ケータイサイト30分クッキング

Seasar Conference 2009 Autumn

Shin Takeuchi

プロフィール

- 竹内真（TAKEUCHI SHIN） id:stakeuchi
  - 株式会社レイハウオリ　代表取締役CTO
    - mobylet作ってます。
    - 事業立案や、たまにちょっとしたデザインやったり。
    - 経営と財務管理もやります。
  - 株式会社ビズリーチ　Chief Architect
    - High-Class求人サイト「ビズリーチ」を1人で制作中。
  - 株式会社リクルート　パートナー
    - 共通化を推進したり、フレームワークを作ったり。
    - セキュリティ/SEO/モバイルも担当しています。

このセッションでは何やるの？

- ケータイサイトを実際に作ってみます。
  - 「mobytter」を作ってみましょう。
    - 「Twitter」みたいにつぶやけるサイト
    - 3キャリア携帯対応します。
    - Google App Engineにアップして、公開させます。
    - T2-Frameworkを使います。
    - もちろんmobyletは使います。
    - 時間がある限りリッチなサイトにしてみます。

作る前に。。。

- ちょっとお勉強しましょう。
  - mobyletとは？
  - T2-Frameworkとは？
  - Google App Engineとは？

mobylet（モビレット）とは？

- 携帯Webアプリを作るためのフレームワーク
  - 色々なFWやDIコンテナと一緒に使える（確認済：Seasar2（SAStruts）、T2等）
  - 最小設定はFilterを仕掛けるだけ
  - 外部ライブラリにほぼ非依存（※）
  - Google App Engine for Javaに対応している
- mobyletに出来ること
  - 絵文字の入出力に「勝手に対応」
  - キャリアや機種を「勝手に判別」
  - 携帯サイトに便利な機能が「勝手に標準搭載」

T2-Frameworkとは？

- とってもシンプルなWebフレームワーク
  - 極めて小さなフレームワーク
  - RESTライクな綺麗なURLを簡単に使える
  - 拡張性が高く、色んなDIコンテナを使える
  - Google App Engine for Javaに対応している
- もっと詳しく言うと。。。
  - この後のセッションを聞いて下さい！

Google App Engineとは？

- Googleのインフラが使えるクラウドサービス
  - Googleのミニ環境が無料で使える（※）
  - 今年の4月7日からJavaが利用可能になった
  - Eclipseと親和性が高く、Javaエンジニアに優しい
- 色々な制限もあります
  - RDBMSが使えない（Bigtableが利用可能）
  - Threadが使えない
  - ファイルの書き込みが出来ない
  - その他（Mail等も）色々制限がある

mobylet+T2 / Google App Engine

- これまでの悩み
  - 携帯サイトはインターネット上に公開しなければ実機で見れないけど、Javaインフラは高価！
  - ちょっとしたサイトを作るだけなのに、やたらと大規模な構成（FW等）になりがち！
  - そもそもJavaで作ると携帯依存の問題が膨大！
- mobylet+T2 / Google App Engineの場合
  - 全部解決！！！超Easy！！

それではそろそろ作り始めましょうか

- 準備
  - Eclipseのインストール（デモはVersion 3.3）
    - 流石にこれはデモしません。。。
  - Google App Engineのアカウント取得
    - ちょっとだけ説明します。
  - Google App Engine用のEclipse plug-in
    - ちょっとだけ説明します。
  - mobylet + T2 / Google App Engine
    - ここからデモになります。

