

## **Giteway 1.0**

## **Technical presentation**

## Table of content

Deployment.....	3
Technology stack .....	3
Global Architecture .....	4
The model.....	5
Data Access Layer .....	7
Service Layer.....	8
Presentation Tier .....	9
Caching .....	12
AOP .....	13
Testing - Check-Style - Code coverage .....	13

## Deployment

Source code : [git://github.com/Couettos/giteway.git](https://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, JQueryUI*

## Global Architecture

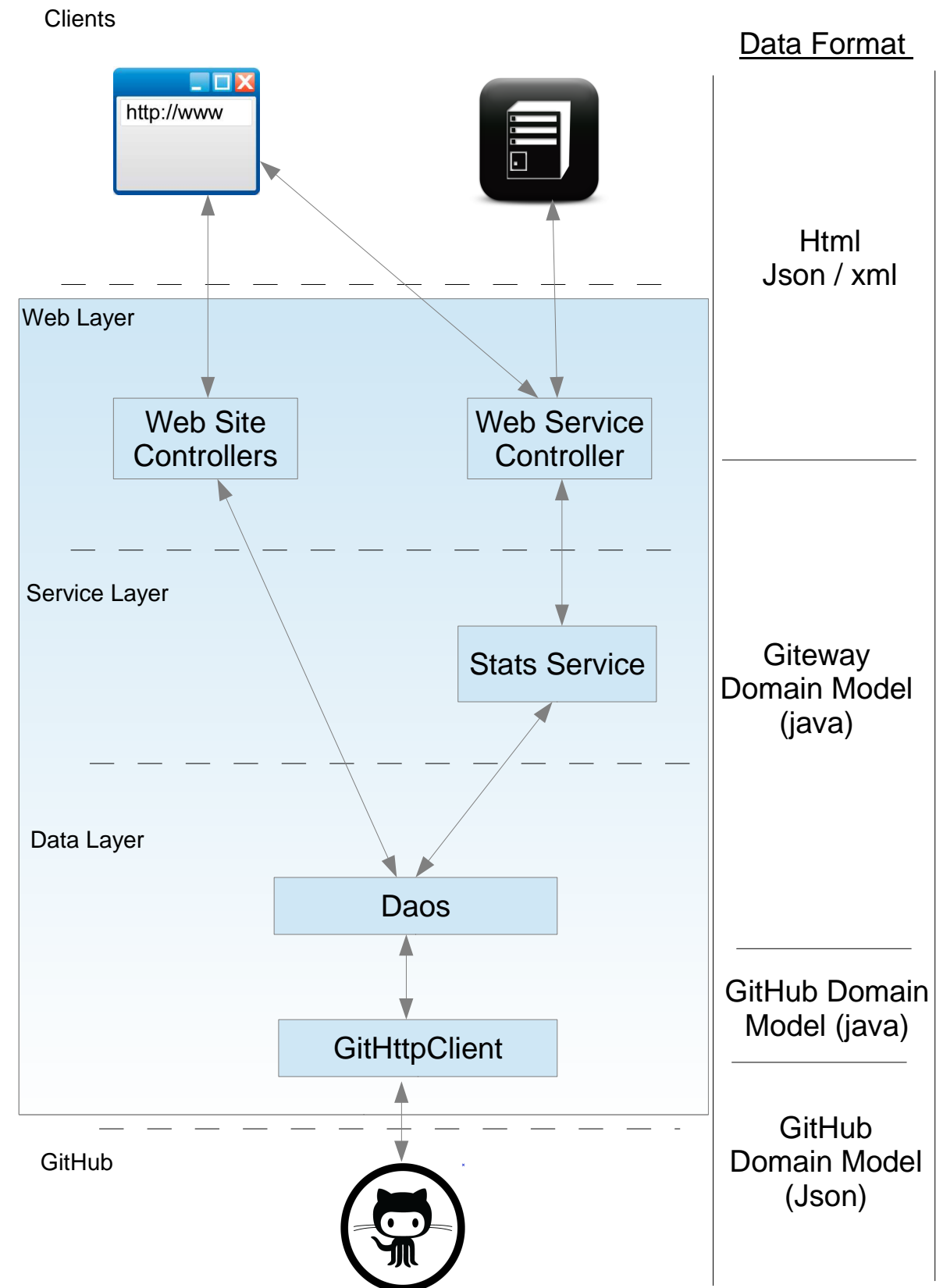


Figure 1 - Global architecture

## Model

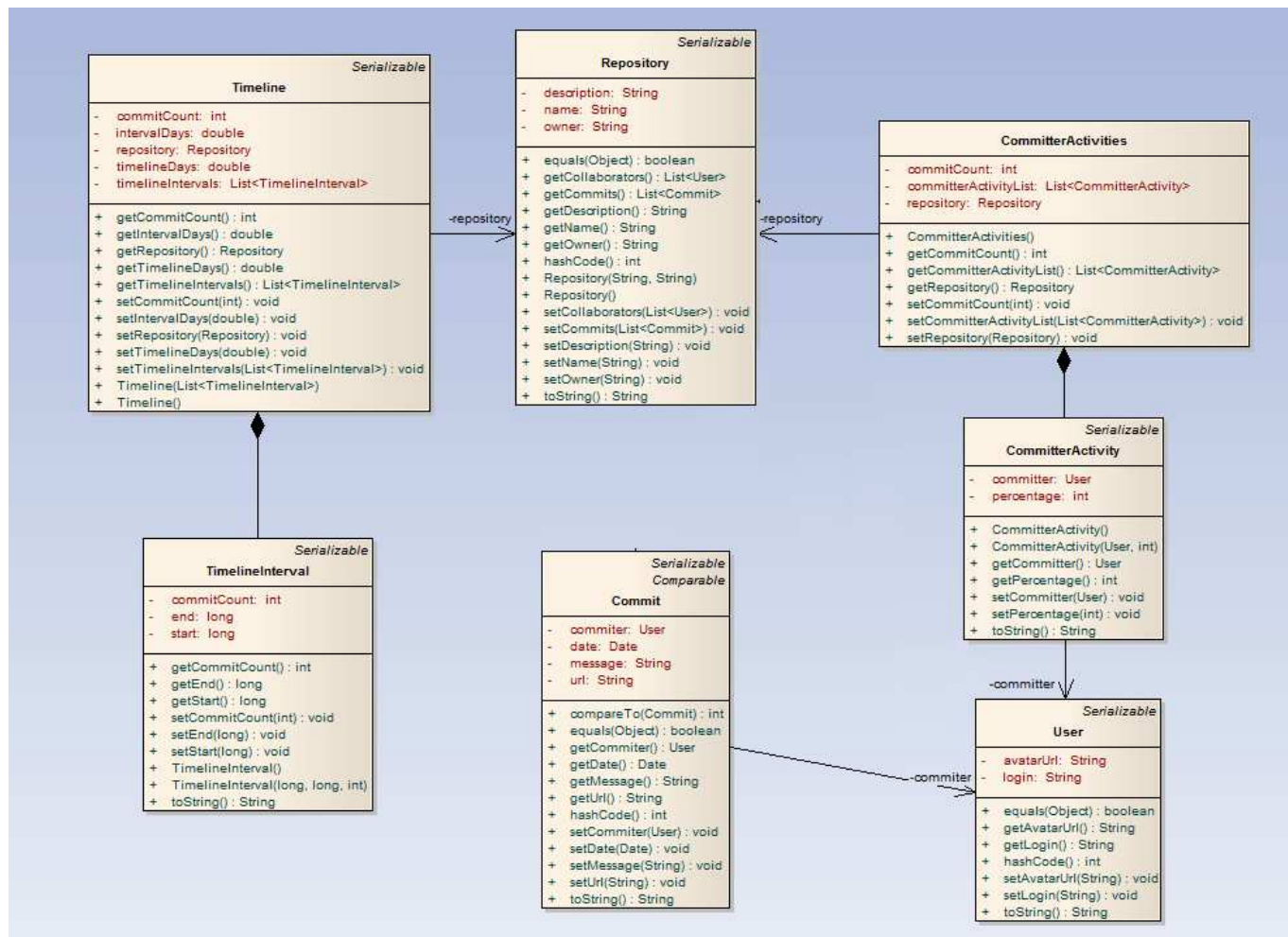


Figure 2 - The model

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.

GitHub Objects are considered as Value Objects or DTOs.

Advantages:

- Clearer architecture and object responsibilities
- Reduce the complexity of GitHub Domain Model
- Isolates the impacts of attribute modification on both DM.

