

Copyright
by
Marcos Suguru Kajita
2009

**The Report Committee for Marcos Suguru Kajita Certifies that this is the approved
version of the following report:**

**Google App Engine Case Study:
A Micro Blogging Site**

Committee:

Adnan Aziz, Supervisor

Sarfraz Khurshid

**Google App Engine Case Study:
A Micro Blogging Site**

by

**Marcos Suguru Kajita
BSCS**

Report

Presented to the Faculty of the Graduate School of
The University of Texas at Austin
in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science in Engineering

**The University of Texas at Austin
December, 2009**

Google App Engine Case Study: Micro Blogging Site

Marcos Suguru Kajita, MSE
The University of Texas at Austin, 2009

Supervisor: Adnan Aziz

Cloud computing refers to the combination of large scale hardware resources at datacenters integrated by system software that provides services, commonly known as Software-as-a-Service (SaaS), over the Internet. As a result of more affordable datacenters, cloud computing is slowly making its way into the mainstream business arena and has the potential to revolutionize the IT industry. As more cloud computing solutions become available, it is expected that there will be a shift to what is sometimes referred to as the Web Operating System. The Web Operating System, along with the sense of infinite computing resources on the “cloud” has the potential to bring new challenges in software engineering. The motivation of this report, which is divided into two parts, is to understand these challenges. The first part gives a brief introduction and analysis of cloud computing. The second part focuses on Google’s cloud computing platform and evaluates the implementation of a micro blogging site using Google’s App Engine.

Table of Contents

List of Tables	vii
List of Figures	viii
Introduction.....	1
Cloud Computing.....	4
Impact on Business	4
Cloud Computing Challenges	6
Choosing the Right Cloud Computing Platform	7
Google App Engine.....	8
Java and Sandbox	8
GAE Plug-in for Eclipse	9
Google Web Toolkit	12
Datastore	13
Micro Blogging Site on Google App Engine.....	14
Motivations	14
Basic Approach	15
MyMicroBlog Architecture	17
MicroBlog	18
IBlogView	21
Create Blog View	23
Edit Blog View	24
Image List Viewer and Uploader	25
BlogEntry and ImageEntry Classes	26
BlogItem and ImageItem Classes	26
IBlogEntryService and IBlogServiceAsynch Interfaces	27
BlogEntryServiceImpl Class	27
Formatting and Style	29
Image Location Tagging.....	30

Other Utility Tools	32
Selenium IDE.....	32
Work Breakdown	34
Results	36
Development Experience	36
Performance	37
Lessons Learnt	40
Future Work	41
Conclusion	43
Bibliography	44
Vita	46

List of Tables

Table 1:	List of Views in MyMicroBlog	21
Table 2:	Work breakdown of main development tasks.....	34
Table 3:	Total lines of code in the client and server side.....	36
Table 4:	Project deployment time information	36
Table 5:	System description and development environment	37
Table 6:	First vs. Subsequent page load time.....	38
Table 7:	Page load time measurement	38

List of Figures

Figure 1:	The Computing Platform Lifecycle	2
Figure 2:	The Cloud Computing Arena.....	5
Figure 3:	GAE menu buttons on Eclipse.....	10
Figure 4:	Eclipse Package Explorer	10
Figure 5:	GAE Web Application Wizard	11
Figure 6:	GAE Deployment Tool on Eclipse	11
Figure 7:	MyMicroBlog main page using GWT Dark Theme	16
Figure 8:	Model-View-Controller Diagram	17
Figure 9:	MyMicroBlog Diagram	18
Figure 10:	Google Sign-in Screen	19
Figure 11:	MicroBlog Class Diagram	20
Figure 12:	Partial Class Diagram of View Classes in MyMicroBlog	22
Figure 13:	Layout of CreateBlogView using GWT Panels	23
Figure 14:	Create Blog view.....	24
Figure 15:	Edit Blog view	24
Figure 16:	Image List Viewer and Image Uploader	25
Figure 17:	Image Uploader expanded	25
Figure 18:	BlogItem Entity on Google App Engine Datastore	26
Figure 19:	BlogEntry RPC Implementation	28
Figure 20:	BlogEntry Servlet Configuration	28
Figure 21:	The MyMicroBlog.gwt.xml file.....	29
Figure 22:	Custom style settings for GWT widgets	30
Figure 23:	Image viewer with lat-long values	31

Figure 24:	Image pin on map.....	31
Figure 25:	Selenium IDE for Firefox	33
Figure 26:	Script used to measure page load time.....	39
Figure 27:	YSlow statistics for MyMicroBlog main page	40
Figure 28:	YSlow statistics for a commercial blogging site.....	40

Introduction

Cloud computing received its name from the representation of the Internet as a cloud on blackboard diagrams. Cloud computing refers to the hardware, systems software, and datacenters that provide services over the Internet [1]. Cloud computing is not a new concept. In fact, it is in use in many places, but on a smaller scale. For example, researchers at university campuses perform complex computations on biochemical models and weather pattern predictions much faster on grid computers [2]. Users play the same online game from different geographical locations. Documents are created and shared by many users through Google Docs.

These are examples of cloud computing concepts in use in smaller scale. Recently, cloud computing has made into the main stream business arena due to the commoditization of computers. Many companies have started to offer their own cloud computing solutions. Companies like Amazon, Google, Microsoft, IBM, and Sun have released innovative cloud computing solutions. These solutions vary from virtualized hardware to well designed APIs [4]. In a few months, users will have an increased number of options when more companies release their own cloud computing solutions. The business of providing cloud computing solutions is growing very fast because businesses taking advantage of such offerings are growing as well [10].

As users move to the cloud, it is expected that there will a migration to the Web Operating System (Web OS). Cloud computing is setting the stage for a revolution in the IT industry. There is a possibility that desktop applications will be replaced by web applications and companies will use cloud computing solutions for their IT needs [10]. At the moment, cloud computing is at its infancy. As any other platform development cycle (Figure 1), there will be an adjustment phase followed by the standardization of cloud

computing technologies [9]. During the initial phase, it is important that users evaluate and gain expertise on the cloud technologies because as in any other development cycle there will be a few technologies more successful than others. It is important that users become familiar with the different technologies in order to choose the prevailing technology when the industry stabilizes. Until the cloud computing industry stabilizes, the user needs to evaluate and choose the cloud technology that best fits his needs for the short term. For the long term, the user needs to compare and contrast different technologies. These experiments should not be a problem since the cloud technologies are cheap. The affordability of cloud solutions makes the technology even more attractive to users [10].

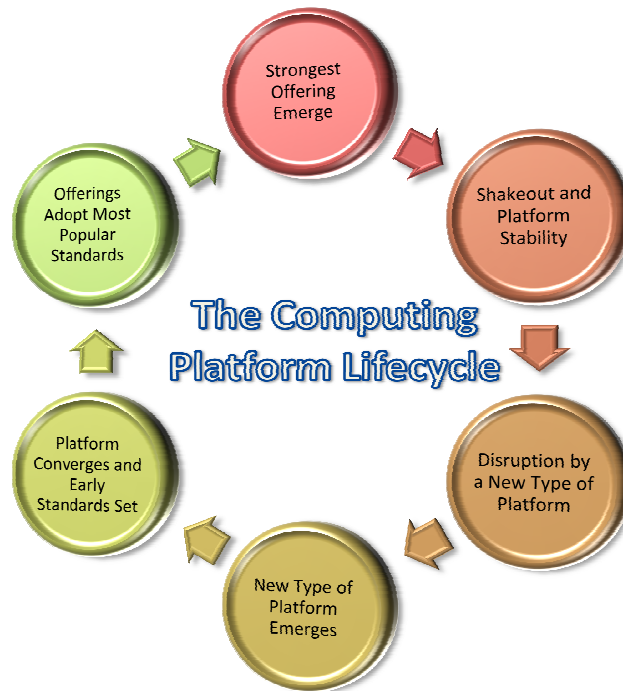


Figure 1: The Computing Platform Lifecycle

The Google App Engine (GAE) platform, which is evaluated in the second part of the paper, is free up to a point. With affordable solutions like GAE, there is no reason for users to stay away from the cloud technologies. In fact, there are plenty of reasons to stay up to date with the technology because the cloud enables creativity to drive the development of products and businesses. The sudden competition from fast-followers can cause a major impact on businesses in general [9]. Therefore, careful evaluation of technologies like GAE helps keep one eye on the technology and the other eye on the competition.

Cloud Computing

IMPACT ON BUSINESS

Cloud computing and the promise of the Web OS is stirring the IT Industry. Businesses that traditionally had very little presence in IT have become major players on this new business endeavor. Amazon is a good example of such company. From its humble beginnings as an online store, Amazon has become one of the leading cloud computing providers [10]. Many other companies are using their strengths to create successful cloud computing offerings. Some companies seem to have an advantage over the competitions since they built their company's DNA around the web. One example of such company is Google. Google has harnessed its experience and presence on the web to provide enterprise solutions on cloud computing. Traditionally, Google has not been involved in enterprise solutions but with its release of GAE it has plans to expand its businesses beyond online advertisements [10]. Many cloud providers are offering new cloud solutions and the market does not seem to be limited to large companies only. The same commoditization of computers that has initially boosted the cloud business has also opened the doors for smaller companies who want to provide their unique solutions [1]. The cloud has become a large centrifugal force that keeps attracting companies and businesses.

