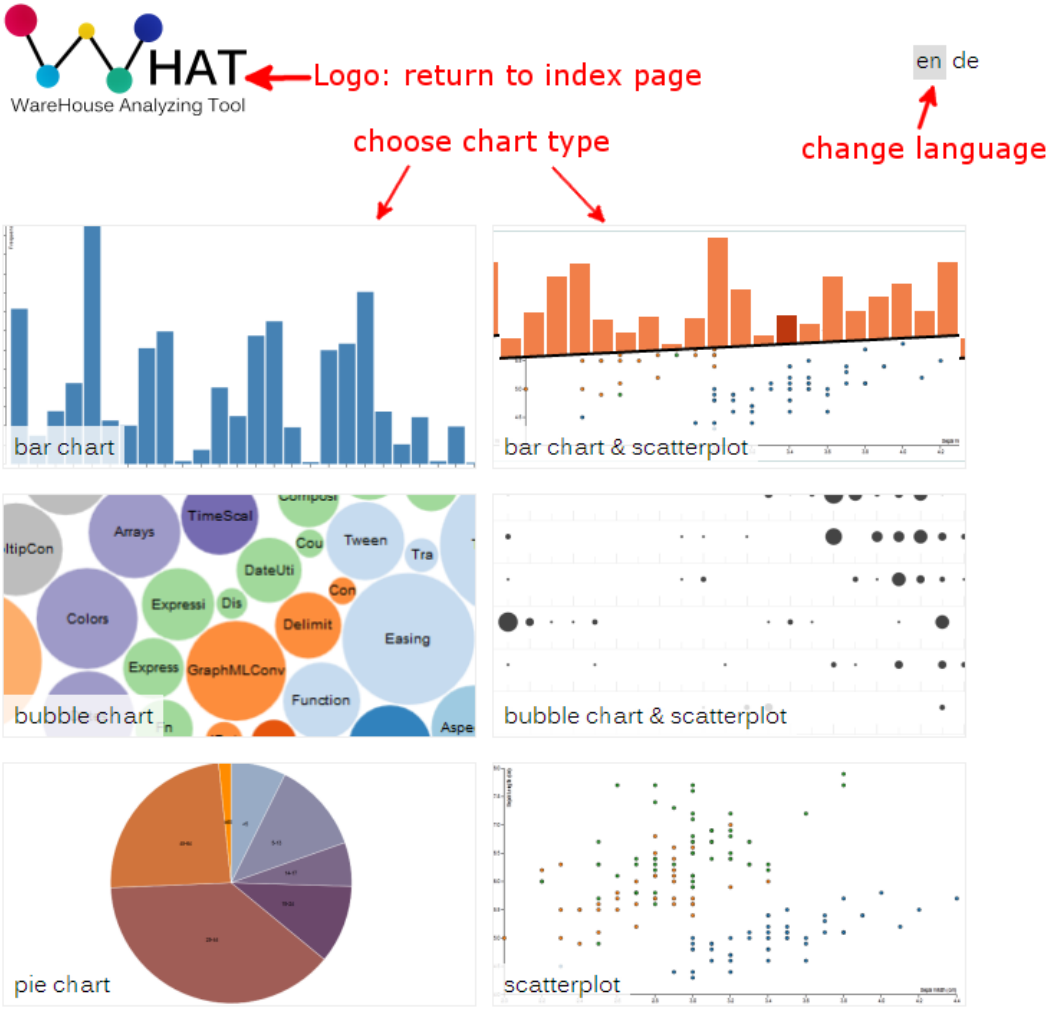
1.4 Data access tier

This tier manages all requests of loading and extracting data from the data tier. So its main task is to build a bridge from the application in Java to the SQL language of the Oracle warehouses and OLAP-Cubes. If there is enough time to implement this optional function, it will also handle the automatic initialization of new data warehouses.

1.5 Data tier

In the data tier the data warehouses and their OLAP-Cubes are stored. This will be done with the Oracle software.