(1)Google App Engineのアカウント取得

- http://code.google.com/intl/ja/appengine/

- Google アカウントと携帯メアドがあればOK。

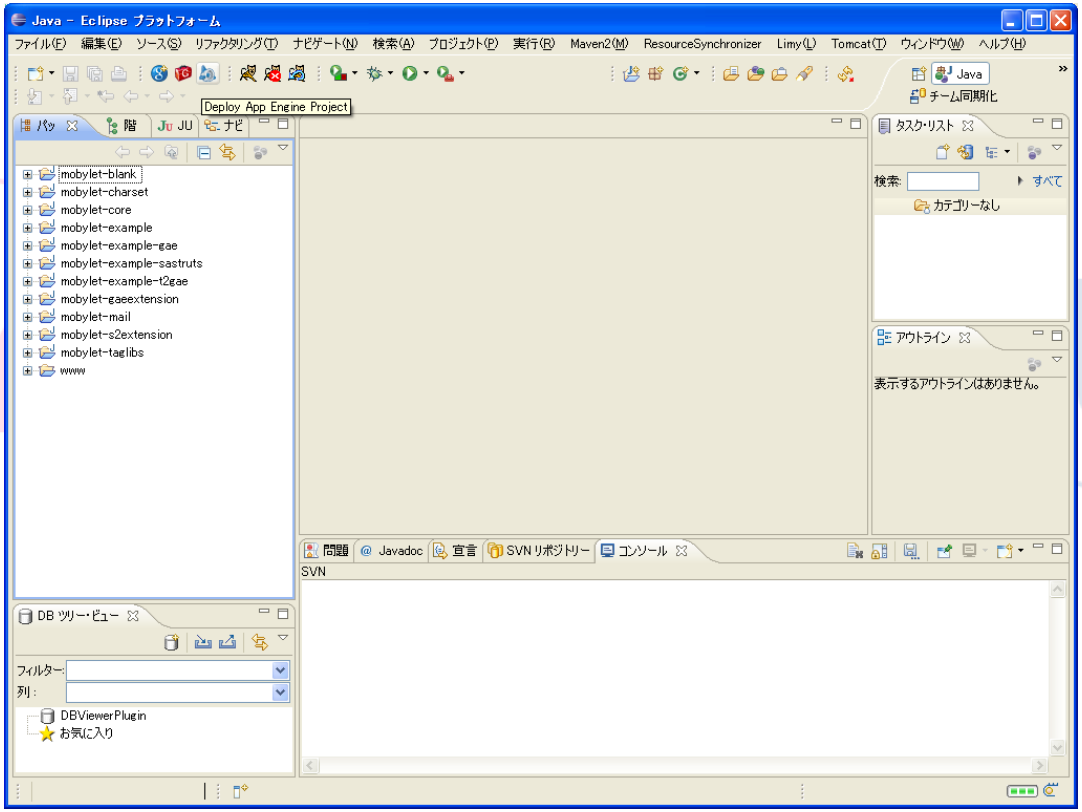
(2)Google App Engine Eclipse Plug-in [1]

- http://code.google.com/intl/ja/appengine/docs/java/tools/eclipse.html

- Plug-inをEclipseにインストールします

(2)Google App Engine Eclipse Plug-in [2]

- インストールすると



  - こんな感じになります。(GAEのアイコンが現れます)

