Giteway 1.0

Technical specifications

Table of content

Deployment

Source code : *git://github.com/Couettos/giteway.git*

Deployed application: http://giteway.cloudfoundry.com

If you wish to deploy giteway in your local environment, please check the deployment instructions in the README file at the root of the project.

Technology stack

Spring 3.1: *Spring-core, Spring-webmvc, Spring-oxm, Spring-context, Spring-aop*

AOP: *Spring-AspectJ*

Logging: *SLF4J – Log4j*

Serializers/Deserializers: *Jackson, Castor*

Rest client: *Apache Commons-http*

Utilities: *LambdaJ-Hamcrest*

Caching: *Spring-EHCache*

Testing: *Junit, Mockito, Spring-test*

Front: *JSP, JSTL, JQuery, Plot*

Global Architecture

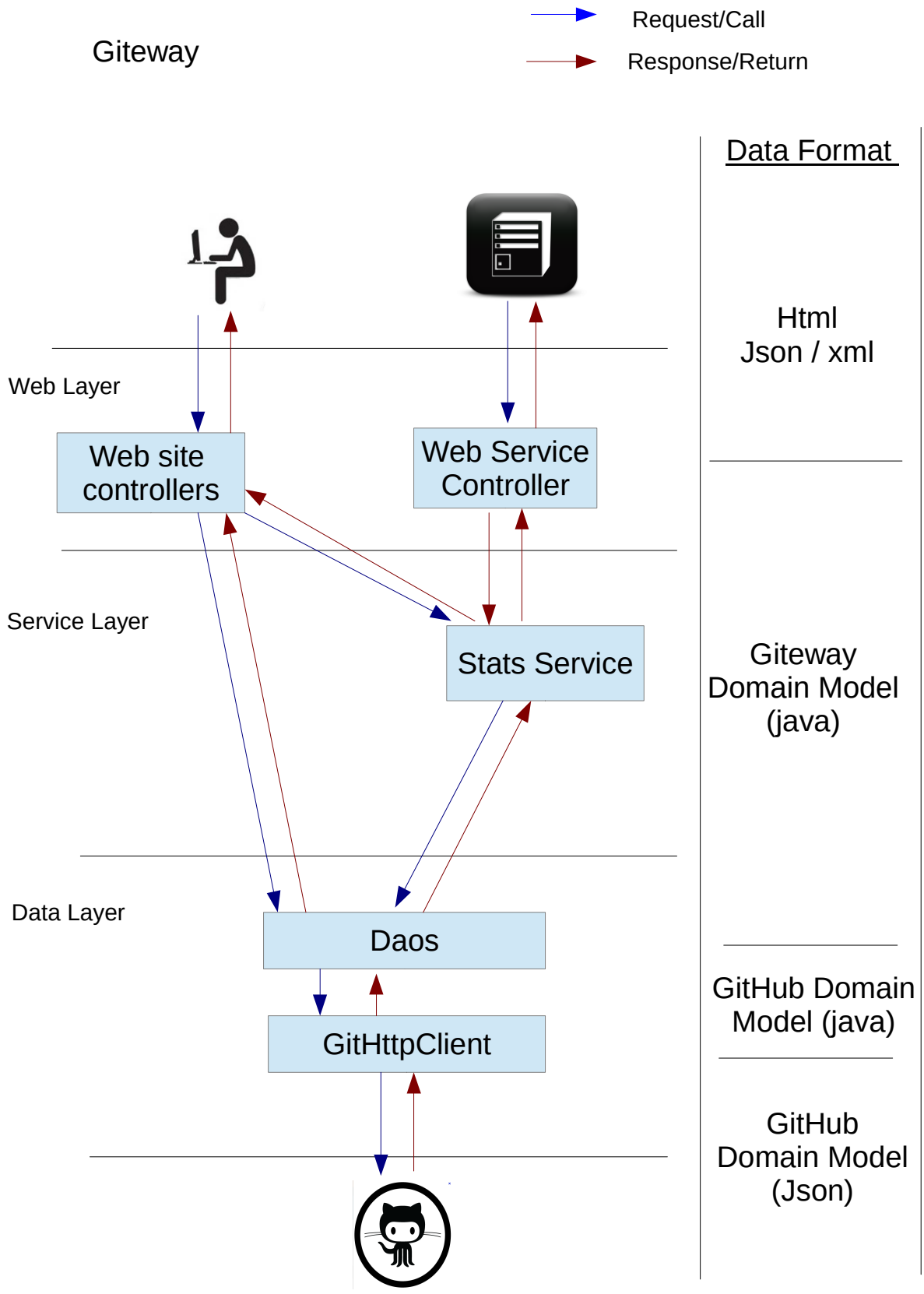


Figure 1 - Global architecture

The model

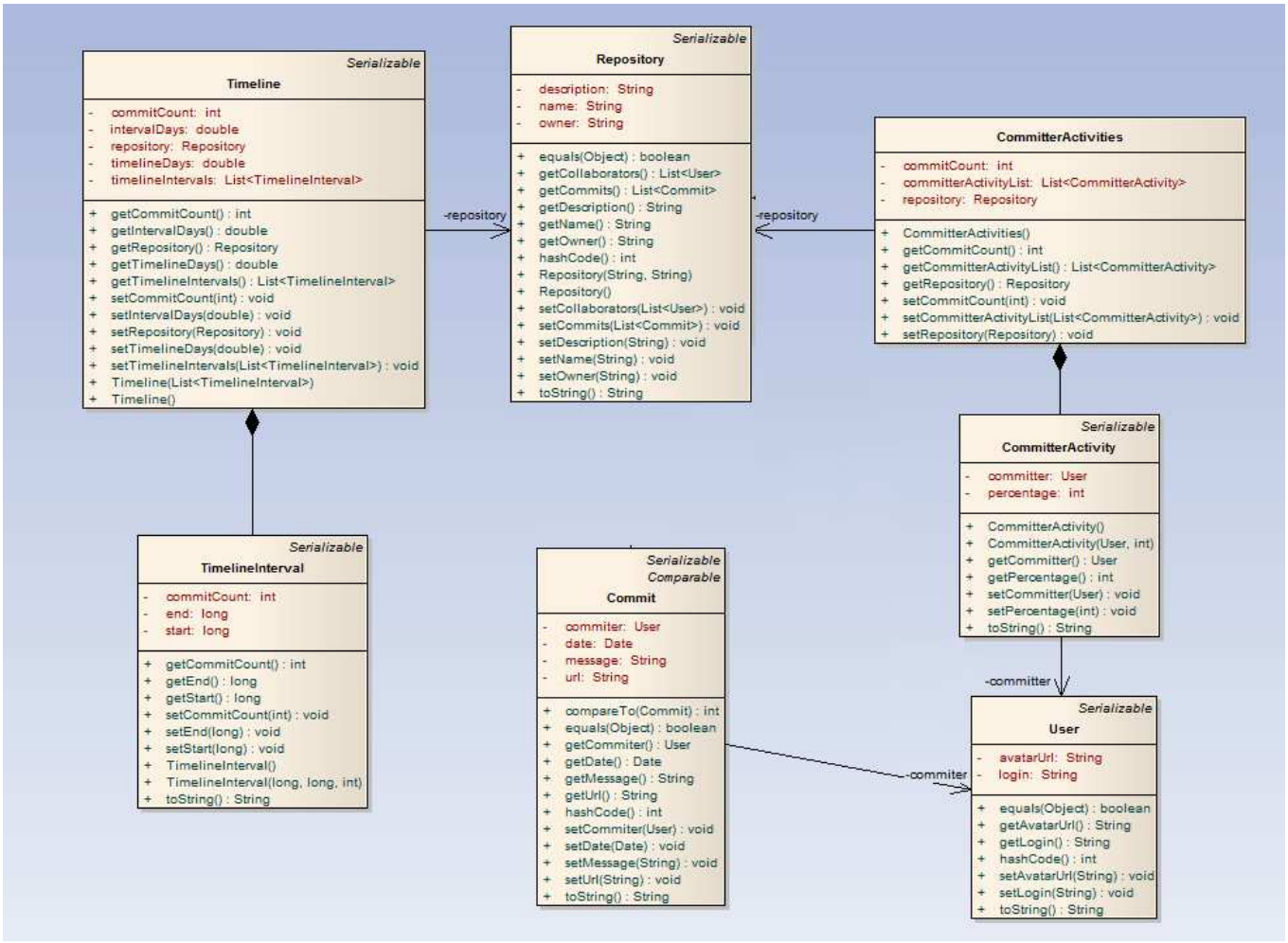


Figure 2 - The model

The model is loosely coupled from GitHub domain model. This architectural choice has been motivated by the inconsistency of github DM. For example, the repositories returned by the keyword search and by the single request do not have the same attributes.