2 Web page design

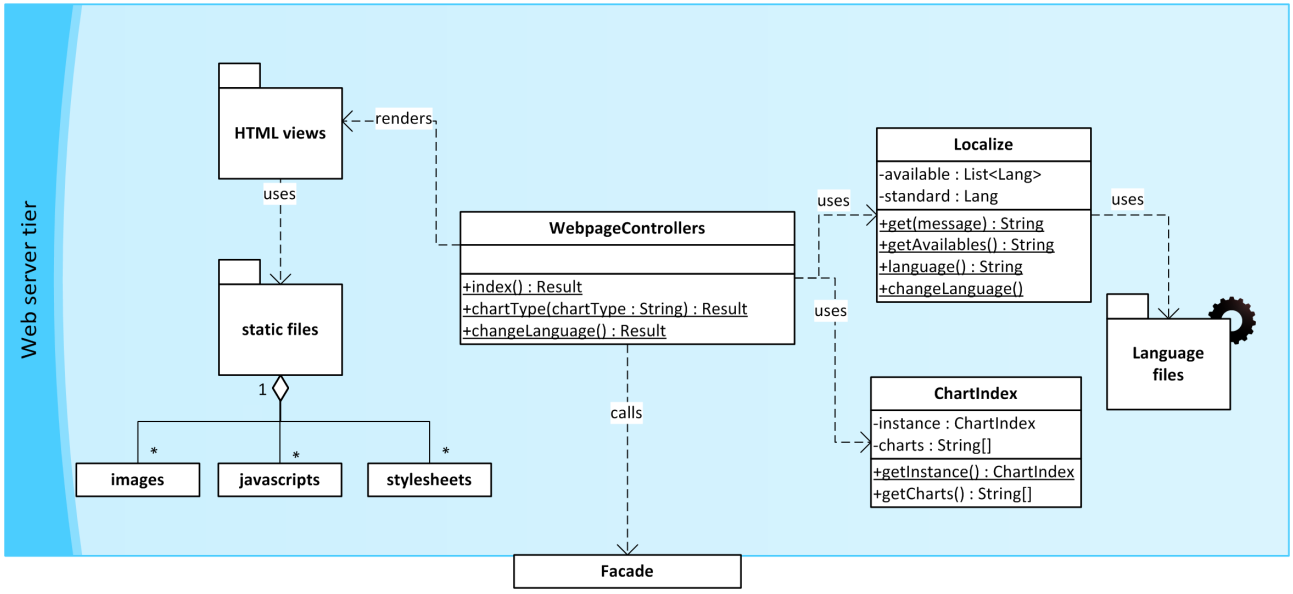


TEXT about admin, blabla, . . .

3 Classes

3.1 Server tier class diagram

This diagram shows the webserver tier of WHAT. It controls the webserver and it can only access the facade of the tier below.



WebpageControllers

All valid HTTP requests are mapped to a method from WebpageControllers. WebpageControllers either uses other helper classes to handle requests itself or makes calls to the facade of the application tier. The last step done by WebpageControllers is to create a valid HTTP response.

HTML views

The HTML views contain the main template for, and all other HTML content of the webpage. They do not have to be static, but can also dynamically integrate some content, e.g. localized strings.

Static files

Static files contains all files of the webpage, which are not changed during runtime. This includes, but is not limited, to images of the webpage, javascript files and stylesheets.

