2Z-6

Go 言語による GAE アプリケーションの開発

石井　涼†,　鈴木美穂‡,　大谷　真‡
湘南工科大学†

1.　はじめに

PaaS 型クラウド基盤である GAE（Google App Engine）は 1.5.0 で Go 言語のサポートが始まった。Go 言語は 2009 年 Google によって開発された新プログラミング言語である。一方、GAE アプリケーションの開発を対象としたときの Go 言語の適用性は現時点では必ずしも明らかでない。本研究では、GAE アプリケーションでの代表的な処理（画面遷移制御、Datastore アクセス、Blobstore 利用、HTML 生成など）に関して、実際のアプリケーションの開発を通して Go 言語の有効性を評価した。

2.　簡易ブログシステム「ごぶろぐ」の開発

2.1　「ごぶろぐ」の機能

Go 言語による GAE アプリケーションの開発の実験素材として、以下の機能を持つ簡易ブログシステムをとりあげた。

・管理者ログイン機能・管理者アカウント作成
・記事の投稿　　・画像の投稿

画面遷移は図 1 のようになっている。

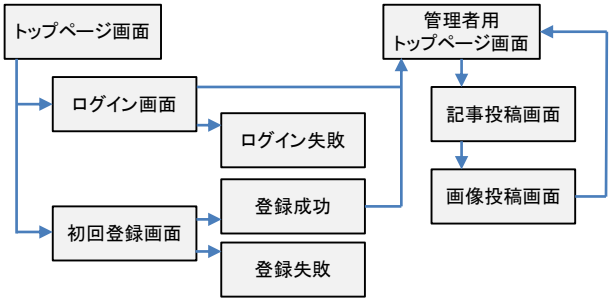

図 1　画面遷移図

（1）トップページ

記事の閲覧とログイン画面への移動ができる。登録されているアカウントがない場合、初回登録画面に移動する。管理者用トップページでは記事投稿画面とログアウトのためのリンクが用意されている。

（2）初回登録・ログイン

ID 用、パスワード用のフォームに任意の文字を入力する事で登録が出来る。2 回目以降のアクセスはログインページとなり、登録 ID とパスワードを入力する事でログインできる。

Development of GAE Applications by Go Language
†Ryo Ishii, Miho Suzuki, Makoto Oya,
Shonan Institute of technology

（3）記事・画像の投稿

タイトル、本文を入力し、投稿ボタンをクリックするとその内容が BigTable に登録される。画像を添付したい場合はフォームでファイルを指定することで、画像ファイルの参照キーが記事と一緒に登録される。

2.2　Datastore の構成

GAE には BigTable と呼ばれる列指向 DBMS が用意されている。BigTable はキーと値が対になっているエンティティによって構成され、エンティティの集まりをカインドと呼ぶ。記事を管理する Report カインドとアカウント名・パスワードを管理する Kagi カインドを用意した。

表 1　カインド構成

カインド名：Report	Title	Content	Date	Img
	記事タイトル	記事本文	日付	参照キー

カインド名：Kagi	Acc	Pwd	Register
	アカウント名	パスワード	登録済フラグ

2.3　Blobstore API の利用

GAE には Datastore とは別に、最大で 2GB のデータオブジェクトを処理できる Blobstore が用意されている。ファイルアップロード用のフィールドを含むフォームを web ページに設置し、フォームが送信されると Blobstore 内に Blob が作成され、Blob への参照キーが返される。本研究では画像ファイルの格納に利用した。

3.　Go 言語を用いた実装

3.1　画面遷移制御

Go 言語では、http パッケージに含まれる HandleFunc 関数を使うことで関数ごとに Web ページを割り当てる事ができる。例えば

```
http.HandleFunc("/first", loginfirst)
```

第 1 引数には割り当てたい URL を、第 2 引数には関数名を指定する。本プログラムの各関数と Web ページの関係を表 2 に示す。

表 2　各関数に割り当てたページの概要

関数名	Webページ	関数名	Webページ
main	トップページ	registsuccess	初回登録成功画面
login	ログイン画面	registerror	初回登録エラー画面
loginchk	ログインチェック(非表示)	loginmain	管理者用トップページ
loginerror	ログイン失敗画面	contribute	記事投稿画面
loginfirst	初回登録画面	image	画像投稿画面
regist	初回登録実行(非表示)	contributesign	投稿実行(非表示)

