TECHNISCHE
UNIVERSITÄT
DRESDEN

**Department of Computer Science** Institute for Systems Architecture, Chair of Computer Networks

Master Thesis

# GRAPHICAL DISCUSSION SYSTEM

Kaijun Chen
Born on: 18th September 1990 in China
Matriculation number: 3942792
Matriculation year: 2013

to achieve the academic degree

Master of Science (M.Sc.)

Supervisor
Tenshi Hara
Iris Braun

Supervising professor
Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill

Submitted on: 18th February 2015

TECHNISCHE
UNIVERSITÄT
DRESDEN

**Department of Computer Science** Institute for Systems Architecture, Chair of Computer Networks

## Statement of authorship

I hereby certify that I have authored this Master Thesis entitled *Graphical Discussion System* independently and without undue assistance from third parties. No other than the resources and references indicated in this thesis have been used. I have marked both literal and accordingly adopted quotations as such. They were no additional persons involved in the spiritual preparation of the present thesis. I am aware that violations of this declaration may lead to subsequent withdrawal of the degree.

Dresden, 18th February 2015

Kaijun Chen

## ABSTRACT

A discussion of the teaching content or the educational material is always essencial for both tutors and students in the teaching activities. In traditional way, a discussion can only be performed normally after courses also requires the absence of the students as well as the tutors.

The traditional approach of discussing shows its limitations. Inefficiency in knowledge acquisition: not all the students have the same question and the tutor is able to provide explanation for only one question at same time; time-consumption: ; low interactivity:

Thus, a discuss system with intense interactivity as well as in crowdsourcing way is highly needed. To achieve high interactivity, a discuss system with graphical tool and real-time data communication is proposed. Students are able to contribute their questions and answers to get to the bottom of his deficiencies of teaching content and the educational material. And students who has the same questions can instantly acquire the best solution which is recommended and approved by the community.

In order to validate and evaluate the concepts of this approach, an implementation of the proposed solution is developed on top of modern web technologies. Moreover, a usability questionnaire survey is proposed and delivered for a quantized evaluation of the client application. The performance of this application is also evaluated at the same time through the created simulation scenarios.

# CONTENTS

# 1 INTRODUCTION

With the rapid development and popularization of internet and technology, the traditional educational activities are moving to the online platforms. MooCs like ..... have taken the responsibilities and ... to .

Tell some histories!

## 1.1 MOTIVATION

What's the situation now?

Pain?

## 1.2 GOALS AND RESEARCH QUESTIONS

What's the features of the new system?

What's the problems/question am i solving?

## 1.3 THESIS OUTLINE

Outline

# 2 BACKGROUND AND RELATED WORKS

## 2.1 ONLINE Q.A. SYSTEMS

## 2.2 MOOCS

# 3 STATE OF THE ART

The following chapter gives an overview of state of the art. To achieve the high interactivity and responsiveness in the graphical discussion system, a lot of modern web technologies should be applied.

However, there are always plenty of alternatives for each technology which could differ from system to system. So it is important to investigate and analyse the existing solutions and capture an overview about different alternatives of technologies. It's is also vital to understand the benifit and drawback of the technologies used. In addition, a

First of all, the general modern web technology and development workflow will be introduced, which goes through our whole development and has a great impact on the development efficiency. The next part describes the different graphical technologies on web, and which fits our system best. Then an overview of the real time communication technologies is illustrated and evaluated. At last, a collection of modern technologies applied within the backdend server is listed.