In Giteway, GitHub Objects returned by the API are considered as Value Objects or DTOs.

This approach has advantages:

- Reduce the complexity of GitHub DM.
- Isolate the impacts of a GitHub API version change.

But it also has drawbacks:

- More code has to be written

- Slightly lower performances
- Redundancy of a few objects (User for example)

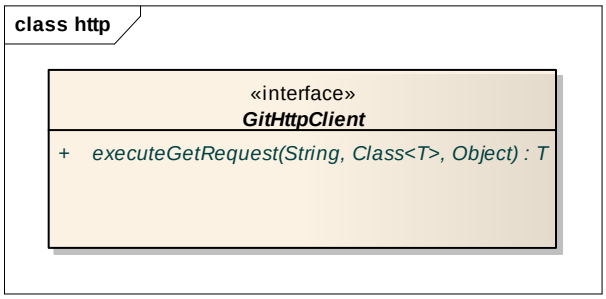
Data Access Layer

The data access layer requests data from GitHub API. It is an abstraction layer over HTTP and GitHub DM.

The data access tier provides a two level abstraction:

1st: GitHub HTTP restful Json API → GitHub DM

The data is accessible over http in a json format. The interface GitHttpClient requests the data and to converts it into a set of mapped class representing the GitHub DM (DTOs).



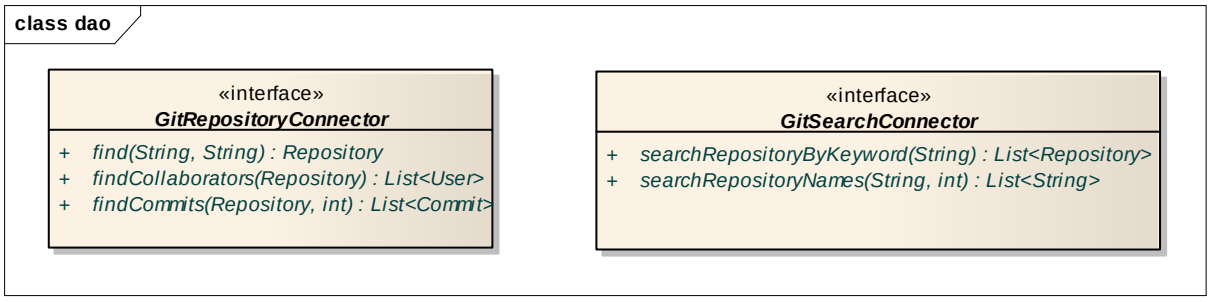
GitHttpClient encapsulates a DefaultHttpClient thread safe singleton object which is instantiated with a PoolingClientConnectionManager. This approach enables DefaultHttpClient to reuse connections from the pool. Between to requests, the connections are kept alive. This is of great interest as the SSL handshake can take a long time.

2nd: GitHub Domain Model → Giteway Domain Model

The DAOs do no expose GitHub DM to the rest of the application.

Basically, the DAOs :

1. call GitHttpClient
2. receive GitHub domain model
3. convert GitHub domain model into Giteway domain model
4. return the data

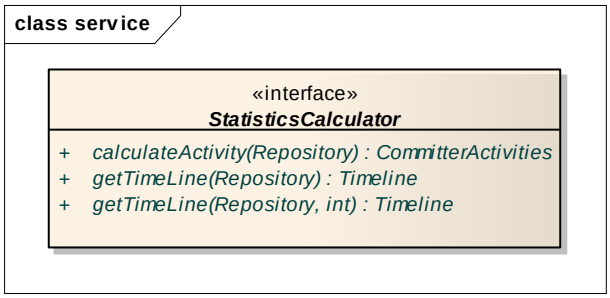


Service Layer

The service layer encapsulates the business logic of Giteway. Its components calculate statistics. The data is retrieved through the data access layer.