Localize

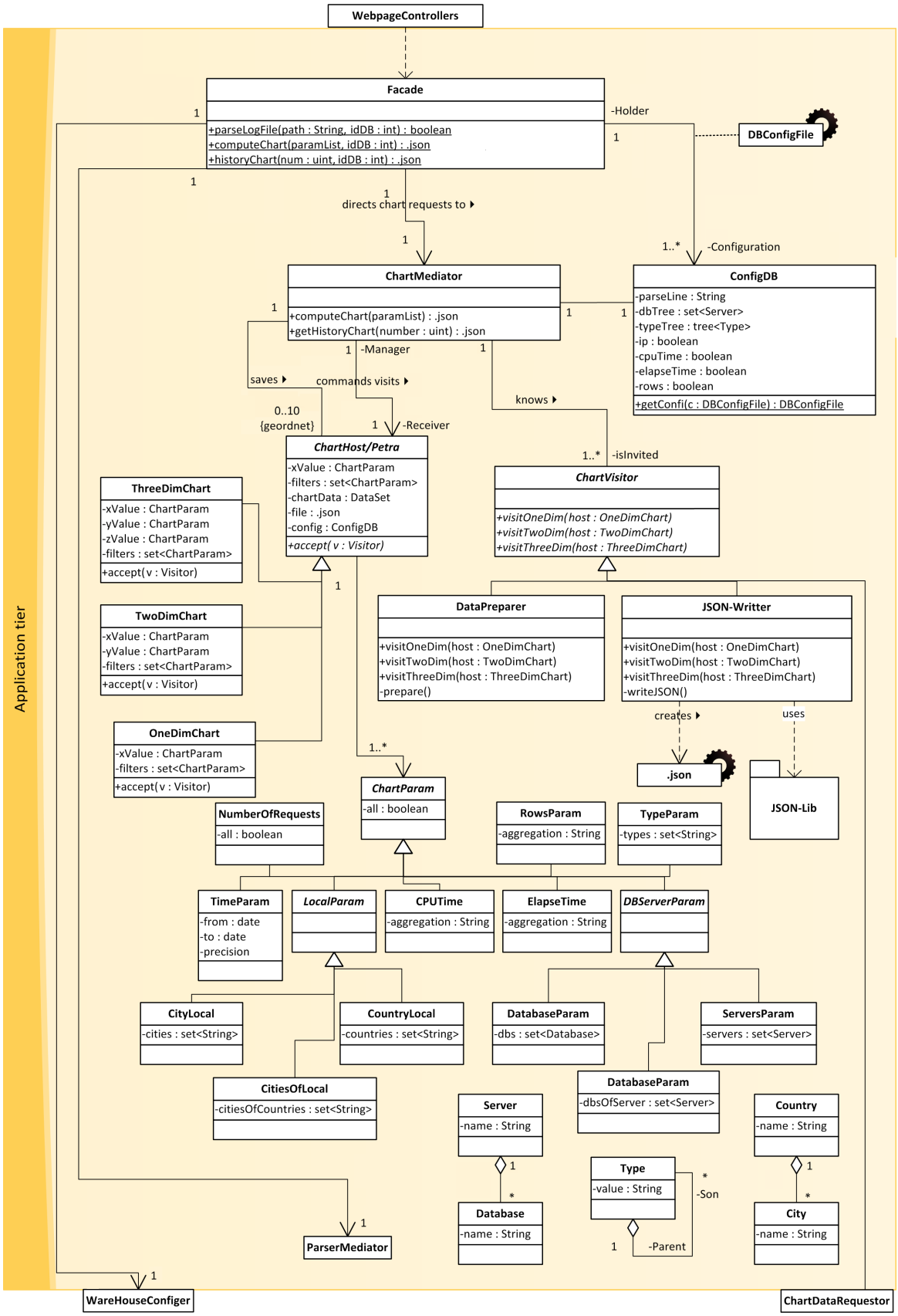
Localize is a static helper class handling all the language related things. It has a method to get localized Strings using the language files, to change the language of the webpage und methods determining all available languages, so they can be integrated in the webpage once they are present.