## 3.1 MODERN WEB DEVELOPMENT

## 3.2 SEPERATION OF FRONTEND AND BACKEND

### 3.2.1 EVOLUTION

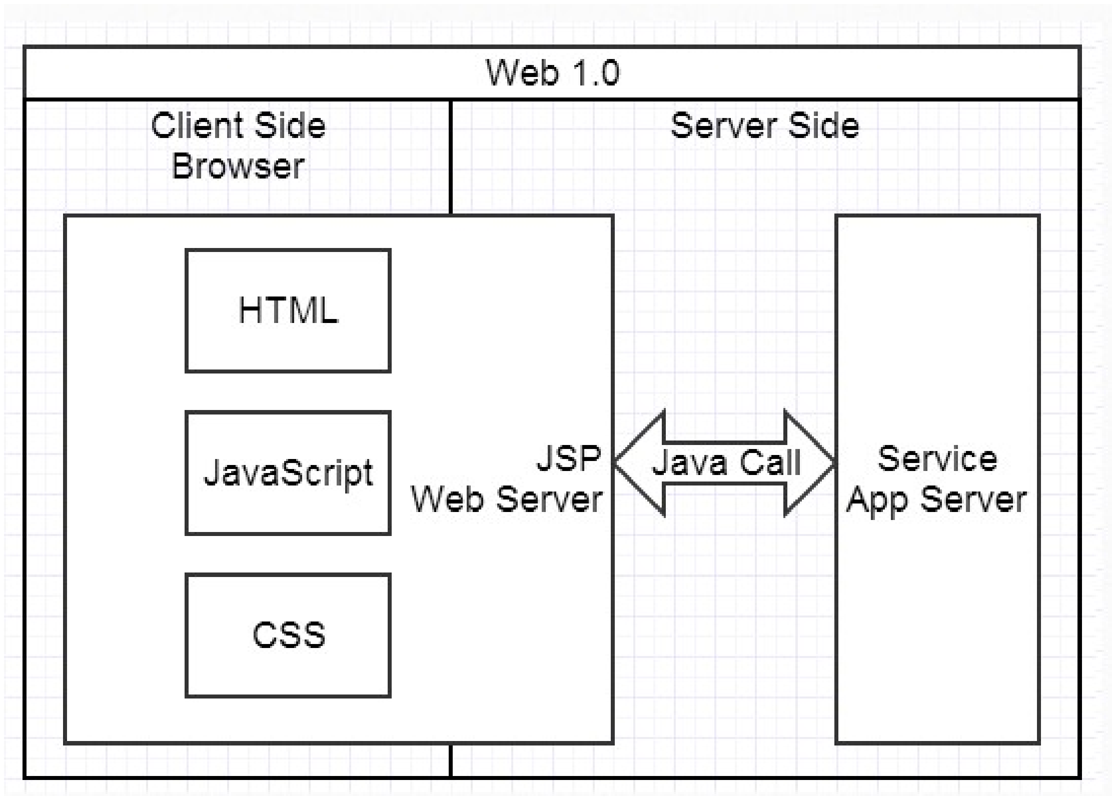
With the increasing complexity of a web app and high demand of agile development, people are always thinking about how to improve patterns of the workflow in web development as well as the architecture of a web app. To achieve better scalability, maintainability and ubiquity of a web app, the architecture of an entire web app has been envolving in the past several years .

#### EARLY AGE

At the early age of web development, the backend did all jobs for both client side(browser) and server side, such as rendering, calling system service, composing data, etc. Figure 3.1 shows the overview of web architecture: The advantage of this pattern is clear: The development and deployment are simple and straightforward, if the bussiness logic doesn't become complexer. But with the increasing complexity of the product, the problems shows up:

1. Development of frontend heavily depends on the whore development environment. Developers have to start up all the tools and services for testing and debugging only

Figure 3.1: Web architecture in early age



some small changes on view. In most cases, the frontend developer who isn't familar with the backend needs help while intergrating the new views into the system. Not only the efficiency of development, but also the cost of communication between frontend developer and backend developer are huge problems.

2. The own responsibility of front-end and back-end mess up, which could be expressed by the commingled codes from different layers, for example there is no clear boundry from data processing tp data representing. The maintainability of the project becomes worse and worse with the increasing complexity.

It's really significant to improve the maintainability of code, as well as the efficiency and resrationality of division of work from both front-end and back-end in the whole web development phase. In the section below, a evolution of the technical architecture will reveal how these problems are solved.