The API:

- Defines the timeline
- Calculates the summary of the user collaboration

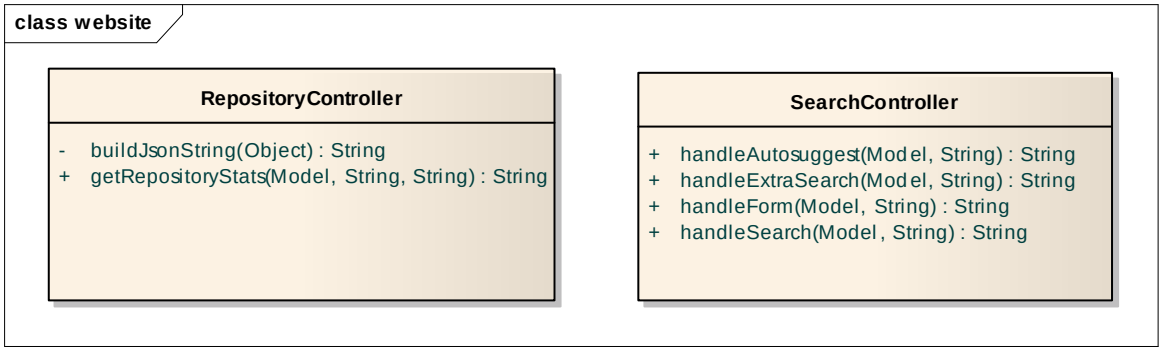


Presentation Tier

The presentation tier is divided in two parts: the website and the restful WS.

1. ***The Website***

Two Spring webmvc controllers handles the two views.



Two JSP views are accessible.