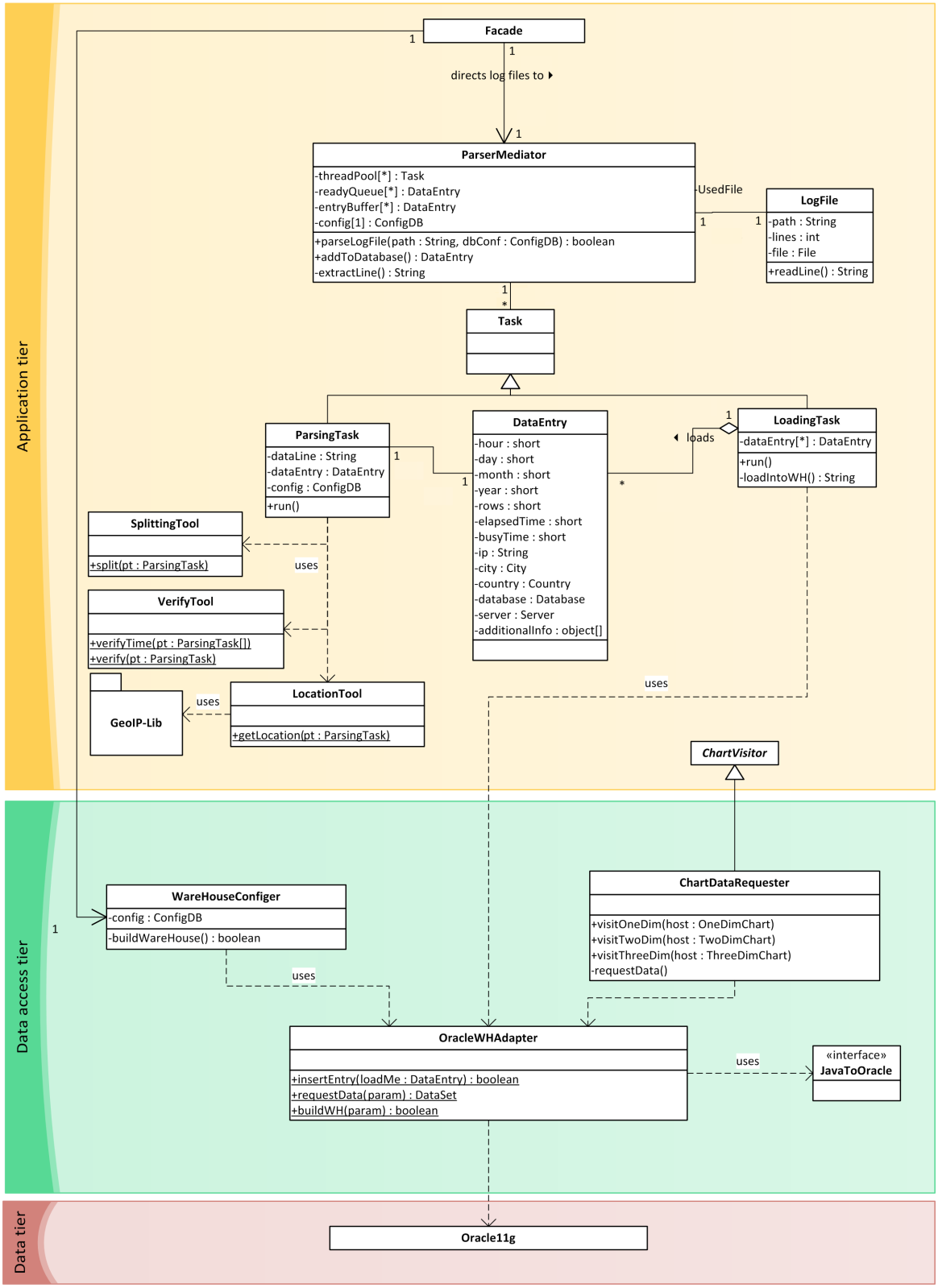
ChartIndex

ChartIndex implements the singleton pattern. On instantiation it scans which chart types are available and saves them internally. This is used by the index page to dynamically display all available chart types, even if one is deleted or added.

3.2 Application & Data access tier class diagram

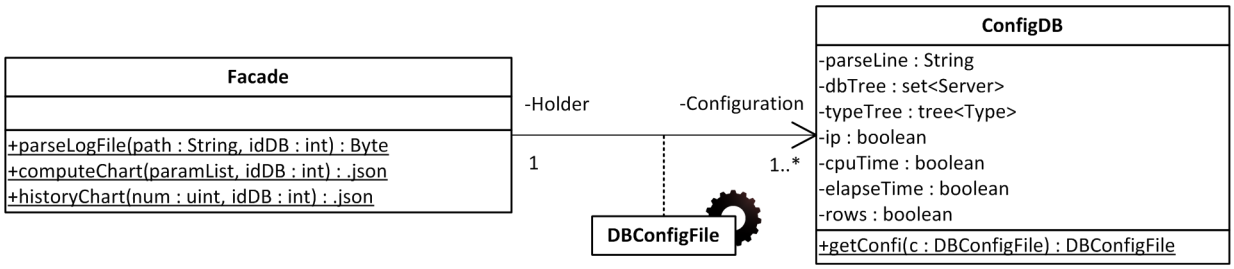
On the next pages the class diagram of the application and data acces tier are displayed.


Application tier
WebpageControllers
Facade
+parseLogFile(path : String, idDB : int) : boolean
+computeChart(paramList, idDB : int) : .json
+historyChart(num : uint, idDB : int) : .json
-Holder
DBConfigFile
directs chart requests to
-Configuration
ChartMediator
+computeChart(paramList) : .json
+getHistoryChart(number : uint) : .json
-Manager
ConfigDB
-parseLine : String
-dbTree : set<Server>
-typeTree : tree<Type>
-ip : boolean
-cpuTime : boolean
-elapseTime : boolean
-rows : boolean
+getConfi(c : DBConfigFile) : DBConfigFile
saves
commands visits
knows
0..10
{geordnet}
-Receiver
-isInvited
ChartHost/Petra
-xValue : ChartParam
-filters : set<ChartParam>
-chartData : DataSet
-file : .json
-config : ConfigDB
+accept(v : Visitor)
ChartVisitor
+visitOneDim(host : OneDimChart)
+visitTwoDim(host : TwoDimChart)
+visitThreeDim(host : ThreeDimChart)
ThreeDimChart
-xValue : ChartParam
-yValue : ChartParam
-zValue : ChartParam
-filters : set<ChartParam>
+accept(v : Visitor)
TwoDimChart
-xValue : ChartParam
-yValue : ChartParam
-filters : set<ChartParam>
+accept(v : Visitor)
OneDimChart
-xValue : ChartParam
-filters : set<ChartParam>
+accept(v : Visitor)
DataPreparer
+visitOneDim(host : OneDimChart)
+visitTwoDim(host : TwoDimChart)
+visitThreeDim(host : ThreeDimChart)
-prepare()
JSON-Writter
+visitOneDim(host : OneDimChart)
+visitTwoDim(host : TwoDimChart)
+visitThreeDim(host : ThreeDimChart)
-writeJSON()
creates
uses
.json
JSON-Lib
ChartParam
-all : boolean
NumberOfRequests
-all : boolean
RowsParam
-aggregation : String
TypeParam
-types : set<String>
TimeParam
-from : date
-to : date
-precision
LocalParam
CPUTime
-aggregation : String
ElapseTime
-aggregation : String
DBServerParam
CityLocal
-cities : set<String>
CountryLocal
-countries : set<String>
DatabaseParam
-dbs : set<Database>
ServersParam
-servers : set<Server>
CitiesOfLocal
-citiesOfCountries : set<String>
DatabaseParam
-dbsOfServer : set<Server>
Server
-name : String
Database
-name : String
Type
-value : String
-Son
-Parent
Country
-name : String
City
-name : String
ParserMediator
WareHouseConfiger
ChartDataRequestor