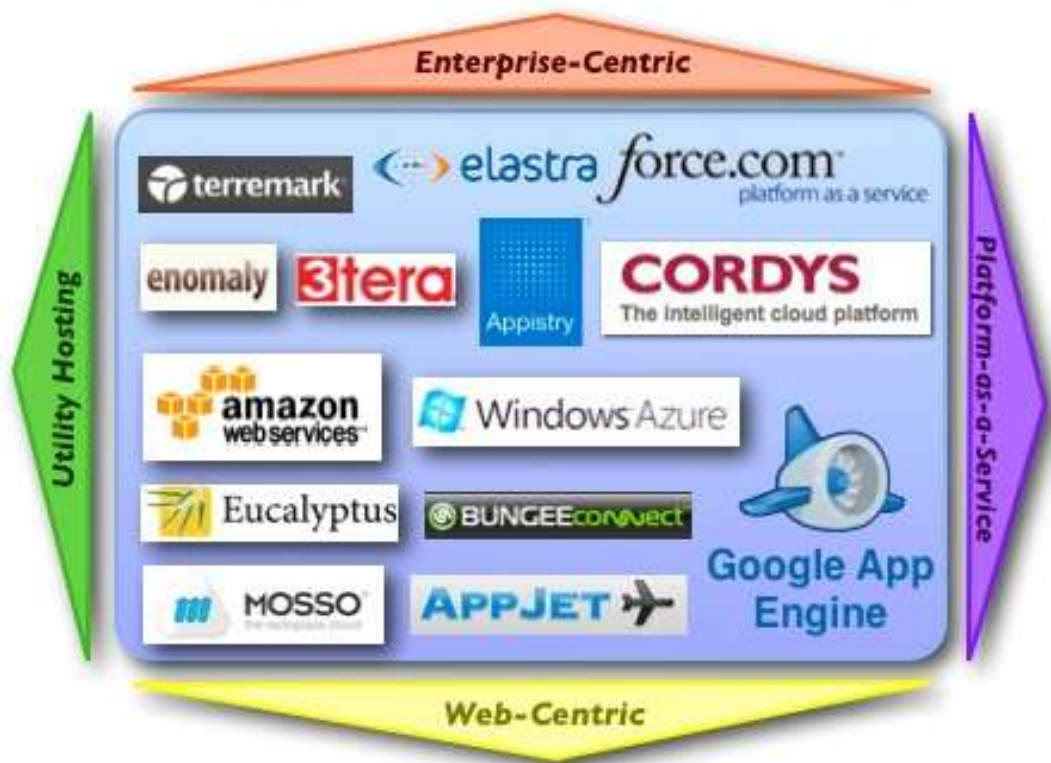


Figure 2: The cloud computing arena [10]

For users, the bottom line is that cloud computing is affordable. Amazon sells its service for only 10 cents per hour and its Scalable Storage Service (S3) for \$0.12 per gigabyte-month. The majority of today's cloud computing clientele are medium and small businesses (i.e., startups). For these companies, affordable services like S3 provide a golden opportunity to launch new products with significantly less capital upfront [10]. The economics around cloud computing allow innovative companies to create products that were economically challenging or are significantly cheaper than the competition [9]. Traditionally, many business ideas took time to realize because of the prohibitive cost of computing power, scalability issues, and primarily the high risk due to large capital. This has all changed and innovation is now the key.

CLOUD COMPUTING CHALLENGES

Many of the challenges in cloud computing are connected to security. The idea of moving the entire company's private data to some remote data center controlled by a third party company may seem like exposing business secrets. It will take time to adjust to this new model and security will be one of the biggest challenges to cloud providers [10]. Imagine that a company has put all its data on Amazon's S3. Despite the fact that Amazon has built its reputation on reliability, there is still the risk of the service going offline for a few hours. If that happens, the user's data is no longer available and he is at the mercy of Amazon's ability to recover [10]. However, with heavier investments on systems software and hardware, Amazon's services should still be more resilient to system failures and secure than any in-house IT solution.

There is also the concern that "once you are in, you are really in" [5]. For many startups, the decision whether to adopt a cloud computing platform is simpler. However, for more established companies, the worry about lock-in is a real concern. Some companies are concerned that if they opt for a platform, they may lose their investment if the cloud provider goes under. There is always the risk of lock-ins if a technology is used before fully understanding it. For example, naïve implementation on Google's App Engine may result in a suite of applications that may be challenging to port to another environment. If for some reason Google decides to increase their fees and the cost no longer makes economical sense for the client to stay with Google, the migration back to the client company's environment or another provider may turn out to be difficult and costly. The challenge is to understand the strengths and limitations of each cloud computing solution to make intelligent long term decisions.

CHOOSING THE RIGHT CLOUD COMPUTING PLATFORM

As mentioned above, there are several cloud computing solutions in the market today. The choices range from virtualized hardware to all-in-one API packages. The key to model computation, storage, communication, and the illusion of infinite resources is the level of abstraction these different solutions provide. Each solution seems to focus on different aspects of abstraction [1]. For example, Amazon's Elastic Compute Cloud (EC2) provides very low level abstraction to the hardware. EC2 virtualizes the hardware and offers a set of libraries that allow developers full control over the software stack. In contrast, Google's App Engine (GAE) provides a framework based on well defined interfaces and APIs that allow quick development with quality. The development on GAE is done on either Java or Python. There is support for a few other languages but they are not as significant as for these two. For example, Google's data storage service supports Java Data Object (JDO), which completely abstracts the implementation of the database.

At the moment, it is difficult to determine the right cloud computing platform. It is up to the user to determine whether he needs the level of hardware control provided by Amazon's EC2 or the easy migration of Java code to the App Engine environment [9]. Hopefully, as the platform development cycle comes to a full circle, there will be enough standards that allow users to develop on one platform and store data on another. At that point, the right choice will be the platforms that support the standards and allow users to migrate seamlessly from one provider to another.

Google App Engine

Google App Engine (GAE) is the platform that allows software developers to leverage Google's cloud computing infrastructure for web application development [3]. Similar computing resources (or services) used by Google Docs are available under the GAE. Google offers the same reliability and openness as its other flagship products like Google Search and Google Mail. GAE is free up to a point, but there is plenty of room for the experiments that are needed to evaluate the technology. The following are some of GAE's key features [3].

- Dynamic web serving, with full support for common web technologies
- Persistent storage with queries, sorting, and transactions
- Automatic scaling and load balancing
- APIs for authenticating users and sending email using Google Accounts
- A fully featured local development environment that simulates Google App Engine on your computer
- Task queues for performing work outside of the scope of a web request
- Scheduled tasks for triggering events at specified times and regular intervals

JAVA AND SANDBOX

GAE released the support for Java only recently. Initially, it only supported Python, but since most of web enterprises have standardized to either .NET or Java, it was a matter of time to have Java support in GAE. The Java run time environment in GAE is feature rich and offers comprehensive support of Java's standard libraries and APIs. All the development can be done in native Java and the application interacts with GAE via Java Servlet Interfaces. The Java environment provides Java 6 JVM, JavaMail,

JCache, and standard interfaces for GAE's datastore with Java Data Object (JDO) and Java Persistence API (JPA). Java standards make the development on GAE easy and familiar.

There are some limitations due to sandboxing, so migrating existing code to GAE requires some minor changes. The JVM runs on a sandbox to provide security to applications. The sandbox assures that each application on GAE do not interfere with each other. It also guarantees performance and scalability.

GAE PLUG-IN FOR ECLIPSE

Eclipse is an open source community with the objective of building a standard development platform. The Eclipse Integrated Development Environment (IDE) provides Java as its primary programming language, but it supports plug-ins for many other languages such as C, C++, etc. Development on GAE is straight forward with the GAE plug-in for Eclipse IDE. Eclipse manages the workspace and makes the development on GAE as simple as developing any other Servlet-based web application. The Eclipse plug-in allows developers to create new GAE application from within Eclipse IDE. Provided that the user has a GAE account, the user can create, deploy, and run web applications directly from the IDE. The App Engine SDK provides a web server for testing applications locally in a simulated environment. The plug-in adds extra buttons to the Eclipse menu bar (Figure 3) that allows the applications to run on the local web server in hosted mode.

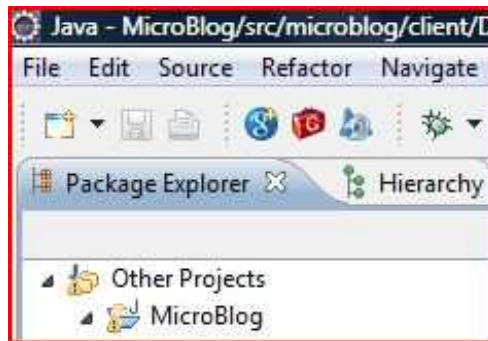


Figure 3: GAE Menu Buttons on Eclipse

The management of client and server source files is also done from within the IDE (Figure 4). Eclipse's development tools and debugging facilities are available for developing web applications. GAE application development on Eclipse is no different than writing a Servlet or desktop application. Once the application is ready, the deployment is as easy as a button click.