#### WEB 2.0

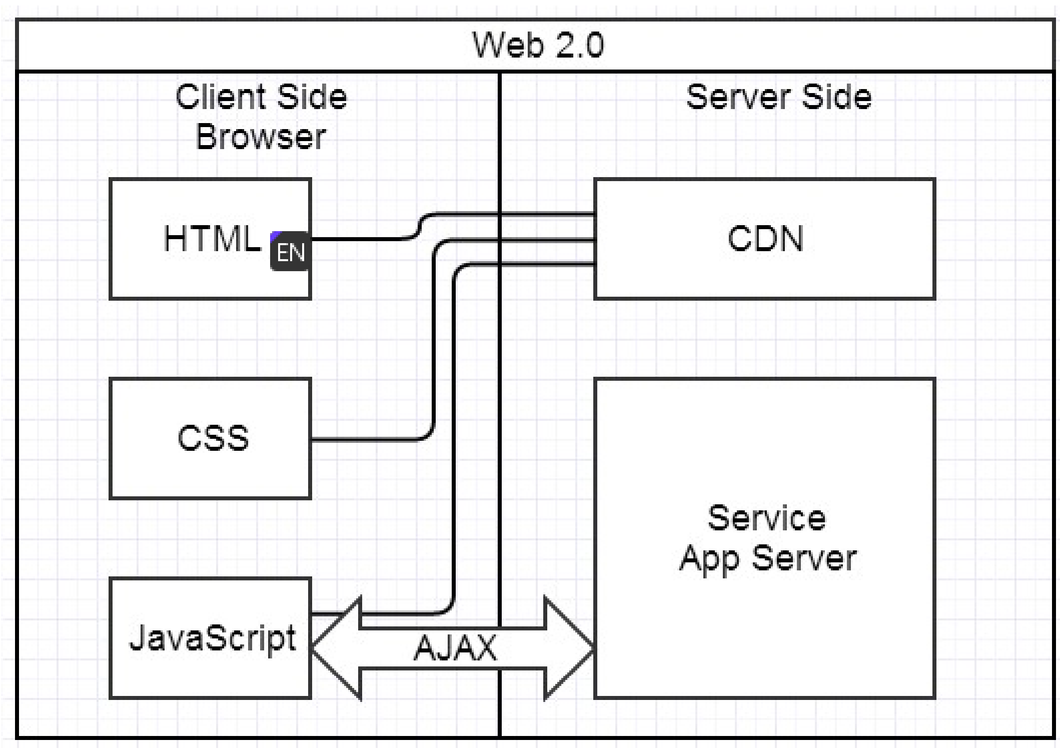
Along with the birth of Gmail[1] in 2004, which is noted for its pioneering use of Ajax, the web application started to behave more interactively. Browser began to take over the job of data fetching, processing, rendering, such a sequence of workflow which could only be done by the server side formerly.

The architecture in Web 2.0 generation is presented in figure 3.2.

By using Ajax, the client has the ability to fetching data stream asynchronously, after which the client will consume the data and render it into the specific section of view.

[1]https://mail.google.com/

Figure 3.2: Web 2.0 architecture



Usability was dramatically improved, because the entired view represented to users will not be refreshed and the front-end is able to process and render data in its own intension, which means more flexible control of the consumption of data.

### 3.2.2 SINGLE PAGE APP

With the evolution of web technologies and promotion of these technologies in modern browswers by brower vendors, a new web development model called Single-page application was proposed and caught the developers' eye. The back-end is no more responsible for rendering and view controling, it only take charge of providing services for the front-end.

The structure in figure 3.3 shows that the client side has the full control of view rendering and data consumption after data acquisition through web services which is released by back-end with promised protocol. All rendering tasks was stripped off from the server side, which means that the server side achieves more efficiency and concentrate more on the core bussiness logics.

But more responsibility in front-end means more complexity. How to reduce the complexity and increace the maintainability of a front-end project becomes a significant problem. Developers come up with an new envolved variant of SPA as demonstrated in figure 3.4.