Application tier
Facade
directs log files to
ParserMediator
-threadPool[*] : Task
-readyQueue[*] : DataEntry
-entryBuffer[*] : DataEntry
-config[1] : ConfigDB
+parseLogFile(path : String, dbConf : ConfigDB) : boolean
+addToDatabase() : DataEntry
-extractLine() : String
-UsedFile
LogFile
-path : String
-lines : int
-file : File
+readLine() : String
Task
ParsingTask
-dataLine : String
-dataEntry : DataEntry
-config : ConfigDB
+run()
DataEntry
-hour : short
-day : short
-month : short
-year : short
-rows : short
-elapsedTime : short
-busyTime : short
-ip : String
-city : City
-country : Country
-database : Database
-server : Server
-additionalInfo : object[]
loads
LoadingTask
-dataEntry[*] : DataEntry
+run()
-loadIntoWH() : String
SplittingTool
+split(pt : ParsingTask)
uses
VerifyTool
+verifyTime(pt : ParsingTask[])
+verify(pt : ParsingTask)
GeoIP-Lib
uses
LocationTool
+getLocation(pt : ParsingTask)
uses
ChartVisitor
Data access tier
WareHouseConfiger
-config : ConfigDB
-buildWareHouse() : boolean
uses
ChartDataRequester
+visitOneDim(host : OneDimChart)
+visitTwoDim(host : TwoDimChart)
+visitThreeDim(host : ThreeDimChart)
-requestData()
OracleWHAdapter
+insertEntry(loadMe : DataEntry) : boolean
+requestData(param) : DataSet
+buildWH(param) : boolean
uses
«interface»
JavaToOracle
Data tier
Oracle11g

3.3 Facade + Configurations

The facade of the application tier and configuration files in this design show a optional feature. According to the database on which is operated - if there are more than SkyServer - just a configuration file for this specific one, has to be created and given to the program. This may give the application to power to manage all functions dynamically according to the database. Facade and ConfigDB handle this configurations.



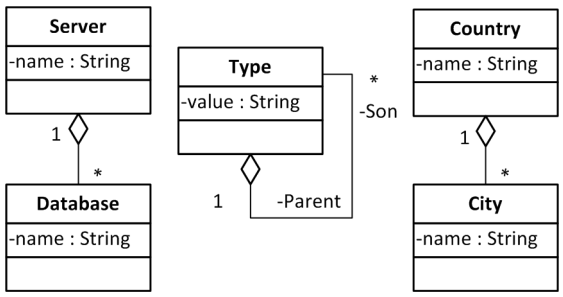
Facade

The Facade is, as you might have suggested, the facade of the application tier. This means the only way to communicate with this tier is via this class. On of its task is to get the right ConfigDB for any call. With this he forwards them to the corresponding mediators or workers. So new log files are passed to the ParserMediator and chart requests to the ChartMediator. Some other optional calls, like the history of charts or the configuration of a new Warehouse are just adumbrated.

ConfigDB

The ConfigDB class provides a static factory for creating itself out of a configuration file, which may use caching. Objects of ConfigDB store the information about the database they describe and while be used by all the dynamic methods, which are hooked on specific databases.