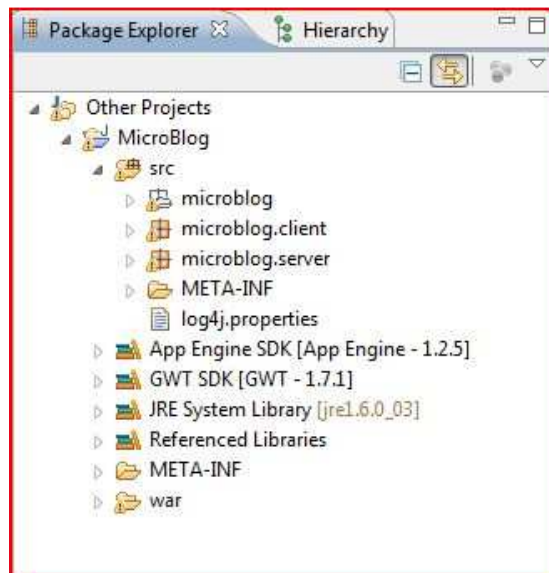


Figure 4: Eclipse Package Explorer

When creating a new web application, the wizard (Figure 5) takes care of the creation of the necessary configuration files needed to run applications on GAE. The web.xml and the main HTML files are generated automatically by the GAE plug-in.

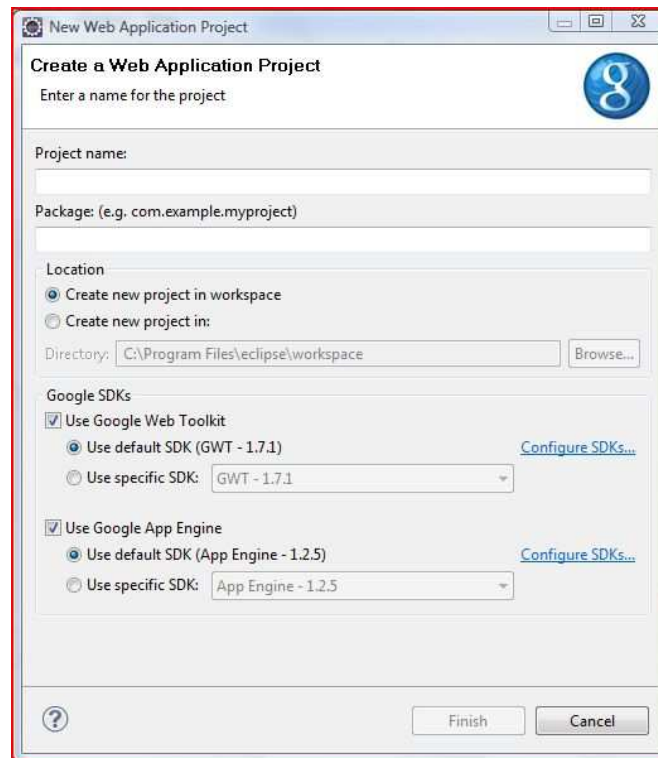


Figure 5: GAE Web Application Wizard

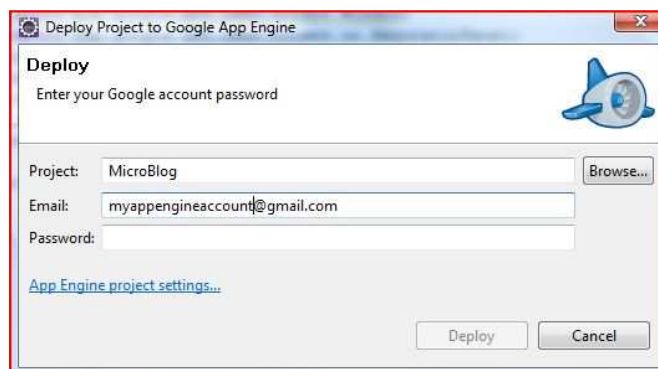


Figure 6: GAE Deployment Tool on Eclipse

GOOGLE WEB TOOLKIT

Developing web applications can be tedious and time consuming. It is fair to say that software developers spend most of their time trying to figure out little quirks in JavaScript or compatibility issues on certain browsers. Asynchronous JavaScript plus XML (AJAX) is usually a popular option for writing interactive and attractive web applications. AJAX is a web development technique based on standards such as JavaScript, XML, HTML, and CSS. AJAX allows client web applications to communicate with the server asynchronously in the background allowing the development of interactive web applications. However, writing scalable, reusable, and maintainable code in AJAX can be challenging.

Google Web Toolkit (GWT) abstracts the complexity of JavaScript by providing a rich set of Java tools to develop attractive front end applications. Developers use GWT to design the layout of web applications and the display of Graphical User Interface (GUI) elements in pure Java. GWT's interfaces and libraries mimic those of existing User Interface (UI) frameworks such as Swing and Standard Widget Tool (SWT). The main difference is that GWT dynamically generates HTML instead of pixel based graphical elements. For example, GWT's Button results in HTML's '<button>' instead of an image or pixel based element. As a result, there are virtually no cross-browser incompatibility issues.

Behind the scenes, GWT compiles the Java code into optimized JavaScript that runs on most of commercial browsers. Events, such as button clicks, are coded directly to the GUI element in Java so it follows object oriented programming. GWT code running on Eclipse in hosted mode enables programmers to use debugging tools allowing them to step into the code line by line. Writing interactive web applications with attractive GUI elements in GWT is very similar to writing desktop applications.

DATASTORE

GAE makes data persistency on Google's distributed data storage facilities very simple. GAE abstracts the administrative work required to maintain a distributed system and database such as Bigtable [6]. Bigtable is Google's solution for high availability distributed storage system. The system manages petabytes of structured data across multiple scalable commodity servers. As resources are needed more machines can be added to the clusters without interrupting the system. Depending on the type of web application, the user could be interacting with several servers at any given time. With GAE, the programmer does not worry about any of the intricacies needed to maintain and synchronize data in the datastore. The engine takes care of all the transactions, load distributions, and even backups. The access to GAE datastore is provided via standard APIs, which result in a portable system. In other words, web applications built on these interfaces can be ported to and from Google environment effortlessly. This is an important characteristic of Google's datastore because the resulting system has no risk of lock-in with Google's BigTable. The engine supports the Java Data Object (JDO) and Java Persistence API (JPA). Both standards are from the DataNucleus Access Platform, which is an open source platform of many Java standards [7].

GAE also supports a SQL like query language called GQL. Because the datastore is not a traditional relational database, GQL provides a layer of abstraction with query notations similar to SQL. GQL simply provides familiarity to programmers accustomed with SQL statements on relational databases. Since GQL is a proprietary abstraction of Google's Bigtable, it is not recommended to design the system on it because of the risk of coupling and lock-in. The resulting system would not allow the user to port to a non-Google environment easily.

Micro Blogging Site on Google App Engine

MOTIVATIONS

The main motivation for implementing the micro blogging site is to evaluate GAE's services and technologies. A standard micro blogging site has the following features:

- User registration and secure login
- Client-server model
- Dynamic updates and data persistence
- Events and notifications
- Email communication

The goal is to implement the features listed above using the tools and APIs provided by GAE development platform. This effort will provide an insight on the level of usability, scalability, portability, and flexibility in developing web applications on the GAE's platform.

The secondary motivation for developing the application is to design components for a micro blogging framework. The goal is to develop a set of modular components and building blocks for developing micro blogging sites. The use of micro blogs has made into corporate life where the tool helps create communities within the company environment [8]. The messaging and notification functionality of blogging tools help the exchange of information among colleagues at work. The exchange of messages does not need to be between people. Systems can be integrated to the blogging system so that it can notify users of events. For example, a blogging system could be used by a software engineering team to monitor the development efforts. When new code is checked into the Version Control System (VCS), a text message is sent to the software architect or developer about certain events (e.g., build or compilation errors). Depending on the type

of business the requirements for a micro blogging application can vary and commercial products may not support them out of the box. Hence, there is a need for a platform for quick development of customized features [8].

BASIC APPROACH

The MyMicroBlog micro blogging site (Figure 7) is implemented in GWT and Java on GAE using the Eclipse IDE. The Web Application Wizard, which is a tool provided by GAE plug-in for Eclipse, helped create all the preliminary client and server code, configuration files, and application settings (Figure 5). Once the application was created, the deployment to Google's cloud environment was a two button click process using the Web Application Deployment tool (Figure 6). Once the deployment completed, the site was running on <http://mymicroblog.appspot.com/>. The entire process of creating and deploying the web application did not take more than a few seconds. The client and server folder structure shown in Figure 5 is created by the wizard and the environment is ready for development of the remaining pieces that makes the micro blogging site.