In general, the architecture is componentized and layered into template, controller and model. Each component is isolated and has its own view as well as correlated logics. Front-end frameworks like EmberJS, AngularJS, ReactJS are providing such a approach and development pattern for developers to build modern web apps. With this approach, a giant and complex front-end app is broken up into fine grained components, therefore, components are easy to reuse if the components are well abstrated in a proper way. In addition, the maintaince of each component is also effortless.

Figure 3.3: SPA architecture

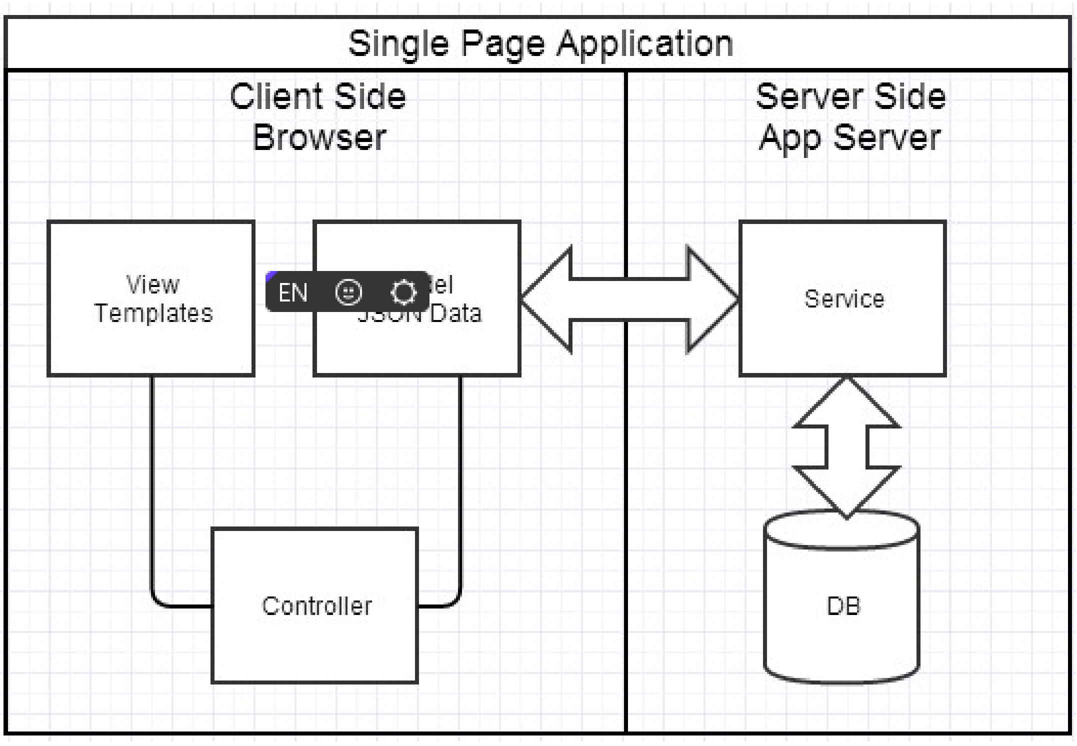
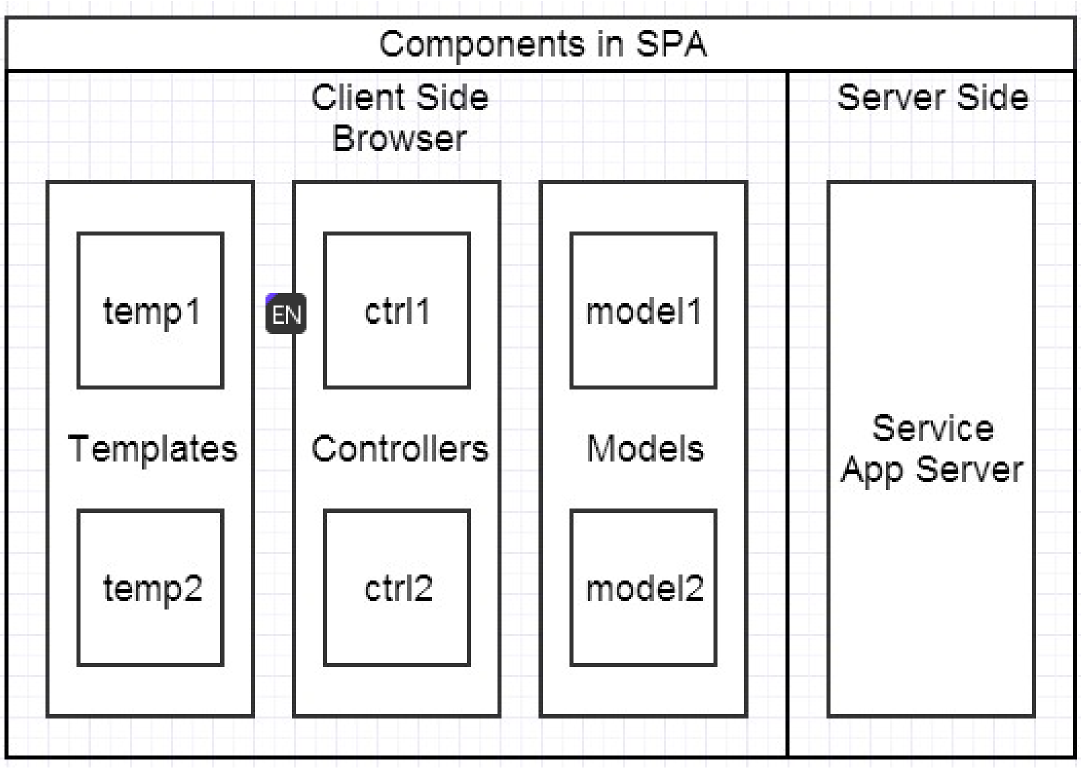