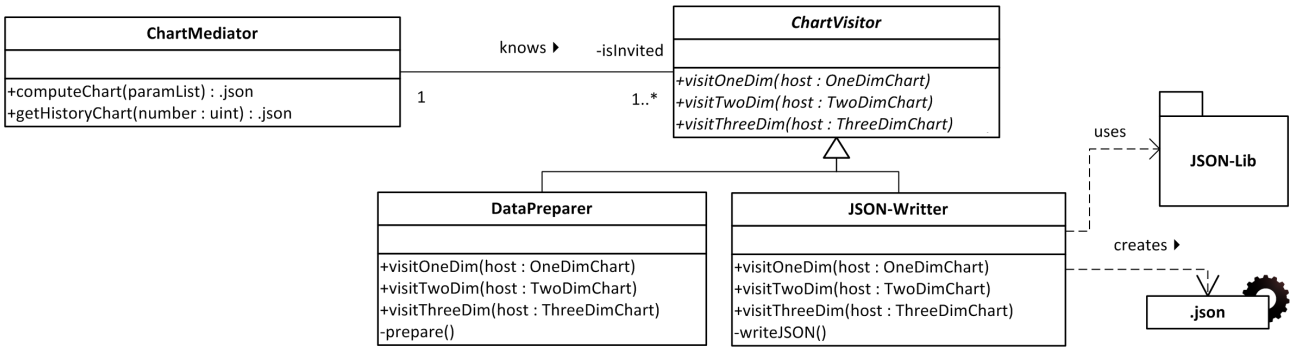
One thing stored in a ConfigDB is a String indicating the form of a line in the log files. Other attributes are booleans for specific measures or dimensions, as well as trees of Strings for dimensions like Databas > Server or the Type.



Server, Database, Type, Country, City

All this classes are Stringwrappers and also build as trees. They stand exemplary for possible things stored in a ConfigDB. The implementation of them may be way more dynamic. Whereas the content of Server & Database and Type are depended on the configuration, the Country & City tree is mostly hooked to the GeoIP library.