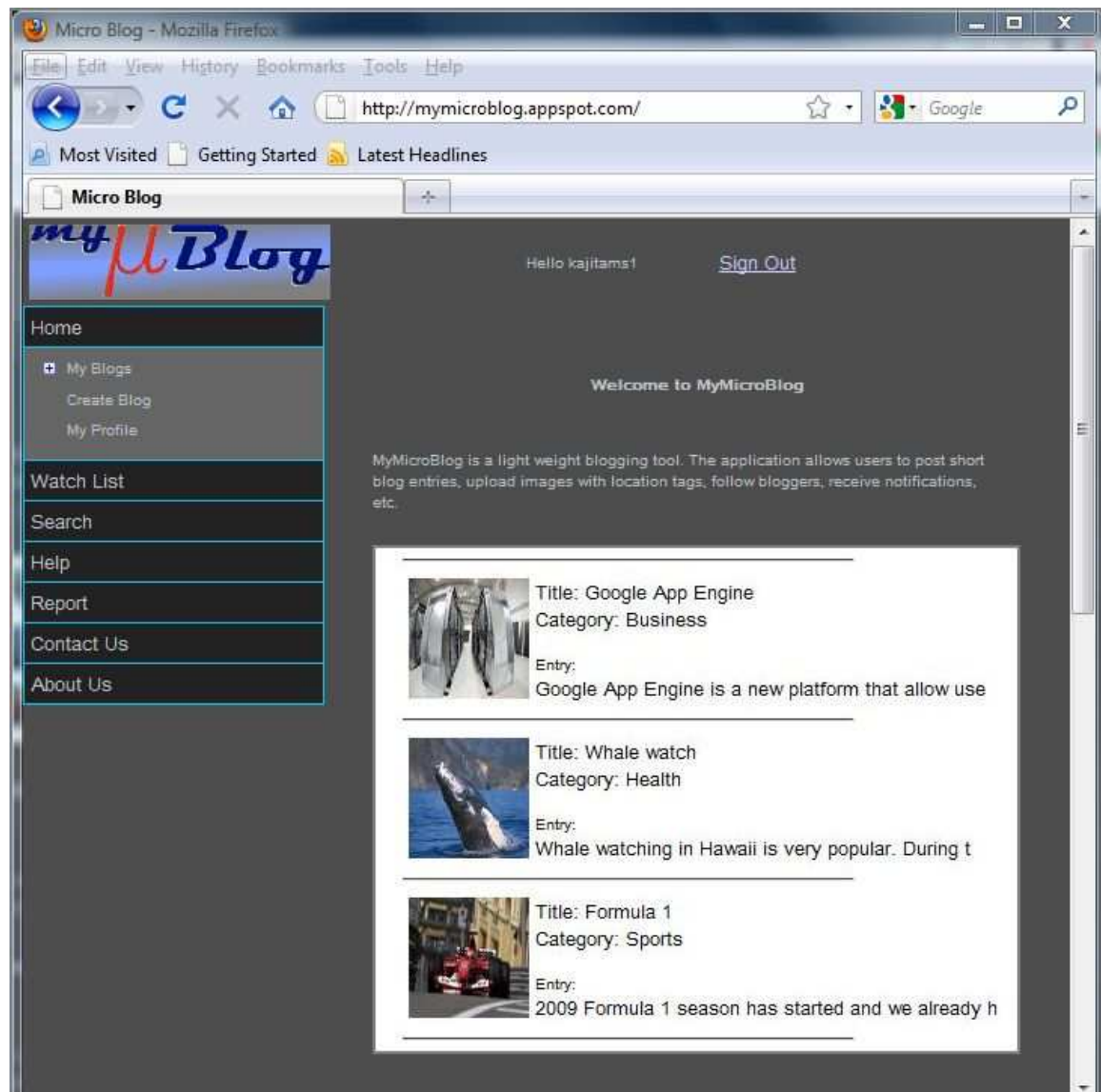


Figure 7: MyMicroBlog main page on GWT Dark Theme

MICROBLOG ARCHITECTURE

The basic architecture of MyMicroBlog is based on the Model View Controller (MVC) architectural pattern (Figure 8). The pattern has the benefit of isolating the business logic from the presentation and data layers. The *model* encompasses the object or data structure used by the application. Any manipulation to the domain's data is dealt within the *model* and notification is sent to the views. The *view* presents the *model* (i.e., data) to the user. Typically, the *view* has only the implementation of the GUI elements that visualizes the state of the *model*. The *controller* is the object that receives inputs (usually user inputs) and notifies the *model* or the *view*. The resulting application using the MVC pattern is modular and easy to test. Modular applications have the benefit of reusability, extensibility, and rapid development.

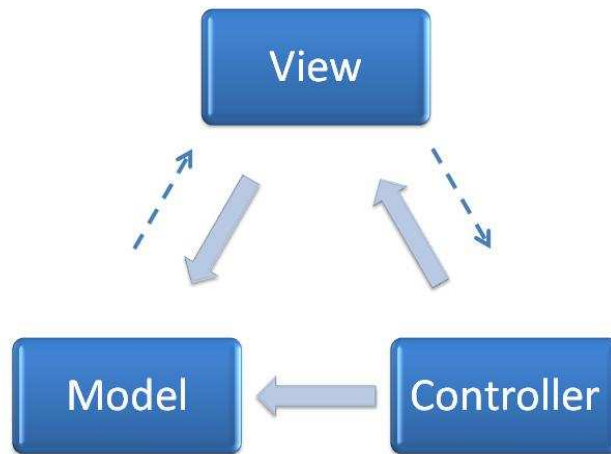


Figure 8: Model-View-Controller Diagram

The implementation of applications using MVC pattern usually uses the Observer design pattern. In the observer pattern, the object, called the subject, manages a list of “observers” and notifies them when an event occurs. In MVC, the controller is usually implemented using the observer pattern.

The MyMicroBlog does not use the observer pattern, but implements a controller class in the MVC pattern. The following diagram depicts the high level architecture of MyMicroBlog.

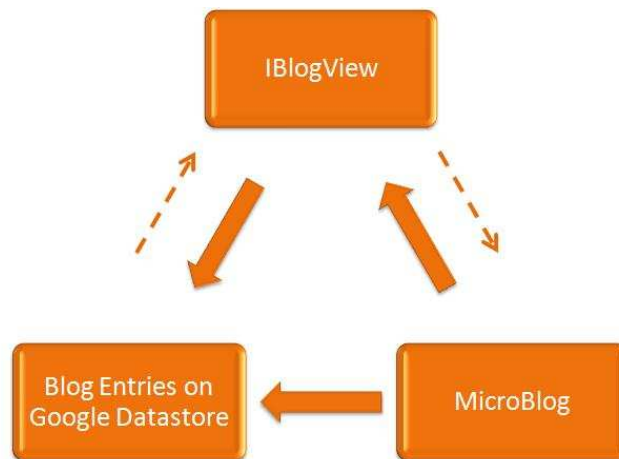


Figure 9: MyMicroBlog Diagram

MICROBLOG CLASS

The MicroBlog class is the main entry point for the application. It implements the `EntryPoint` interface from `com.google.gwt.core.client` package. The `'onModuleLoad'` method from the `EntryPoint` interface is called when the application first starts. The user validation is performed in the `onModuleLoad` method. The user authentication is done via GAE's `UserServieFactory`, which is a service that authenticates users with Google accounts. If the user is validated successfully, the main MyMicroBlog page is loaded. If the user is not validated, the user is directed to the Google account sign-in page (Figure 10). The class is also the controller on the MVC pattern. The class handles the messages from the views and generates notifications. Figure 12 shows the MicroBlog class diagram.



Figure 10: Google Sign-in Screen

The main navigation bar for the application is also in the MicroBlog class. The navigation bar receives the user input and the MicroBlog class loads the appropriate views or re-routes the notifications to the model.

The MicroBlog class has unnecessary complexity. In order to improve the object, the controller logic can be replaced by an observer pattern. The navigation bar also adds complexity and makes the MicroBlog class less flexible. If the navigation bar logic is moved into its own class, it will allow for better customization and reuse. The result would be a lightweight MicroBlog class that serves as the entry point for the application and delegates the core functionality to the MicroBlog MVC architecture.

One important point to make about the class is the fact that it uses the EntryPoint interface. The use of EntryPoint can impose a problem for portability since the interface is part of the GWT package. However, after the two improvements mentioned above, the class should be very light weight and porting the application to a non-Google environment would require very little refactoring. In addition to that, GWT is open source and is available under the Apache 2.0 license.



Figure 11: MicroBlog Class Diagram

IBLOGVIEW

IBlogView implements the interface that allows the different views in the MyMicroBlog application to be displayed. The interface implements the following methods.

- void Initialize(MicroBlog)
- Widget GetWidget()
- void Reset()

The view that implements the IBlogView interface can be plugged into the MicroBlog main page seamlessly. Each of the view classes renders the GUI interface and handles the user inputs. The following table shows the different views implemented in MyMicroBlog.