(3)mobylet+T2 / GAE プロジェクト

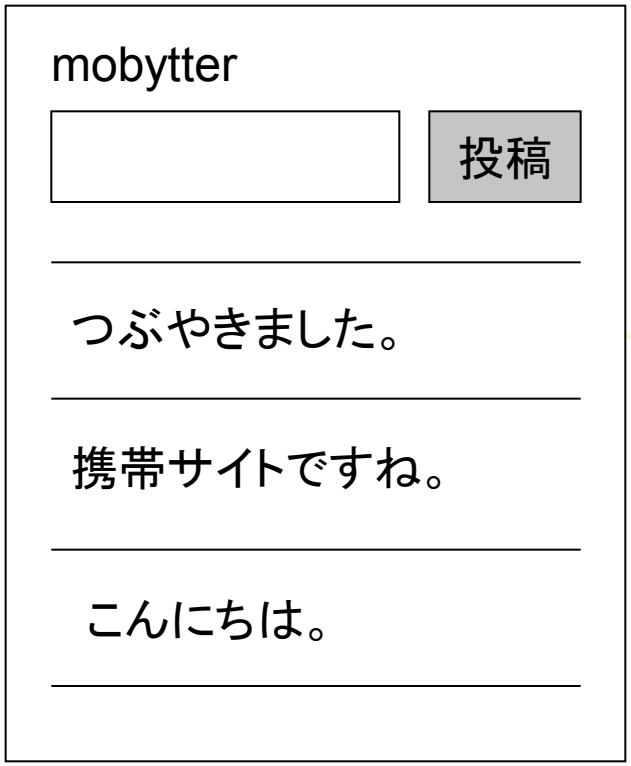
- 実は。。。
  - ブランクプロジェクトを用意しました。

    http://mobylet.sandbox.seasar.org/resources/0.9.1/mobylet-blank-t2gae.zip

    （DIコンテナにGuiceを利用しています）
  - まずはダウンロードします。
  - そしてプロジェクトにインポートします。

それでは作りましょう

- まずは画面設計します。
  - 1画面で完結する簡単なTwitterにします。



Google App EngineでIDを取得

- アプリケーションIDを取得する
  - アプリケーションIDがURLとなったり、デプロイ時のキーになったりするので、最初に取っておく。
    - 今回は「mobytter」で取ってみる。
    - 時間が無いかもしれないのでIDだけ取得済み。
  - 下のURLから取得する（あらかじめGAEのアカウントが必要です）

    https://appengine.google.com/

(4)設定ファイルを確認しましょう

- WEB-INF/web.xml
  - フィルタの設定をチェックします。
- WEB-INF/logging.properties
- WEB-INF/appengine-web.xml
  - ログの設定をチェックします（基本そのまま）
- META-INF/jdoconfig.xml
  - JDOの設定をチェックします（基本そのまま）
- logback.xml
  - ログ（T2）の設定をチェックします（そのまま）

(5)mobylet.xml [1]

- mobylet.xmlの設定をチェックする
  - initializers/chain
    - 初期化クラスの設定（内部コンテナの初期化）
  - emoji
    - emoji/basedir
      - 絵文字のキャリア間変換設定ファイル配置場所
    - emoji/imagePath
      - OTHERキャリアでの絵文字画像出力パス
  - device
    - device/basedir
      - バリューエンジン社の端末プロファイル設置場所

(5)mobylet.xml [2]

- mobylet.xmlその他の設定
  - through/carrier
    - mobyletに余計な処理をさせたくないキャリアの設定
  - proxy
    - proxy/host
    - proxy/port
      - プロキシサーバが介在する通信処理を行う場合に設定
  - default
    - default/contentType
      - HTML/XHTMLのどちらかを指定可能（初期値はHTML）

(6)静的なトップページの作成 [1]

- index.jspの作成
  - 今回は直接JSPにアクセスさせたくないので「WEB-INF/view/」の下とかに作りましょう。
  - とりあえずタイトル部とフォーム部だけ作ってみましょう。
  - こっそり絵文字も入れてみましょう。

(6)静的なトップページの作成 [2]

- pageディレクティブ
  - <%@page pageEncoding="UTF-8" isELIgnored="false" %>
    - Content-Typeはmobyletが設定するので書かなくて良い。
    - GAE/Jはデフォルトで EL式が使えないので、isELIgnored="false"を指定する。

(6)静的なトップページの作成 [3]

- taglibディレクティブ
  - どうせ使うので2つ書いておきます
    - <%taglib prefix="c" uri="http://java.sun.com/jsp/jslt/core" %>
    - <%taglib prefix="m" uri="http://taglib.mobylet.org" %>
  - GAEではinclude-preludeが使えないのでcommon.jspとかにtaglib指定を書いておくようなことが出来ないので注意。

(7)静的なトップページのPageクラス [1]

- TopPage.class
  - T2でRESTライクなURLを使いたい。
  - 名前は何でも良いので、これにする。
  - POJO（Plain Old Java Object）で良い。
  - rootpackage配下のpageパッケージに作る。

(7)静的なトップページのPageクラス [2]

- TopPageクラスにアノテーションを付ける
  - @SingletonScope
    - シングルトンインスタンスで良い時はこれを付ける。
  - @Page(“/top”)
    - http://hoge.com/topとしてアクセスさせる指定
      - GAEはコンテキストパスが無いので↑な感じになる。