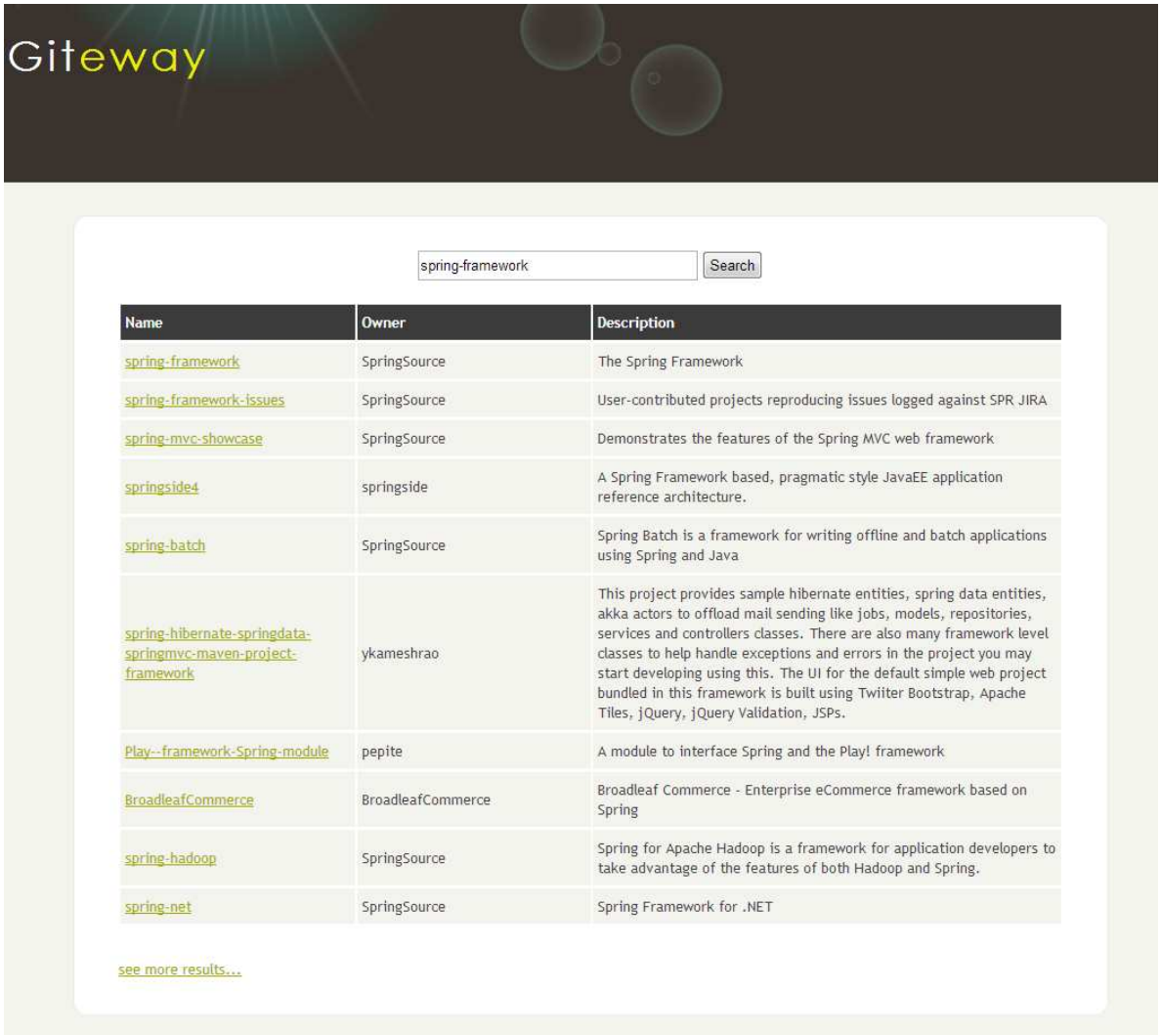


Figure 3 - The search view

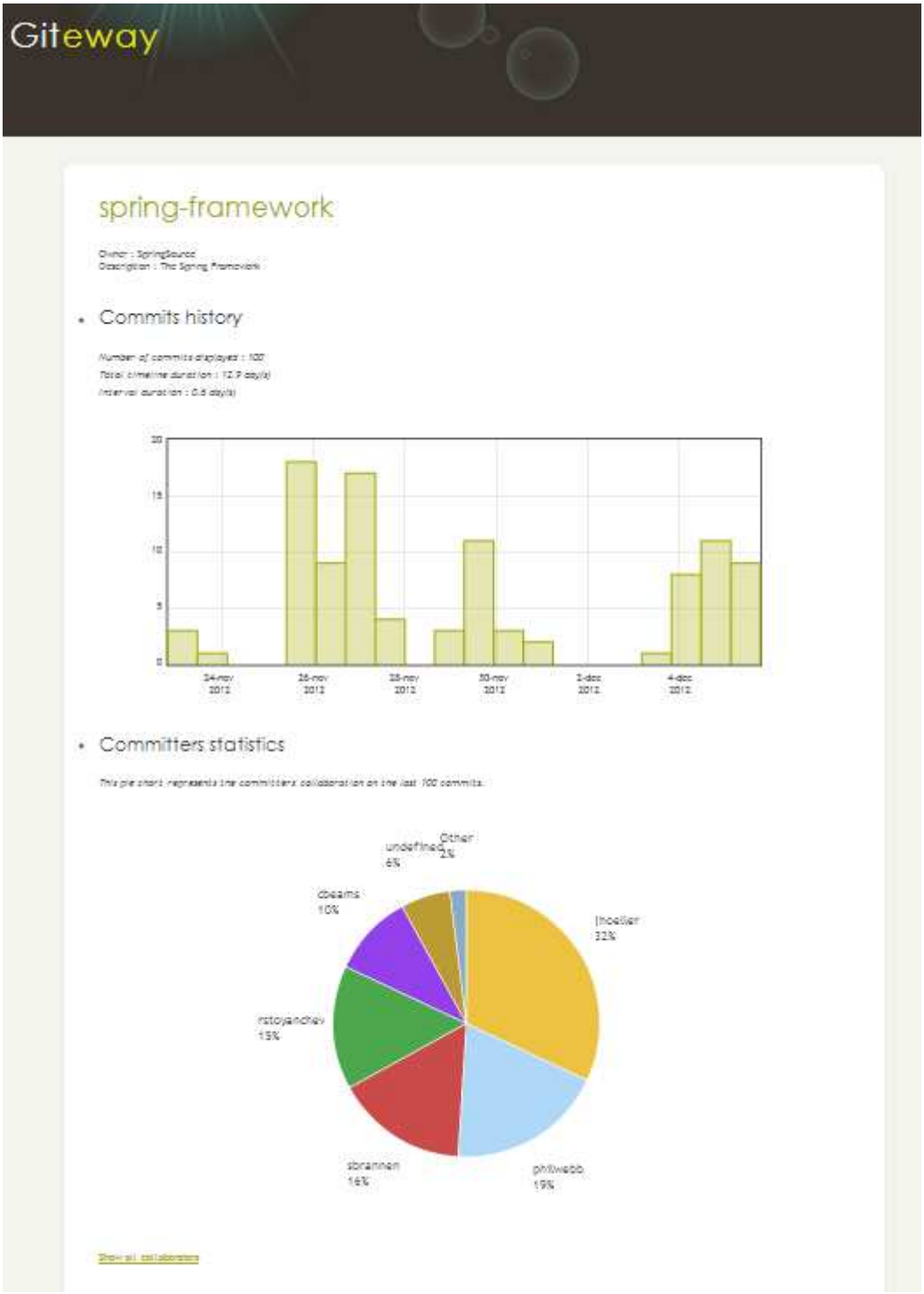


Figure 4 - The statistic view

The views are based on the following technologies/Framework:

JSP / JSTL

JQuery

JQueryUI (autosuggest)

Plot (charts)

Ajax (autosuggest and “search extra repositories”)

Note: The Website has been designed with a restful approach. Every page is bookmarkable as URLs define which information is displayed. No session or history is held on the server.

2. *The restful web service*

Giteway also exposes a restful WS which provides statistics.

Services:

1. The timeline data at : *<app>/restful/timeline/<owner>/<repositoryName>*
2. The committers’ impact at : *<app>/restful/timeline/<owner>/<repositoryName>*

The data can be returned in two different formats by specifying a different header value:

1. Json (Accept: application/json)
2. Xml (Accept: application/xml)

```
$ curl.exe -H "Accept: application/json" http://giteway.cloudfoundry.com/restful/timeline/guillaumebort/play-scala
{
  "repository" : {
    "name" : "play-scala",
    "owner" : "guillaumebort",
    "description" : "Scala module for playframework"
  },
  "timelineIntervals" : [ {
    "start" : 1289158160000,
    "end" : 1289834354650,
    "commitCount" : 2
  }, {
    "start" : 1289834354651,
    "end" : 1290510549301,
    "commitCount" : 3
  }, {
    "start" : 1290510549302,
    "end" : 1291186743952,
    "commitCount" : 2
  }, {
```

Figure 5 - Timeline json

```
$ curl.exe -H "Accept: application/xml" http://giteway.cloudfoundry.com/restful/timeline/guillaumebort/play-scala
<?xml version="1.0" encoding="UTF-8"?>
<timeline>
    <repository>
        <name>play-scala</name>
        <name>guillaumebort</name>
        <name>Scala module for playframework</name>
    </repository>
    <commitCount>100</commitCount>
    <timelineDays>156.52653936342594</timelineDays>