View Name	Description
AboutUsView	Displays the AboutUs page
CreateBlogView	Displays the create blog page with controls to allow new blog entries
DisplayBlogListView	Displays a page with a list of blogs
DisplayBlogView	Displays the blog entry
DisplayImageView	Displays the page for image loading and display
EditBlogView	Displays the page to edit blog entries
HelpView	Displays the Help
ReportAbuseView	Displays the report abuse page
ReportProblemsView	Displays the report problems page
SearchBloggerView	Displays the search page for bloggers
SearchByCategoryView	Displays the search page for categories
SendEmailView	Displays the email communication page
UserProfileView	Displays the user profile page

Table 1: List of Views in MyMicroBlog

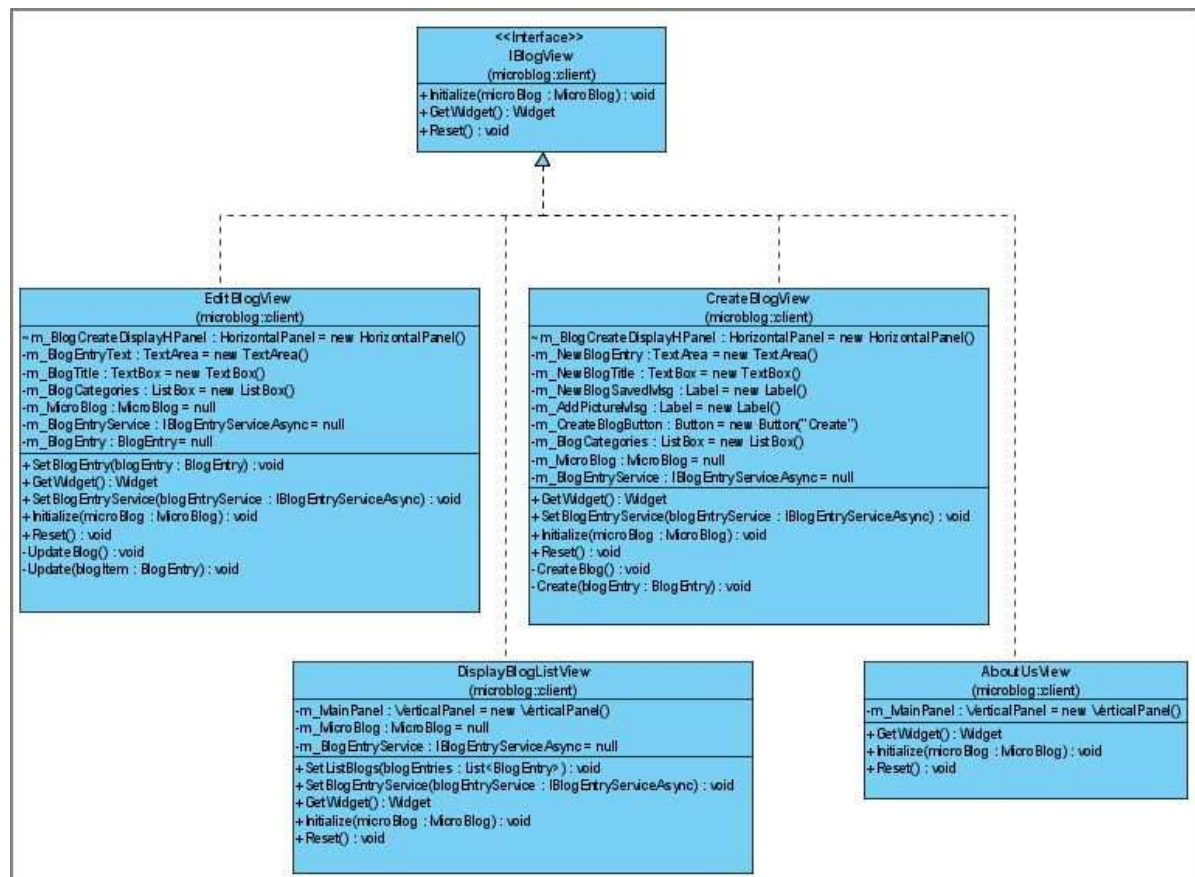


Figure 12: Partial Class Diagram of View Classes in MyMicroBlog

Each view class renders the GUI interface for a particular function in the system. For example, the `EditBlogView` class creates the controls needed to edit the blog entry and handles the user commands (i.e., Update or Cancel). The commands (events) are sent to the `MicroBlog` class, which in turn re-routes it to the `BlogEntry` object, which is an object in the *model* component on MVC.

The micro blog view classes use GWT to create the GUI elements. The layout of the controls is done via the GWT panels. The concept of paneling allows developers to take a page layout and translate it into code. The alignment of controls within a page is all done by specialized GWT panels. Having an upfront idea of the layout helps the selection

of the panels. Figure 13 shows the different panels used in the layout of the CreateBlogView.

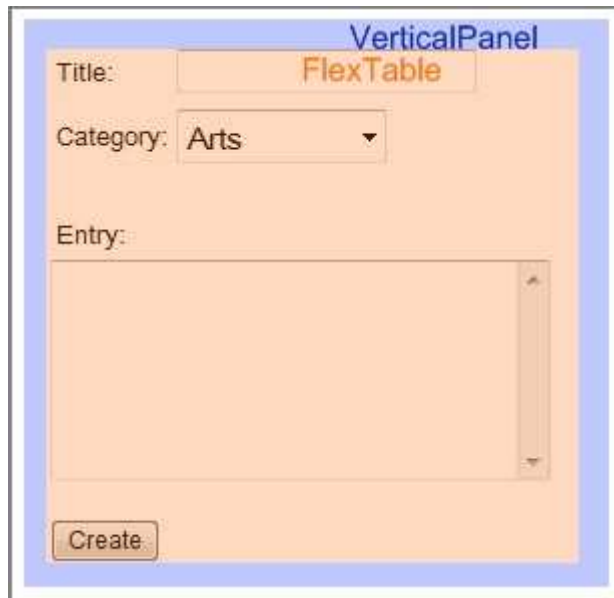
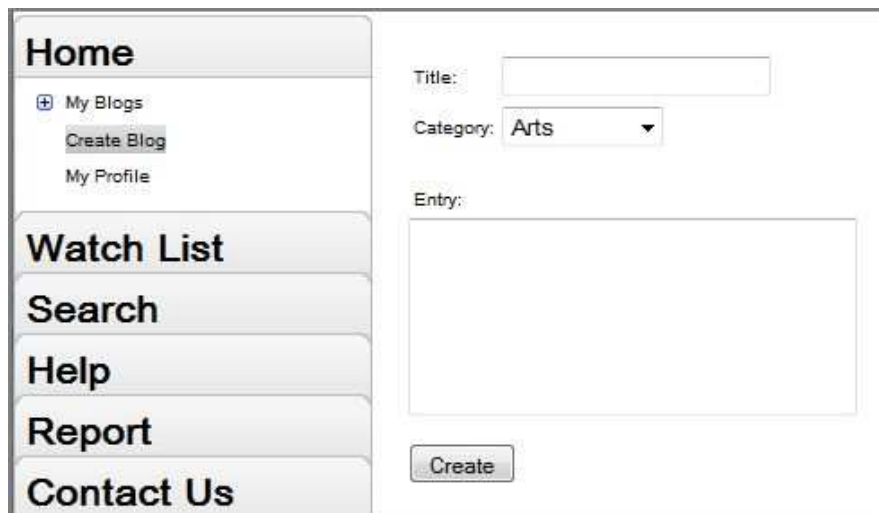


Figure 13: Layout of CreateBlogView using GWT Panels

Since the view classes make heavy use of GWT, the same observation regarding lock-in due to GWT applies. However, as long as the application conforms to the Apache 2.0 license, the code should be fairly portable.

Create Blog View

The Create Blog view allows users to create a new blog entry. The view is a simple page with the blog title input box, the blog category selection box, and the blog body input box (Figure 14).



The screenshot shows a web interface for creating a new blog entry. On the left is a sidebar with a 'Home' header and a list of links: 'My Blogs' (with a plus icon), 'Create Blog' (highlighted), and 'My Profile'. Below these are buttons for 'Watch List', 'Search', 'Help', 'Report', and 'Contact Us'. The main content area on the right has a 'Title:' label followed by an empty text input field. Below that is a 'Category:' label followed by a dropdown menu currently showing 'Arts'. A large 'Entry:' label is followed by a large empty text area. At the bottom right of the main area is a 'Create' button.

Figure 14: Create Blog view

Edit Blog View

The Edit Blog view allows user to edit, delete, or show images associated with the blog entry.

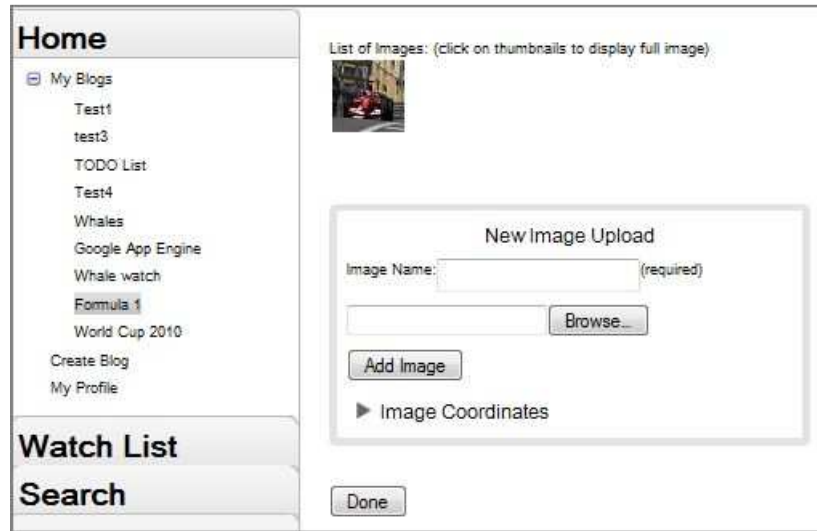


The screenshot shows the 'Edit Blog' view. The sidebar on the left is identical to Figure 14, but the 'My Blogs' link now has a minus icon. The list of blogs includes 'Test1', 'test3', 'TODO List', 'Test4', 'Whales', 'Google App Engine', 'Whale watch', 'Formula 1', and 'World Cup 2010' (which is highlighted). The main content area displays the details for the selected blog: 'Title: World Cup 2010', 'Date Created: Sat Nov 21 11:31:52 GMT-600 2009', and 'Category: Sports'. The 'Entry:' section shows the text 'The stage is almost set for next year's Worl Cup in South Africa' followed by a truncated view '...'. Below the entry are three buttons: 'Edit', 'Delete', and 'Show Images'. At the bottom, there is a 'Comments:' label.