3.4 Chart request operator



ChartMediator

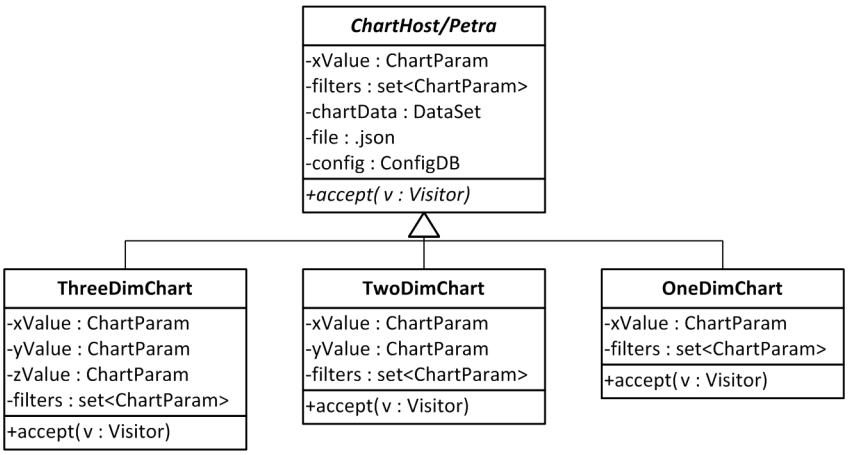
ChartVisitor

DataPreparer

JSON-Writter

.json

JSON-Lib



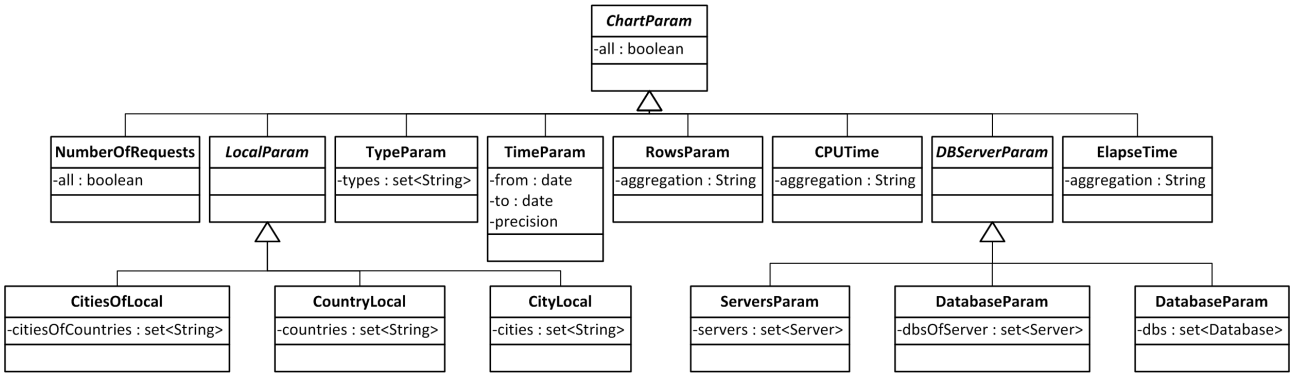
ChartVisitor

OneDimeChart

TwoeDimeChart

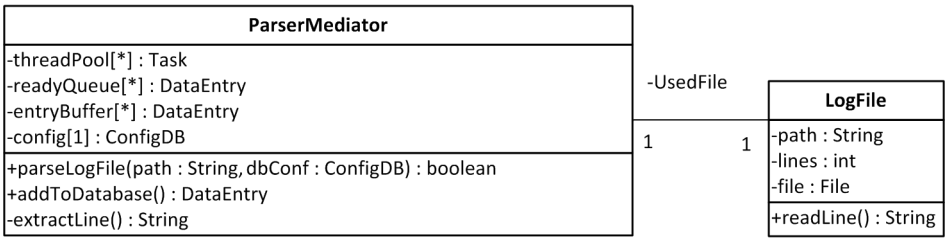
ThreeDimeChart

3.5 Chart parameters



ChartParam

3.6 Parser



ParserMediator

ParserMediator is the ’main’-Class of the Parser. It creates and administrates a thread-pool, which contains several tasks, contains the entryBuffer for finished DataEntries, the stringBuffer for strings, which were extracted from the logfile and saves which log file and configuration file is used.

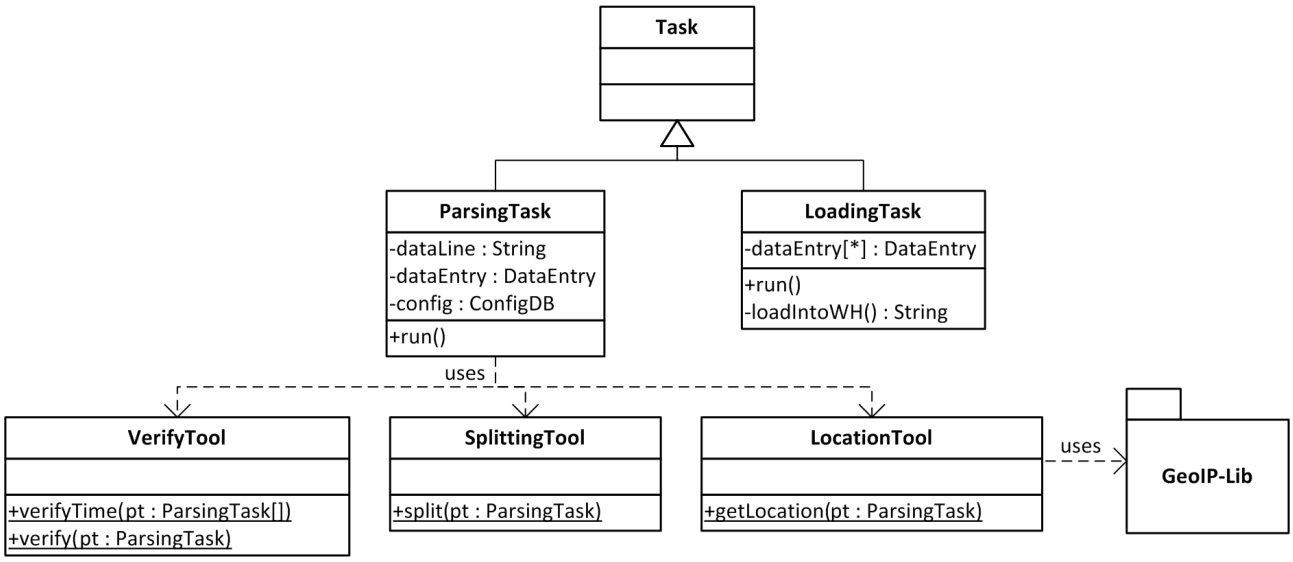
LogFile

The gateway between parser and logfile - it contains the path of the logfile and an integer which saves how many lines have been read from this file. It can read single lines from

the logfile and return them to the Parser.

DataEntry

The dataEntry which will be written in the warehouse. It contains -hour, day, month, year of the request, -rows which were read from the logfile-the elapsed and busy time on the server -the ip from which the request came -the type of request -the database and server which handled the request. It may contain additional information depending on the actual logfile, as specified in the configuration file.



ParsingTask

A ParsingTask is one of the tasks which are created from Parser. It gets a line from the logfile and uses, SplittingTool, VerifyTool and LocationTool to create a dataEntry which contains the same information.

SplittingTool

The splitting tool splits the dataLine from its parsingTask and enters the splitted parts into the dataEntry.

VerifyTool

The VerifyTool checks if the dataEntry is correct - if it has a mistake (e.g. Month = 13 or Elapsed Time = -1), it will be deleted and the VerifyTool sends an detailed error, which can be displayed elsewhere.