    <intervalDays>7.826326979166667</intervalDays>
    <timelineInterval>
        <start>1289158160000</start>
        <end>1289834354650</end>
        <commitCount>2</commitCount>
    </timelineInterval>
    <timelineInterval>
        <start>1289834354651</start>
        <end>1290510549301</end>
        <commitCount>3</commitCount>
    </timelineInterval>
    <timelineInterval>
        <start>1290510549302</start>
        <end>1291186743952</end>
        <commitCount>2</commitCount>
    </timelineInterval>
```

Figure 6 - timeline xml

Committer's impact :

```
$ curl.exe -H "Accept: application/json" http://giteway.cloudfoundry.com/restful/activity/guillaumebort/play-scala
{
  "repository" : {
    "name" : "play-scala",
    "owner" : "guillaumebort",
    "description" : "Scala module for playframework"
  },
  "committerActivityList" : [ {
    "committer" : {
      "login" : "sadache",
      "avatarUrl" : "https://secure.gravatar.com/avatar/d349588ba91256515f7e2aa315e8cfae?d=https://a248.e.akamai.ne
ges%2Fgravatars%2Fgravatar-user-420.png"
    },
    "percentage" : 59
  }, {
    "committer" : {
      "login" : "guillaumebort",
      "avatarUrl" : "https://secure.gravatar.com/avatar/adcd749d588278dbd255068c1d4b20d3?d=https://a248.e.akamai.ne
ges%2Fgravatars%2Fgravatar-user-420.png"
    },
    "percentage" : 39
  }, {
    "committer" : {
      "login" : "pk11",
      "avatarUrl" : "https://secure.gravatar.com/avatar/e98302802fb211bed73365b9ab809039?d=https://a248.e.akamai.ne
ges%2Fgravatars%2Fgravatar-user-420.png"
    },
    "percentage" : 1
  }, {
    "committer" : {
      "login" : "chrislewis",
      "avatarUrl" : "https://secure.gravatar.com/avatar/314fcdd97720b250c96a88e6fa4213a4?d=https://a248.e.akamai.ne
ges%2Fgravatars%2Fgravatar-user-420.png"
    },
    "percentage" : 1
  } ],
  "commitCount" : 100
}[D:\curl]
```