Figure 15: Edit Blog view

Image List Viewer and Uploader

The Image List Viewer displays the images associated with the blog entry. The view also allows the upload of new images with location tags (Figure 17).



The screenshot shows a web interface with a sidebar on the left and a main content area. The sidebar has a 'Home' header, a 'My Blogs' section with a list of blog entries (Test1, test3, TODO List, Test4, Whales, Google App Engine, Whale watch, Formula 1, World Cup 2010), and links for 'Create Blog' and 'My Profile'. Below the sidebar are sections for 'Watch List' and 'Search'. The main content area has a 'List of Images: (click on thumbnails to display full image)' section with a single thumbnail. Below this is a 'New Image Upload' form with an 'Image Name' field (marked as required), a 'Browse...' button, an 'Add Image' button, and an 'Image Coordinates' section with a right-pointing arrow. At the bottom of the main area is a 'Done' button.

Figure 16: Image List Viewer and Image Uploader



This screenshot shows the 'New Image Upload' form with the 'Image Coordinates' section expanded. It includes an 'Image Name' field (required), a 'Browse...' button, and an 'Add Image' button. The expanded 'Image Coordinates' section contains an 'Enter Street Address:' label, a text input field, and a 'Get Lat/Lng' button. Below the input field is an example address: 'ex. "1600 Amphitheatre Parkway, Mountain View, CA"'. At the bottom of this section are labels for 'Lat' and 'Lng'.

Figure 17: Image Uploader expanded

BLOGENTRY AND IMAGEENTRY CLASSES

The BlogEntry and ImageEntry classes are the data structure that handles the blog and image entries. They both make the *model* component in the architecture. The BlogEntry class has the interfaces to create and edit a new blog entry. The ImageEntry class has the interfaces that allow the loading and storage of images in the application. They both implement the Serializable interface allowing them to be passed to the server by the Servlets (see BlogEntryServiceImpl below).

BLOGITEM AND IMAGEITEM CLASSES

The BlogItem and ImageItem classes are the counterparts for the BlogEntry and ImageEntry on the server side. The BlogEntry and ImageEntry are stored in the datastore as instances of BlogItem and ImageItem respectively (Figure 18). The main difference is that the item classes are simple data structures that allow the storage of blog and image information. Both BlogItem and ImageItem classes have only getter and setter functions. On the other hand, BlogEntry and ImageEntry can perform specialized functionality to handle events in the application.

BlogItem Entities							
◀ Prev 20 1-10 Next 20 ▶							
<input type="checkbox"/> ID/Name	m_CreateDate	m_LastUpdated	m_NickName	m_strCategory	m_strEntry	m_strImages	m_strTitle
<input type="checkbox"/> id=7021	2009-10-27 03:53:07.696000	2009-10-29 02:28:50.588000	kajitams1	Auto	This is a test entry on mymicroblog.	7026	Test2
<input type="checkbox"/> id=7025	2009-10-27 04:17:20.128000	2009-10-29 00:17:03.336000	kajitams2	Computers	For best visual results use IE 7 or 8. There are a few items on this page that do not render well on Chrome.	9005	MicroBlog1
<input type="checkbox"/> id=9002	2009-10-27 03:24:23.559000	2009-10-29 02:29:09.923000	kajitams1	Business	Test1	9003,11002,12001	Test1

Figure 18: BlogItem Entity on Google App Engine Datastore

IBLOGENTRYSERVICE AND IBLOGSERVICEASYNCH INTERFACES

The IBlogEntryService and IBlogServiceAsynch interfaces makes up the code needed for the Remote Procedure Call (RPC). The interfaces are used to invoke the server and pass the BlogEntry object between the client and server. Although both interfaces are written in Java, GWT compiles then into JavaScript. Both the client and server serialize and deserialize the data object to and from simple text form during the data exchange.

BLOGENTRYSERVICEIMPL CLASS

The BlogEntryServiceImpl class inherits from the RemoteServiceServlet and implements the IBlogEntryService (Figure 19). RemoteServiceServlet is the servlet base class that manages the RPC calls and automatically performs deserialization on incoming data (e.g., BlogEntry and ImageEntry) from the client and serialization on the outgoing data to the client. Ultimately, the BlogEntryServiceImpl is a Servlet that handles the requests from clients through the methods implemented by the IBlogEntryService interface.

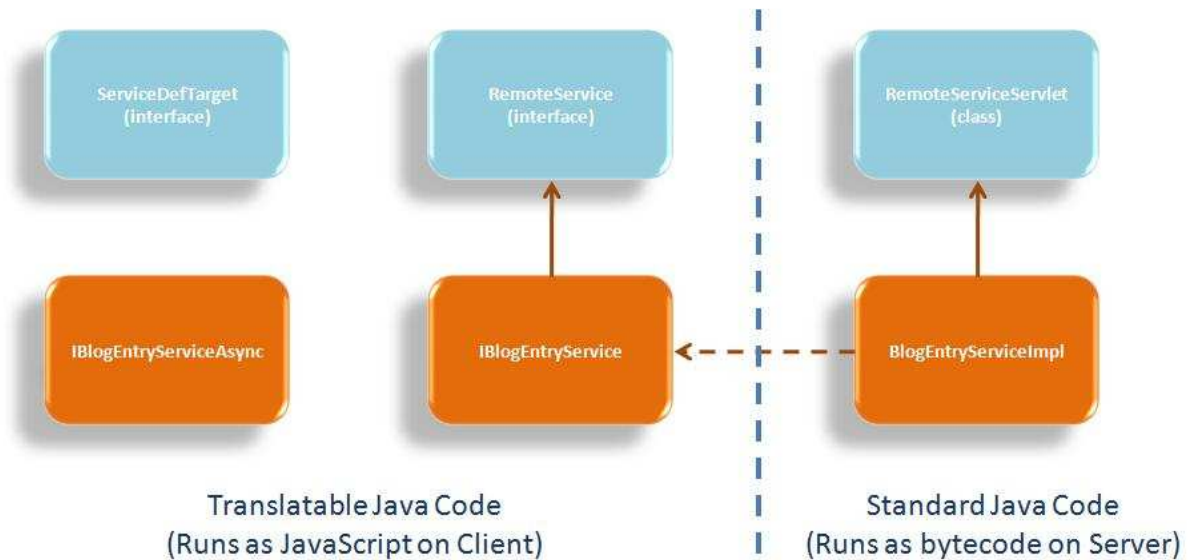


Figure 19: BlogEntry RPC Implementation

The configuration of the new Servlet is done on the war.WEB-INF.web.xml (Figure 20) and MyMicroBlog.gwt.xml files (Figure 21). After adding new Servlets the web.xml file needs to be updated and re-deployed.

```
<!-- Servlets -->
<Servlet>
<Servlet-name>MicroBlogServlet</Servlet-name>
<Servlet-class>microblog.server.BlogEntryServiceImpl</Servlet-class>
</Servlet>

<Servlet-mapping>
<Servlet-name>MicroBlogServlet</Servlet-name>
<url-pattern>/microblog/microblogs</url-pattern>
</Servlet-mapping>
```

Figure 20: BlogEntry Servlet Configuration

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE module PUBLIC "-//Google Inc.//DTD Google Web Toolkit 1.7.1//EN" "http://goo
<module rename-to='microblog'>
  <!-- Inherit the core Web Toolkit stuff.                        -->
  <inherits name='com.google.gwt.user.User' />

  <!-- Inherit the default GWT style sheet.  You can change      -->
  <!-- the theme of your GWT application by uncommenting         -->
  <!-- any one of the following lines.                          -->
  <!-- <inherits name='com.google.gwt.user.theme.standard.Standard' /> -->
  <!-- <inherits name='com.google.gwt.user.theme.chrome.Chrome' />   -->
  <inherits name='com.google.gwt.user.theme.dark.Dark' />

  <!-- Other module inherits                                    -->
  <inherits name="com.google.gwt.http.HTTP" />
  <inherits name="com.google.gwt.maps.GoogleMaps" />
  <script src="http://maps.google.com/maps?file=api&v=2.x&key=ABQIAAAAr14CSGe5
  <!-- <inherits name='com.mapitx.gwt.googleMaps.GoogleMaps' /> -->
  <!-- Specify the app entry point class.                      -->
  <entry-point class='microblog.client.MicroBlog' />
  <servlet path="/ImageServlet" class="microblog.server.ImageServlet"/>
  <servlet path="/EmailServlet" class="microblog.server.EmailServlet"/>
</module>

```

Figure 21: The MyMicroBlog.gwt.xml file

FORMATTING AND STYLE

GWT provides a very simple mechanism to format and style its widgets. By adding a stylesheet the programmer or designer has full control over the look and feel of the overall web application. The display properties of each GWT control can be customized in the stylesheet by following a naming convention. The naming convention follows a strict structure “.gwt-[WidgetName]” (e.g., ‘.gwt-Button’). The stylesheet used in the MyMicroBlog application uses the GWT default styles with a few modifications. There are limited customizations to make the overall display area and button smaller to fit the application in smaller screens. The following is a snippet of the stylesheet used in MyMicroBlog that modifies the appearance of the GWT DialogBox.