Drawbacks:

- Glue code
- Transfer Object instantiation per request
- More code to write if a new attribute has to be mapped

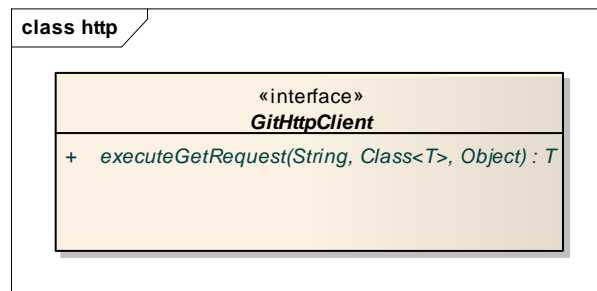
## Data Access Layer

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

The data access tier provides two mechanisms:

1<sup>st</sup>: Request Github over HTTP, map it to DTOs

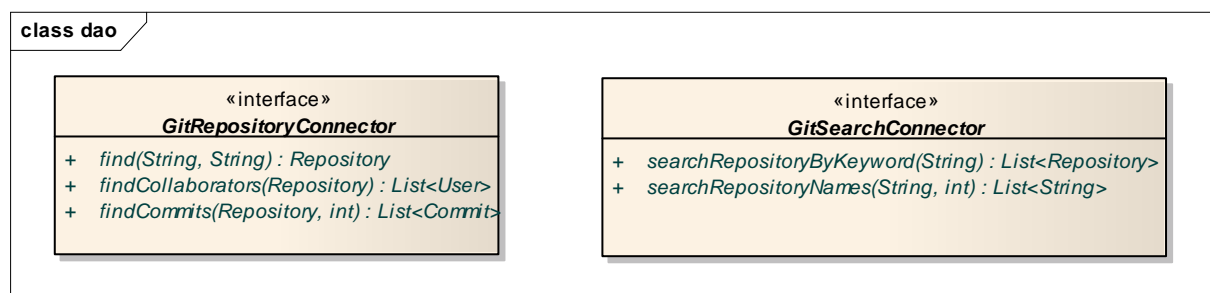
The data is accessible over HTTP in a json format. The interface `GitHttpClient` requests the data and converts it into a set of mapped class representing the GitHub DM (DTOs).



*Note : `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 two requests, the connections are kept alive. This is of great interest as the SSL handshake increases the latency.*

2<sup>nd</sup>: Map GitHub DM to Giteway DM

The DAOs receive DTOs from `GitHttpClient` and convert these into Giteway domain model.

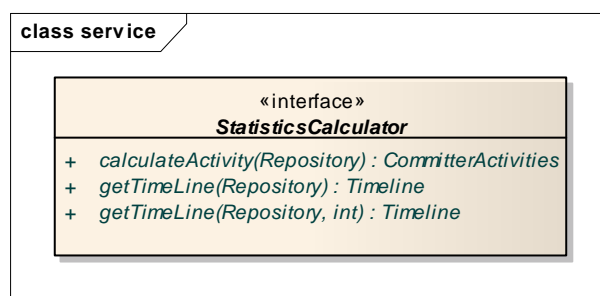


## Service Layer

The service layer encapsulates the business logic of Giteway. The component calculates statistics. The data is accessed through the data access layer.

The API:

- Defines the timeline
- Calculates the user's collaboration





## Presentation Layer

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

### 1. *Restful web service*

Giteway exposes a restful WS which provides statistics on GitHub repositories.

Services:

1. The timeline data at : `<app>/restful/repos/<owner>/<repositoryName>/timeline`
2. The committers' activity at : `<app>/restful/repos/<owner>/<repositoryName>/activity`

The data can be returned in two different formats depending on the "Accept" request header value:

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

```
$ 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>
</timeline>
```

Figure 3 - timeline xml

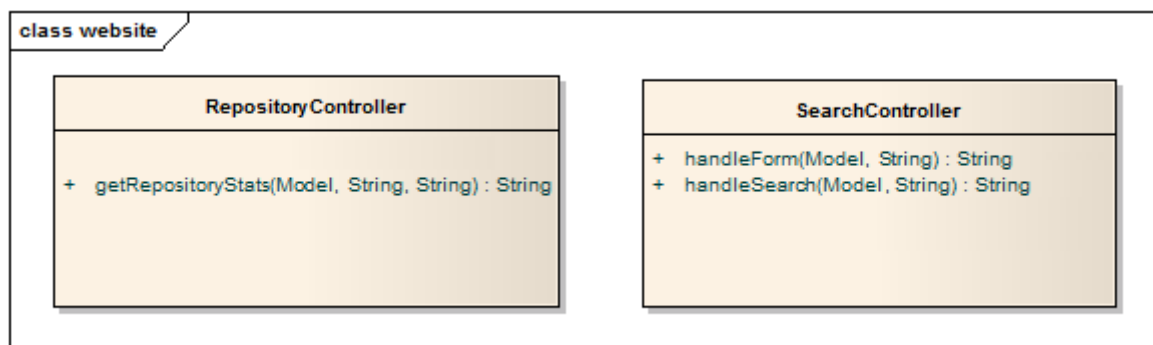
Committer's impact :

```
$ curl.exe -H "Accept: application/json" http://gitway.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.net/secure.gravatar.com/avatar-user-420.png"
    },
    "percentage" : 59
  }, {
    "committer" : {
      "login" : "guillaumebort",
      "avatarUrl" : "https://secure.gravatar.com/avatar/adcd749d588278dbd255068c1d4b20d3?d=https://a248.e.akamai.net/secure.gravatar.com/avatar-user-420.png"
    },
    "percentage" : 39
  }, {
    "committer" : {
      "login" : "pk11",
      "avatarUrl" : "https://secure.gravatar.com/avatar/e98302802fb211bed73365b9ab809039?d=https://a248.e.akamai.net/secure.gravatar.com/avatar-user-420.png"
    },
    "percentage" : 1
  }, {
    "committer" : {
      "login" : "chrislewis",
      "avatarUrl" : "https://secure.gravatar.com/avatar/314fcdd97720b250c96a88e6fa4213a4?d=https://a248.e.akamai.net/secure.gravatar.com/avatar-user-420.png"
    },
    "percentage" : 1
  } ],
  "commitCount" : 100
} {D:\curl}
```