Figure 7 – Committers' impact json

```
$ curl.exe -H "Accept: application/xml" http://giteway.cloudfoundry.com/restful/activity/guillaumebort/play-scala
<?xml version="1.0" encoding="UTF-8"?>
<committer-activities>
    <repository>
        <name>play-scala</name>
        <name>guillaumebort</name>
        <name>Scala module for playframework</name>
    </repository>
    <commitCount>100</commitCount>
    <committerActivity>
        <user>
            <login>sadache</login>
            <avatarUrl>https://secure.gravatar.com/avatar/d349588ba91256515f7e2aa315e8cfae?d=https://a248.e.akamai
mages%2Fgravatars%2Fgravatar-user-420.png</avatarUrl>
        </user>
        <percentage>59</percentage>
    </committerActivity>
    <committerActivity>
        <user>
            <login>guillaumebort</login>
            <avatarUrl>https://secure.gravatar.com/avatar/adcd749d588278dbd255068c1d4b20d3?d=https://a248.e.akamai
mages%2Fgravatars%2Fgravatar-user-420.png</avatarUrl>
        </user>
        <percentage>39</percentage>
    </committerActivity>
    <committerActivity>
        <user>
            <login>pk11</login>
            <avatarUrl>https://secure.gravatar.com/avatar/e98302802fb211bed73365b9ab809039?d=https://a248.e.akamai
mages%2Fgravatars%2Fgravatar-user-420.png</avatarUrl>
        </user>
        <percentage>1</percentage>
    </committerActivity>
    <committerActivity>
        <user>
            <login>chrislewis</login>
            <avatarUrl>https://secure.gravatar.com/avatar/314fcdd97720b250c96a88e6fa4213a4?d=https://a248.e.akamai
mages%2Fgravatars%2Fgravatar-user-420.png</avatarUrl>
        </user>
        <percentage>1</percentage>
    </committerActivity>
</committer-activities>
```

Figure 8 - Committers' impact xml

Caching

Giteway accesses Github data remotely. This can imply high latency. To speed up things, a cache has been set up on the Daos methods. Spring provides an easy, non-intrusive way of handling caching without any coding. The spring-ehcache approach has been chosen.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ehcache xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="ehcache.xsd" updateCheck="true"
        monitoring="autodetect" dynamicConfig="true">

        <!-- Repository -->
        <cache name="searchRespositories" maxEntriesLocalHeap="100" eternal="false" timeToLiveSeconds="3600" me

        <!-- List of commits -->
        <cache name="commits" maxEntriesLocalHeap="100" eternal="false" timeToLiveSeconds="3600" memoryStoreEvi

        <!-- List of users -->
        <cache name="users" maxEntriesLocalHeap="100" eternal="false" timeToLiveSeconds="3600" memoryStoreEvict

        <!-- List of repositories -->
        <cache name="repositories" maxEntriesLocalHeap="100" eternal="false" timeToLiveSeconds="3600" memorySto

        <!-- List of String for suggestion -->
        <cache name="autosuggests" maxEntriesLocalHeap="250" eternal="false" timeToLiveSeconds="86400" memorySt


</ehcache>
```

Figure 9 - EhCache configuration

One of the drawbacks of caching is that if the data changes on GitHub, Giteway might not “see” it for the duration of the item life time. Therefore, the timeToLive attributes have been set to “small” values (1 hour to 1 day).

AOP

AOP handle the logging with an around advice weaved on all methods of the application except the front methods. Spring-AspectJ approach has been used.

Testing - Check-Style - Code coverage

Every tier of giteway is unit tested.

Figure 10 - Unit test analyse

The components have been isolated from each other by mocking the dependencies with mockito. Spring-test has been used for the tooling but no integration tests are performed.

The application code has been analyzed with Sonar.

Figure 11 - CheckStyle