```
.gwt-DialogBox {  
    sborder: 8px solid #C3D9FF;  
    border: 2px outset;  
    background-color: white;  
}  
  
.gwt-DialogBox .Caption {  
    background-color: #C3D9FF;  
    padding: 3px;  
    margin: 2px;  
    font-weight: bold;  
    cursor: default;  
}
```

Figure 22: Custom style settings for GWT widgets

The overall application style can also be configured by setting the module to inherit a style from any of the GWT styles (Figure 21). The application style is applied in the Application GWT XML file (i.e., MyMicroBlog.gwt.xml).

IMAGE LOCATION TAGGING

The images loaded via the micro blogging site can be tagged with latitude and longitude values (Figure 23). This enables the application to display the exact location on the map where the picture was taken. The application uses another utility from GWT called Geocoder. The Geocoder is a web service that takes the user input address and converts it into latitude and longitude values. The latitude and longitude values are used to display a “pin” on a map indicating the location where the picture was taken. The map is another GWT tool called MapWidget. The MapWidget allows programmers to display maps on web applications with similar functionalities as in Google Maps.



Figure 23: Image viewer with lat-long values



Figure 24: Image pin on map

Both Geocoder and MapWidget are Java libraries and the integration of location and mapping capabilities to a web application is almost effortless. The only configuration required in order to use any of Google Mapping services is the creation of an account with Google and the use of the key provided with the account in the application xml file (i.e., MyMicroBlog.gwt.xml).

OTHER UTILITY TOOLS

Selenium IDE

The use of automated testing tool was limited during the initial development due to constant changes in the user interface. Once the user interface became stable, Selenium IDE [11] (Figure 25) was used to automatically test the user interface. The tool integrates with Mozilla Firefox and helps record users actions on the web page. Once the recording is complete, the tool helps play back the users actions. The utility assisted finding a bug in the implementation of the blog Servlet where an occasional error was preventing the insertion of new blog entries. Selenium created new blog entries via the CreateBlogView page by automatically filling in the fields. Once the script completed the form and hit the Create button it waited for one second and verified if the entry was on the datastore by querying the newly created entry from the SearchByCategory page. The script repeated the steps every second until the problem re-occurred. The script helped pinpoint the problem and resolve it. When a new blog entry was inserted to the datastore an error was occurring intermittently because the Create method in BlogEntryServiceImpl on the server was not using javax.jdo.Transaction. Once the problem was resolved, the same script was used to verify the fix.



Figure 25: Selenium IDE for Firefox

WORK BREAKDOWN

Table 2 shows the breakdown of the major tasks performed during the development of the MyMicroBlog application.

MyMicroBlog Work Breakdown			
Task	Feature	Development Time (days)	Comments
MyMicroBlog Application	Overall application setup	0	Application setup took no time due to the help of the Application and Deployment Wizards.
	Client-server folder structure		
	Deployment configuration		
	Deployment files		
	Application Deployment		
Requirements Gathering		1	Time spent gathering requirements
Specification		1	Time spent drafting the specifications
Architecture		2	Total time spent on the architecture of the MyMicroBlog architecture. This includes the time spent on researching for modeling tools to use in Eclipse. These modeling tools helped generate the class and data flow diagrams used to define the software architecture.
Design		7	Time spent on designing the MyMicroBlog system. This includes the time spent on researching for the different GWT technologies and prototyping. About 80% of the time spent in this stage was on reading and learning on GWT.

Table 2: Work breakdown of main development tasks

Task	Feature	Development Time (days)	Comments
Implementation	Total 11 views	3	Once the IBlogView interface was defined, the development of the different views were very straight forward. The system now supports the plug-in for any view seamlessly.
	MicroBlog controller	4	The initial implementation of the MicroBlog class was not intended to use it as a controller. As a result, it required refactoring to adapt to the MVC pattern and support the IBlogView interface.
	Blog Entry Data Object	3	Large amount of time was spent on debugging problems with the servlet configuration files. Automatically generated files helps development but when edited manually can lead to hard to find bugs.
	Image Upload	7	95% of the time was spent on debugging the Servlet code and GWT configuration files (i.e., xml files). The initial implementation was done using sample code from online sources. It turned out that some of these sources contained errors in the Servlet code or configuration files. The errors were very hard to find because at runtime the debugger generated messages that were very cryptic and hard to understand. Some of the errors were due to an extra '/' in front of the path name in the xml file and only caused problems at runtime.
	Image Data Object	3	Similar problems understanding the need for dedicated objects in the server side caused delays in the development of the image data object. It is important to understand that client and server objects follow very strict Servlet or JDO rules.
	Style and Formatting	2	Time spent on researching for techniques to apply styles to the web application.
	Mapping	1	The development of the mapping feature was very straight forward.
	Total	34	

Table 2, cont.: Work breakdown of main development tasks

RESULTS

Development Experience

The overall development experience in GAE was extremely rewarding. The platform and GWT allow for quick development of interactive web applications. The GWT plug-in for Eclipse is a must have. There is very little to do as far as configuration and setup once you use the Create and Deploy Application Wizards. The developer can truly focus on the architecture and design of the web application. The resulting code base is very small. Table 3 shows the breakdown of the total number of lines of code in the client and server side of the application. Table 4 shows the average time it takes to compile and deploy the application to Google servers.

Total Lines of Code	
Client side	3406
Server side	1782

Table 3: Total lines of code in the client and server side

Project Deployment	
	Avg. (sec)
Compilation Time	47.876
Files Upload Time	25.362
Total Deployment Time	73.238

Table 4: Project deployment time information

Table 5 shows the system description and development environment on which the application was developed

System Description	
Operating System	Windows Vista
System Type	32-bit Operating System
Processor	Intel Dual Core 1.73GHz
Memory (RAM)	2.0 GB
Laptop Model	Hewlett-Packard Presario C500
Development Environment	
IDE	Eclipse Galileo
Java Version	1.6.0_03
Network Information	
Type	Wireless Broadband
Download	7.18 Mb/sec
Upload	4.71 Mb/sec

Table 5: System description and development environment

Performance

The overall performance of GWT does not seem to affect the usability of the application. GWT compiles the GWT components in optimized JavaScript, so at run time there is no performance difference between JavaScript coded manually and auto generated by GWT. In fact, Google states that the JavaScript code generated by GWT performs better because of optimizations.

The only noticeable performance problem is during the initial creation of the PersistenceManagerFactory. The PersistenceManagerFactory used during JDO data transactions is very costly at first. In order to cope with the initial hit, a singleton class wraps the PersistenceManagerFactory instance. After the singleton class is created, the

PersistenceManagerFactory instance is kept in memory, and there is no significant performance hit thereafter. The performance hit is noticeable only if the singleton is unloaded due to low traffic or the application is reloaded. In that case, the user who visits the site for the first time will notice a 5~6 seconds of delay (Table 6).

	Page load time (msec)	Comments
First time	~6000	Most of the time is spent on the instantiation of the PersistenceManagerFactory.
Subsequent times	<500	PersistenceManagerFactory is in memory and JDO transactions can take place immediately.
Note: The time was measured at the server side by placing timers in the server code and it does not account for the network latency. The objective was to find the place in the code that was causing the long page load time. The number above is the average of 100 page loads.		

Table 6: First vs. Subsequent page load time

Table 7 shows the page load time comparison between MyMicroblog and some of the commercial web sites. The value is the average of 1000 page load requests. The measurement was taken programmatically using a script written in C# as shown in Figure 26.

MyMicroblog	Twitter	Google	Yahoo	CNN	MSN	Ebay
487.5577194	379.7156	492.8234	542.9673	503.1726	550.1769	579.463

Table 7: Page load time measurement in msec.