(7)静的なトップページのPageクラス [3]

- 処理するメソッドを作る
  - public Navigation index(WebContext context)
    - 処理メソッドはNavigatorクラスを返すお決まり
    - 引数は色々設定可能だが、WebContextを取る王道でやってみましょう
  - @Defaultアノテーションを付ける
    - GETでもPOSTでも、どんなMethodでリクエストされても処理出来るように、このアノテーションを付けます。
      - 限定したい場合は@GETとか@POSTとか使います

(7)静的なトップページのPageクラス [4]

- JSPを表示する
  - return Forward.to(“/WEB-INF/view/index.jsp”)
    - JSPのパスを指定して表示させる。
    - Forward.to()を使うことで、WEB-INF以下のパスも指定可能。

(8)投稿を処理するPageクラス [1]

- PostPage.class
  - とりあえず同じように作っておきましょう
  - URLは「post」にしましょう
  - メソッドは相変わらず@Defaultにします
  - 処理したら表示ページに返しましょう
    - Redirect.to(“/top”)

(8)投稿を処理するPageクラス [2]

- せっかくなのでDIしてみる
  - MobyletインスタンスをDIする。
    - META-INF/services/com.google.inject.module
    - src/org/mobylet/t2gae/MobyletModule
  - Pageクラスを@RequestScopeに変更
    - MobyletインスタンスがRequestScopeのため
  - @Inject protected Mobylet mobylet;
    - これでDI可能
  - これで色々な情報が取れます

(9)投稿をBigtableにストアする [1]

- まずはBigtableを超簡単に理解しましょう。
  - Bigtableとは「key-value」ストアエンジンです。
  - イメージはJavaのHashMapみたいなものです。
  - MapにputしたらDBに保存されていると思えば、イメージは大体オッケイです。
  - MapからgetしたらDBからデータが取得できていると思えば、イメージは大体オッケイです。
  - 今回はJDOを使ってBigtableにアクセスするので、JDOQLなる、変なQueryが使えますが、あまり気にしない方が良いです。
  - 詳しく知りたい方は以下のURLへ！
    http://code.google.com/intl/ja/appengine/docs/java/datastore/
  - さらに詳しく知りたい方は「ひがさんのセッション」へ！

(9)投稿をBigtableにストアする [2]

- ストアするデータクラスを作りましょう
  - 一意になるkeyと、格納したいメンバを持つデータクラスを作ります。
    - Long key
      - 一意なキーを自動採番させます。
    - String comment
      - 投稿したコメントをここに入れます
    - Date postDate
      - 投稿日時をここに入れ、この日時でソートします。
  - Class名はCommentDtoとでもしておきます。
    - rootpackage/page以外のパッケージに作りましょう

(9)投稿をBigtableにストアする [3]

- データクラスにアノテーションを付ける
  - クラスに付けるアノテーション
    - @PersistenceCapable(identityType=IdentityType.APPLICATION)
      - JDOを使ってデータの格納/取得をするために必要
  - Keyとなるメンバに付けるアノテーション
    - @PrimaryKey
    - @Persistent(valueStrategy=IdGeneratorStrategy.IDENTITY)
      - 自動採番させるための指定です
  - メンバに付けるアノテーション
    - @Persistent
      - Bigtableに格納することを宣言するものです

(9)投稿をBigtableにストアする [4]

- 格納するデータインスタンスを作りましょう
  - Requestからパラメータを取得します。

```
Request request = context.getRequest();
String comment = request.getParameter(“comment);
```

  - Nullチェックでもしておきましょうか。

```
if (StringUtils.isEmpty(comment)) {
  Redirect.to(“/top”);
}
```

  - 格納するデータインスタンスを作ります。

```
CommentDto dto = new CommentDto();
dto.setComment(comment);
dto.setPostDate(new Date());
```

(9)投稿をBigtableにストアする [5]

- PersistenceManagerを使ってstoreします。
  - ブランクプロジェクト付属のPMFクラスを使って取得します。

```
PMF.get().getPersistenceManager();
```