Figure 4 – Committers' activity json

## 2. *The Website*

Two Spring webmvc controllers handles the two views.



The website is divided in two views.

Name	Owner	Description
<a href="#">spring-framework</a>	SpringSource	The Spring Framework
<a href="#">spring-framework-issues</a>	SpringSource	User-contributed projects reproducing issues logged against SPR JIRA
<a href="#">spring-mvc-showcase</a>	SpringSource	Demonstrates the features of the Spring MVC web framework
<a href="#">springside4</a>	springside	A Spring Framework based, pragmatic style JavaEE application reference architecture.
<a href="#">spring-batch</a>	SpringSource	Spring Batch is a framework for writing offline and batch applications using Spring and Java
<a href="#">spring-hibernate-springdata-springmvc-maven-project-framework</a>	ykaimeshiao	This project provides sample hibernate entities, spring data entities, akka actors to offload mail sending like jobs, models, repositories, services and controllers classes. There are also many framework level classes to help handle exceptions and errors in the project you may start developing using this. The UI for the default simple web project bundled in this framework is built using Twitter Bootstrap, Apache Tiles, jQuery, jQuery Validation, JSPs.
<a href="#">Play--framework-Spring-module</a>	pepite	A module to interface Spring and the Play! framework
<a href="#">BroadleafCommerce</a>	BroadleafCommerce	Broadleaf Commerce - Enterprise eCommerce framework based on Spring
<a href="#">spring-hadoop</a>	SpringSource	Spring for Apache Hadoop is a framework for application developers to take advantage of the features of both Hadoop and Spring.
<a href="#">spring-net</a>	SpringSource	Spring Framework for .NET

[see more results...](#)

Figure 5 - The search view

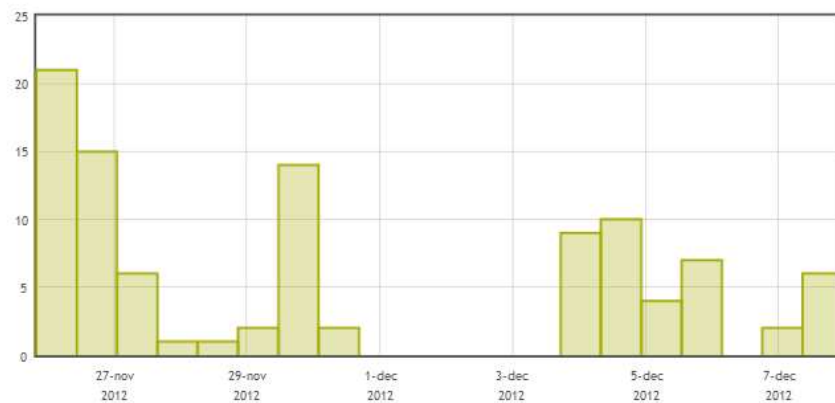
## spring-framework

Owner : SpringSource  
Description : The Spring Framework

Timeline Committers' activity Show collaborators

### Timeline

Number of commits displayed : 100  
Total timeline duration : 12.1 day(s)  
Interval duration : 0.6 day(s)



**Figure 6 - The statistic view (timeline)**

The statistics loading mechanism is based on Ajax and the restful WS. This approach enables a nice, flexible and extendable mechanism.

*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.*

## Caching

Giteway accesses Github data remotely. Because the communication can be slow, a cache has been set up on the DAOs method returns. Spring provides an easy, non-intrusive way of handling caching without any coding. The spring-ehcache approach has been chosen.

## 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 7 - 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 8 - CheckStyle