```

HttpWebRequest request = (HttpWebRequest)WebRequest.Create("http://mymicroblog.appspot.com/");
//HttpWebRequest request = (HttpWebRequest)WebRequest.Create("http://twitter.com/");
//HttpWebRequest request = (HttpWebRequest)WebRequest.Create("http://google.com/");
//HttpWebRequest request = (HttpWebRequest)WebRequest.Create("http://yahoo.com/");
//HttpWebRequest request = (HttpWebRequest)WebRequest.Create("http://cnn.com/");
//HttpWebRequest request = (HttpWebRequest)WebRequest.Create("http://msn.com/");
//HttpWebRequest request = (HttpWebRequest)WebRequest.Create("http://ebay.com/");

Stopwatch timer = new Stopwatch();

for (int nIndex = 0; nIndex < 1000; nIndex++)
{
    timer.Start();
    HttpWebResponse response = (HttpWebResponse)request.GetResponse();
    if (response.StatusCode == HttpStatusCode.OK)
    {
        timer.Stop();
        TimeSpan timeTaken = timer.Elapsed;
        Debug.WriteLine(string.Format("{0}", timeTaken.TotalMilliseconds));
    }
}

```

Figure 26: Script used to measure page load time

Figure 27 shows the statistics generated by YSlow [12] for Firebug when loading the MyMicroBlog main page. According to the report generated by YSlow, the main page makes poor use of Content Delivery Network (CDN) and gave MyMicroBlog main page an overall grade of C. However, this is because the main page displays images and use stylesheet and these are standard files for any web page. Figure 28 shows the YSlow statistics for a commercial blogging site. The YSlow report also points out the problem with CDN and gave a D grade to the site.

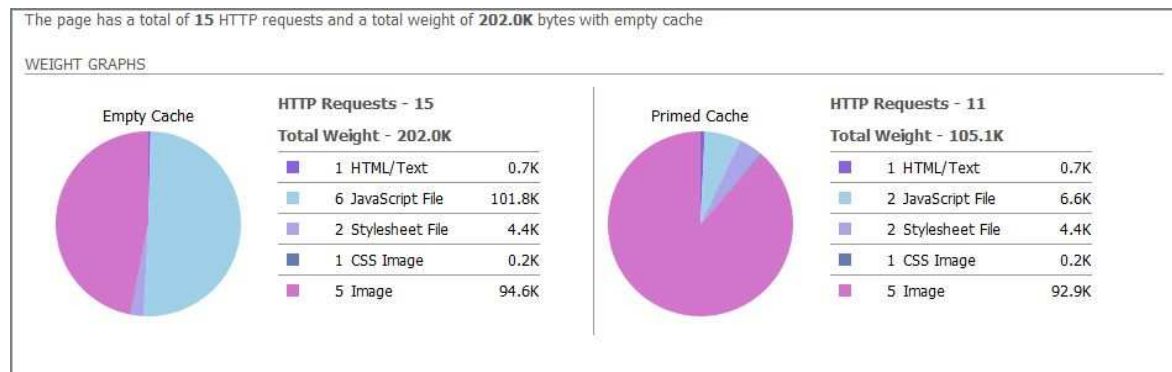


Figure 27: YSlow statistics for MyMicroBlog main page

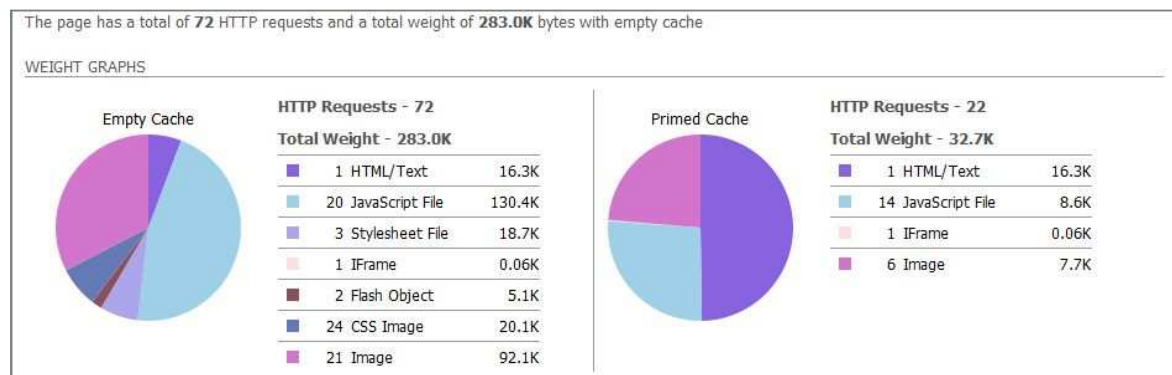


Figure 28: YSlow statistics for a commercial blogging site

Lessons Learnt

The Create Application and Deploy Application wizards are very useful. They automatically generate configuration files, folder structure, and source code. The configuration of the application is done once, so the auto configuration feature is very helpful. The auto generation of the greeting page is useful because the wizard builds the package structure for the client and server as well. The only drawback of auto generation of code and configuration files is that the developer may end up with unnecessary code. Debugging through auto-generated code can be time consuming due to unfamiliarity and amount of unnecessary code. Errors in the configuration file introduced by new features

added manually can be hard to detect, especially when programmer is a novice on Servlet coding. As reported in table 2, most of the time spent during the development of the client and server code was spent on troubleshooting Servlet and configuration file errors. These errors can be extremely hard to find if the programmer does not have a good understanding on the technology. The wizard does most of the work, but it is important that the programmer knows exactly what the wizard is doing because manual changes are needed.

FUTURE WORK

There is room for improvement on the MyMicroBlog platform. The current implementation requires refactoring on the MicroBlog class in order to make the platform truly modular. For example, the complexity of the MicroBlog controller class does not allow for easy plug-in of new functionality in the navigation bar. The two refactoring points made in the MicroBlog Class section will result in a better design.

Security is also another aspect that needs attention in the platform. The calls to blog and image Servlets have no authentication at the moment. Any malicious web crawler could invoke the Servlets causing unnecessary loads on the server.

The user interface uses the default GWT styles. The default styles are regular Google styles and the page has the look and feel of any other Google application. Adding a mechanism to load a custom style sheet will allow the user to add personality to the web application. GWT controls have their own properties and need to be set on the main style sheet.

The image location tagging uses only the user entered street address. Further improvements to the system will be to allow users to use GPS coordinates from devices such as the iPhone to tag images.

Conclusion

There is no doubt that cloud computing has opened new business opportunities and has started a new Internet revolution. For end users, the cloud promises unlimited computing power at very affordable prices. Creative companies can take advantage of the cloud services to deploy applications that before were too expensive to develop or maintain. There is the opportunity of getting rid of the IT department in the company all together resulting in savings and less overheads. For cloud providers, there is a new business opportunity where commoditized computer resources can be sold as pay-as-you-go model.

The race is on for cloud providers and the market does not have a clear winner yet. All the cloud solutions available in the market today make the early stage of the platform development cycle. Until the cycle comes to a full circle, users should gain experience on the cloud technologies available. This is a very important step to create the sense of direction in the industry and the creation of standards. From the start of the cycle to the end when the cloud computing platform stabilizes, users will most likely be winners since the capital to experiment on technologies such as GAE is low. GAE has proven to be a well thought-out and designed platform. The use of standard APIs in GAE allows the development of well designed and attractive web applications. The experience on GAE shows that users can focus on the business logic while developing flexible and portable web applications.

Bibliography

- [1] Armbrust, M., et al., Above the Clouds: A Berkeley View of Cloud Computing, February 10th, 2009.
- [2] Delic, K. A., et al., Emergence of the Academic Computing Clouds, Hewlett-Packard Co., ACM Ubiquity, Volume 9, Issue 31, August 5, 2008.
- [3] Google App Engine, <http://code.google.com/appengine/>
- [4] Hinchcliffe, D., How the Web OS Has Begun to Reshape IT and Business, September 6, 2009, <http://blogs.zdnet.com/Hinchcliffe/?cat=78>.
- [5] Rogers, G., The Problem with Google App Engine, April 11, 2008, <http://blogs.zdnet.com/Google/?p=1002>
- [6] Chang, F., et al., Bigtable: A Distributed Storage System for Structured Data, OSDI'06: Seventh Symposium on Operating System Design and Implementation, Seattle, WA, November, 2006, <http://labs.google.com/papers/bigtable-osdi06.pdf>
- [7] DataNucleus Access Platform, <http://www.datanucleus.org/>
- [8] Hinchcliffe, D., Twitter in your Intranet: 17 Micro Blogging Tools for Business, June 1, 2009, ZDNet, <http://blogs.zdnet.com/Hinchcliffe/?p=414>.
- [9] Hinchcliffe, D., Cloud Computing and the Return of the Platform Wars, March 26, 2009, ZDNet, <http://blogs.zdnet.com/Hinchcliffe/?p=303#more-303>.
- [10] Hinchcliffe, D., Cloud Computing: A New Era of IT Opportunity and Challenges, March 3, 2009, ZDNet, <http://blogs.zdnet.com/Hinchcliffe/?p=261#more-261>.
- [11] Selenium IDE Web Application Testing System, <http://seleniumhq.org/projects/ide/>

[12] Yahoo! YSlow, <http://developer.yahoo.com/yslow/>

Vita

Marcos Suguru Kajita was born in Ituiutaba, Minas Gerais, Brazil. After completing high school at CETEBAN in Tokyo, Japan, he entered Hawaii Pacific University in Honolulu, Hawaii. After completing his freshman year at Hawaii Pacific University, he transferred to the University of Texas at Austin and enrolled in the Computer Science program. In May of 2003, he graduated from the University of Texas at Austin with a Bachelor of Science in Computer Science. He has been working in a major oil service company since his graduation in 2003 as a software engineer. In January of 2007, he entered the Graduate School at the University of Texas at Austin and enrolled in the Software Engineering Program

Permanent Address: 2703 Skyview Crest Ct.
Houston, Texas 77047

This report was typed by Marcos Suguru Kajita