  - makePersistentメソッドでstoreして、終わったらちゃんとクローズします。

```
try {
  pm.makePersistent(dto);
} finally {
  pm.close();
}
```

(10)ストアした情報を取得/表示する [1]

- まずは取得する
  - やっぱりPersistenceManagerを使います。
    PMF.get().getPersistenceManager();
  - Queryクラスを使って取得したいと思います。(他にも方法はありますが)

    Query query = pm.newQuery(CommentDto.class);
  - ソート条件を指定します

    query.setOrdering("postDate desc");
  - 先頭の30件だけ表示することにします

    query.setRange(0, 30);
  - オブジェクトのリストを取得します

    (List<CommentDto>)query.execute();
  - 取得し終わったらちゃんとクローズします

(10)ストアした情報を取得/表示する [2]

- Requestの属性に情報を設定する
  - CommentDtoのリストからコメントだけを抜き出してリスト化する
    - 大人の事情により（GAEのページを読んでね）CommentDtoのメンバはpublicじゃないので、JSPでアクセスしやすいように入れ替えます。

```
List<String> commentStrList = new ArrayList<String>();
for (CommentDto commentObj : commentList) {
  commentStrList.add(commentObj.getComment());
}
```

  - リクエストの属性に設定する

```
context.getRequest().setAttribute(“list”, commentStrList);
```

(10)ストアした情報を取得/表示する [2]

- JSPに表示する
  - <c:forEach>とか使って表示する

```
<c:forEach items="${list}" var="comment">
  <div>${comment}</div>
</c:forEach>
```

  - <hr />とか使って見やすくしてみる
  - サニタイジングもした方が良いですね

(11)とうとうGAEにデプロイする [1]

- Eclipseを使ってデプロイする
  - Projectを右クリックして「Google」->「Deploy to App Engine」を選択
  - 「App Engine project settings」でアプリケーションIDとかを設定する
    - アプリケーションIDは「mobytter」にする
  - 出来たらパスワードを入力して、「配置」ボタンをクリック！
  - 上手く行っていれば以下のURLで動くはず！

    http://mobytter.appspot.com/top/

(11)とうとうGAEにデプロイする [2]

- QRコードで簡単アクセス

近距離の方はこちら

(12)おめでとうございます！

- というわけで
  - 30分程度で小さな携帯サイトが作れました！
  - 是非みなさんも試してみてください！

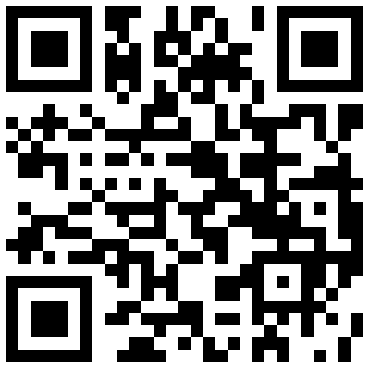
番外編 [1] Mailを使った投稿

- MailBoxerサービスを使う
  - 株式会社レイハウオリのASPでメールの内容からURLをキックしてくれるサービスがあります。

    http://mailboxer.jp/
  - これが使えるように新しいPageクラスを作ってみる。（MailBoxerPage）
  - デプロイして実験する

番外編 [1] Mailを使った投稿

- QRコードで簡単アクセス



近距離の方はこちら

番外編 [2] mobytterロゴのサイズ調整

- mobytterのロゴを使ってみる
  - しかし、携帯のブラウザサイズはまちまちなので、解像度の高いブラウザだと「ちっっっっちゃな」ロゴになっちゃう。
  - mobyletのブラウザサイズ別画像サイズ出し分け機能を使ってオートリサイズして出力してみる。
    - （注意）最近GAE/JのImage APIの実装が変わった際に、リサイズ後の画像はPCブラウザでは表示出来るが、docomo/SoftBankで表示出来ないケースが発生しています。

以上

ご清聴ありがとうございました。

＜ご連絡はこちら＞
s.takeuchi@leihauoli.com
http://mobylet.sandbox.seasar.org/
http://www.leihauoli.com/