Figure 3.4: Components in SPA



In summarize, a single-page app has a lot of benefits:

1. Rational seperation of works from front-end and back-end: client takes charge of view rendering and data representation, as well as slight data processing if needed; the server focus on providing services of the core logics, persistance of data, and also computational tasks.
2. High interactivity and user experience in client side: asynchronous data fetching and view rendering implies no more need of hard reloading the page which user is viewing and the current states of the page could also be preserved.
3. Efficiency in server side: rendering tasks are stripped off from server side.
4. Ubiquity: with the seperation of services provided by server side, not only the web browser but also other clients in other platforms such as Android, iOS apps are able to access and comsume the services.

But SPA also has its deficiencies:

1. Search engine optimization unfriendly: because the page are not directly rendered by server side, and the web crawlers are not able to run JavaScript codes like a browser does, the site could not be crawled properly under normal circumstances. So if SEO results really matter for the app, SPA is obviously not the best choice.
2. Excessive http connections: all the data is acquired from different services through diversed Application programming interfaces, thus multiple HTTP connections are established and performed parallelly, whose initial time of connections for partial data could be much more than a single connection in the traditional way. So it's highly needed to merge the services and find a balance between data model complexity and time consumption.

## 3.3 RESTFUL INTERFACE

## 3.4 CONTINUOUS INTEGRATION

## 3.5 OPERATING-SYSTEM-LEVEL VIRTUALIZATION

## 3.6 GRAPHICS ON THE WEB

## 3.7 REAL-TIME COMMUNICATION

## 3.8 EFFICIENT CLIENT SIDE

## 3.9 EFFICIENT SERVER SIDE

# 4 AIMS AND OBJECTIVES

# 5 CONCEPTION

## 5.1 RESTFUL API DEFINITIONS

## 5.2 DATA MODEL

## 5.3 GRAPHICAL DATA CONVERTION

## 5.4 REAL-TIME DEMAND

# 6 IMPLEMENTATION

## 6.1 ARCHITECTURE

## 6.2 SERVER

## 6.3 CLIENT

## 6.4 DIFFICULTIES AND OPEN QUESTIONS

# 7 EVALUATION

## 7.1 USABILITY

## 7.2 SYSTEM OVERLOAD

# 8 CONCLUSION AND FUTURE WORK

## 8.1 CONCLUSION

## 8.2 FUTURE WORK

# LIST OF FIGURES

# LIST OF TABLES