また、Go プログラム内もしくは外部ファイルとして HTML を扱うことが出来る。ページ間の移動は基本的に HTML の<a>タグや<form>タグ

を利用したが、一部のページでは下例のようにリダイレクトを用いた部分もある。

```
q := datastore.NewQuery("Kagi").Filter("Register =","1")
kgc, err := q.Count(c)
if err != nil {
  http.Error(w, err.String(), http.StatusInternalServerError)
}
if kgc == 0 {
  http.Redirect(w, r, "/first", http.StatusFound)
  return
}
```

上記はログイン画面にアクセスした際にアカウントが登録されているか判断する部分になる。まず、Kagi カインドの項目 Register に登録済フラグ“1”が格納されているエンティティを検索する。次に、検索件数をカウントしたものを変数 kgc に代入する。結果が 0 件ならば Redirect 関数により、“/first”へリダイレクトされる。

3.2 Datastore のアクセス

（1）カインドの登録

Datastore の利用には、カインドを構造体の形で定義しておく必要がある。構造体の名前がカインド名、メンバ名と型が各項目になる。

カインドへの登録も同じく構造体でのみ受け付けている。各項目に値を代入後、datastore パッケージに含まれる Put 関数でその構造体とカインド名を指定すればよい。

```
rep := Report{
  Title:   r.FormValue(“Title”),
  Content:   r.FormValue(“Content”),
  Date:   datastore.SecondsToTime(time.Seconds()),
  Img:   "/serve/?blobKey="+string(file[0].BlobKey),
}
datastore.Put(c, datastore.NewIncompleteKey(c, "Report", nil), &rep)
```

（2）カインドからの出力

カインドの内容を出力するには、スライスを利用した。スライスとは、配列へのポインタとサイズ、最大サイズの 3 つの基本データを持った構造体のようなもので、スライス経由で配列の読み書きができる。

以下に記事の閲覧部分を例にあげると、まず make 関数を使用し reports スライスを定義する。次に、Report カインドのデータを取得し、reports スライスに格納する。それを Execute 関数により HTML ファイルと関連付けさせる事で web ページにカインドの内容を出力する。

```
reports := make([]Report, 0, 10)
q := datastore.NewQuery("Report").Order("-Date")
if _, err := q.GetAll(c, &reports); err != nil {
  http.Error(w, err.String(), http.StatusInternalServerError)
}
mT, _ := template.ParseFile("tmpl/main.html")
if err := mT.Execute(w, reports); err != nil {
  http.Error(w, err.String(), http.StatusInternalServerError)
}
```

3.3 Blobstore へのアップロード

HTML ファイルで<form>タグを利用し、action 属性に Blobstore へのアップロード用 URL 指定する。この URL は、Blobstore パッケージに含まれる UploadURL 関数を利用することで生成され、HTML 内の“{{.}}”に代入される。処理後は第 2 引数の URL にジャンプする。

```
c := appengine.NewContext(r)
uploadURL, err := blobstore.UploadURL(c, "/contrisign", nil)
if err != nil {
  http.Error(w, err.String(), http.StatusInternalServerError)
}
iT, _ := template.ParseFile("tmpl/image.html")
if err := iT.Execute(w, uploadURL); err != nil {
  http.Error(w, err.String(), http.StatusInternalServerError)
}

<form action="{{.}}" method="post" enctype="multipart/form-data">
<input type="file" name="file">
<input type="submit" value="投稿">
```

3.4 HTML へのデータの埋め込み

HTML ファイルを割り当てる際に、関数内で処理された変数やカインドに保存されているデータを埋め込むことが出来る。下のように表示したい変数名を”{{“と”}}”で囲む。

```
<td class="RepMain" colspan="2">{{.Content}}</td>
```

4. 評価

「ごぶろぐ」開発を通して、GAE アプリケーション開発への Go 言語適用を評価した主な結果は以下のとおりである。

- 画面制御：HandleFunc 関数の利用により、Web ページとの関連を容易に持たせることができた。
- Datastore アクセス：構造体の形にすることでカインドと DB 構造の関係が分析できた。
- Blobstore 利用：UploadURL 関数を用いて参照するためのキーを生成できた。
- HTML 生成：Execute 関数と HTML ファイルへの埋め込みにより出力が多様になった。

以上により、Go 言語適用の有効性が検証できたと考える。

5. まとめ

GAE と Go 言語はまだ実験的な要素を含み使用も安定してはいないが、今後のクラウドでの利用が期待できる。

6. 参考文献

[1] golang.jp, http://golang.jp/
[2] Getting Started: Go, http://code.google.com/intl/ja/appengine/docs/go/gettingstarted/