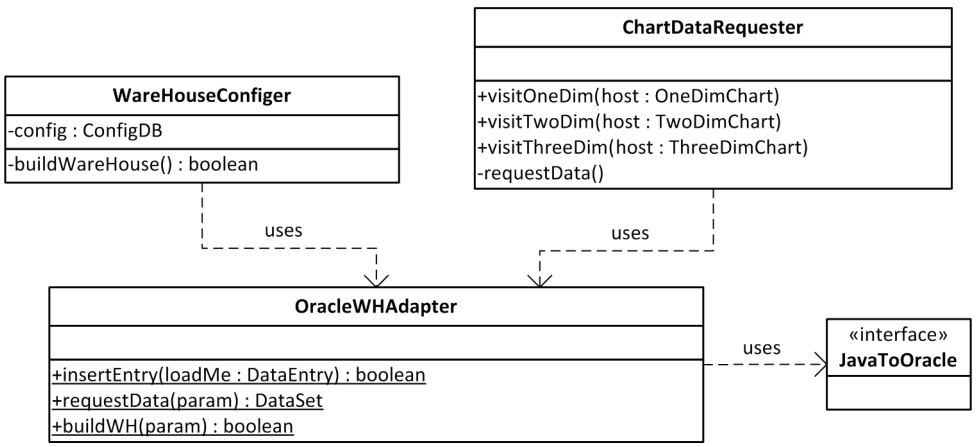
LocationTool

The LocationTool takes the IP from the dataEntry and uses GeoIP-libraries[geo] to determine the city and country of the request. Those will be added to the dataEntry.

LoadingTask

A loading task is another possible task for the threadPool. It takes a finished dataEntry from the buffer and sends it to the warehouse.

3.7 Data access tier



ChartDataRequester

The ChartDataRequester is one of the three visitors of a ChartHost. Its task is to create the queries to gain the data needed for the charts. It takes the informationen about what he has to request from the hosted ChartHost.

OracleWHAdapter

Every query, whether loading, extracting or anythin else, runned against the Oracle server is made by this, and only by this, class. So it represents the interface out of the application to the data.

JavaToOracle

Any library used for the connection and communication with the Oracle server, where the Warehouse is stored. It is only used by the OracleWHAdapter.

WareHouseConfiger

This class will only be needed if a very optional function will be implemented. It's task is to create a WareHouse for one certain new database just with the information of its configuration file information.

geo link to the maxmind geoip library in the library section

4 Data warehouse design

4.1 Overview

4.2 Dimension descriptions

4.3 Measure descriptions

4.4 Type description

As Type depends which data base is operated, it may get it’s own subsection.

5 Sequences

6 Other data

6.1 Static data

6.2 Dynamic data

6.3 Extern data

7 Libraries

FIXME: Needs links and tex in its life.

The program uses a number of external libraries to achieve its functionality. This section describes which they are, how they interact with the program, as well as licensing information.

-Java Play Java play is primarily a web framework, but it includes a webserver too [play]. We use it to create a modern, dynamic and interactive website, instead of a conventional java Swing GUI. http://www.playframework.org/assets/images/logos/normal.png Our web server tier is just a modification of appropriate java play classes[playintegration].

-d3.js D3.js [d3link] is a javascript visualization library for the browser. It allows the creation of beautiful, interactive visualizations. It runs, as one would imagine, in our browser tier.

-Geoip We use the Maxmind geoip Lite[geo] library to map user IPs to locations. This allows queries based on location, as well as preserving anonymity. Our parser uses the geoip library.

-Oracle database and warehouse We host our data in an oracle warehouse. Our data tier consists of them.

-oracle jdbc Oracle jdbc is the oracle database access driver. We use it in the data access tier. //FIXME: be more specific

Licensing information Most of the libraries used are open source, as well as free of charge. The rest are from oracle. We can use them in our product, because our university has bought licenses but I don't know what happens when we release it. FIXME, too

Java play is released under the Apache 2 License [apache2] d3.js is released under the BSD License, BSD 3 Clause [bsd3] The geoip library we use is licensed under the LGPL. [lgpl]

Oracle libraries are released under the oracle license[oracle]

[play]http://www.playframework.org/ [playintegration] see the class diagram for more details [d3link] d3js.org [geo] http://www.maxmind.com/download/geoip/api/java/ [oracle] http://www.oracle.com/technetwork/licenses/distribution-license-152002.html. [apache2] http://www.apache.org/licenses/LICENSE-2.0.html [bsd3] http://opensource.org/licenses/BSD-3-Clause [lgpl] http://opensource.org/licenses/lgpl-license