Building Web Applications With the Leading Java Framework

Cubby
IN ACTION

アジェンダ

- Cubby in "real" service
  - 実サービスでの Cubby の利用例
- Here Comes Cubby 2.0 !!
  - Cubby 2.0 での新機能紹介

自己紹介

- 名前：染田貴志
  - http://d.hatena.ne.jp/tksmd/
- 所属：株式会社チョイスタジオ
  - http://www.choistudio.jp/

自己紹介

- 名前：馬場保幸
  - http://d.hatena.ne.jp/y-baba/
- 所属：株式会社ヌーラボ
  - http://www.nulab.co.jp/

Cubby in “real” Service

実稼働システムで
カビーを使ってみた！

「ちょい」っと学べる
choistudy
ちょいスタディ

Cubby in "real" service

- Cubby の特徴
- 使い方のポイント
- 前半部まとめ

前半は 1.1 を前提でお話します

Cubby の特徴

- Webアプリケーションフレームワーク
  - シンプル＆スモール
  - CoolなURIをサポート
  - JSP 2.0 Love!

Cubby の特徴

- 基本 Struts スタイル
- 「必要最低限」求められる機能
  - URL と記述したコードの結びつけ
  - リクエストパラメータのバインディング
  - 入力値のバリデーションの仕組み
  - ビューにおける値の出力補助

Cubby の特徴

- Cubby がしないこと
  - Ajax な Javascript コードの動的生成
  - Request/Session 以外のスコープ管理
  - PDF や Excel などリッチな出力方式
  - など

Cubby の特徴

- CoolURI

```
@Path("todo")
public class ToDoAction extends Action{
    // /todo/new
    public ActionResult new(){...}

    public Integer id;
    // /todo/100
    @Path("{id,[0-9]+}")
    public ActionResult index(){}
}
```

使い方のポイント

- Action の設計
- 基本コンポーネントの拡張
  - Validator を拡張
  - ActionResult を拡張
- 秘密の機能「コンバータ」

使い方のポイント

- Action の設計
  - デフォルトの Actionはほぼ何もしない
  - 汎用的な機能を持つ基底クラスを設計する
- Choistudy での例
  - AbstractAction
    - AbstractBasicAction
      - AbstractPagerAction
      - AbstractMobileAction
      - etc

Action は拡張して使うがよろし

使い方のポイント

```
public abstract class AbstractPagerAction
<T extends BasicSearchFormDto>
extends                                    AbstractBasic
Action {

    // オフセットが指定されてきた場合のページの処理
    public ActionResult page() {...}

    // ソートキーが指定されてきた場合のページの処理
    public ActionResult sort() {...}
```

```
public class DrillAction extends
AbstractPagerAction<DrillSearchFormDto>{
    // /drill/page?offset=...
    // /drill/sort?key=...
}
```

拡張するだけで基本 URL が作成

使い方のポイント

- 基本コンポーネントの拡張
  - Validatorの拡張
  - 汎用バリデーション
  - Choistudy での利用例
    - ReferedValidator
    - MultiRecordEmailValidator

使い方のポイント

- 汎用バリデーション

```
public abstract class BaseValidationRules extends
DefaultValidationRules {
    private static final RequiredValidator
REQUIRED_VALIDATOR = new RequiredValidator();
    protected Validator required() {
        return BaseValidationRules.REQUIRED_VALIDATOR;
    }

    public ActionResult fail(String errorPage) {
        HttpServletRequest req =
ThreadContext.getRequest();
        if ("XMLHttpRequest".equals(req.getHeader("X-
Requested-With"))) {...
            return new Json(result);
        }
        return super.fail(errorPage);
    }
}
```

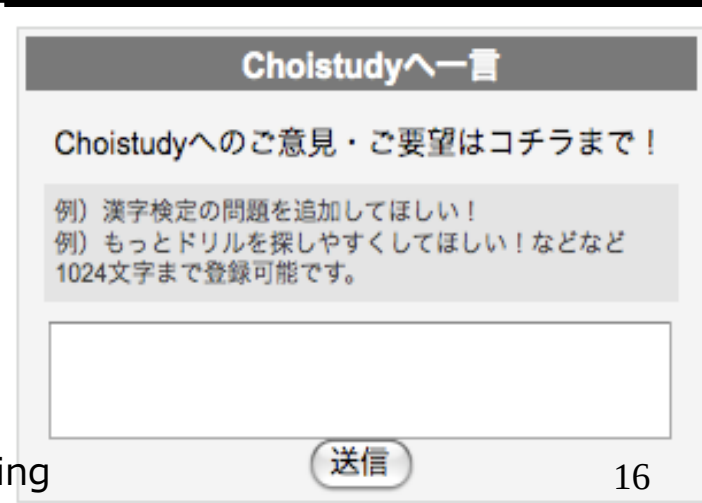
使い方のポイント

- Referer に戻るバリデーション

```
public class RefereredValidationRules extends
DefaultValidationRules {

    @Override
    public ActionResult fail(String errorPage) {
        String referer =
ChoistudyUtil.getInternalRefererPath();
        if (referer != null) {
            return new Redirect(referer);
        }
        return super.fail(errorPage);
    }
}
```

どのページでも表示されるフォーム

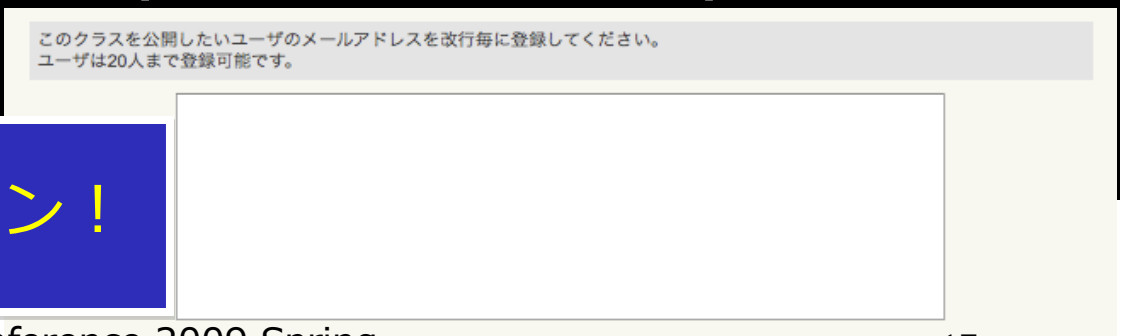


使い方のポイント

- MultirecordEmailValidator

```
public class MultiRecordEmailValidator implements
ScalarFieldValidator {

    public void validate(ValidationContext context, Object
value) {
        List<String> list =
ChoistudyUtil.splitUniq((String) value);
        for (String address : list) {
            String message = Messages.getText("valid.email",
address);
            EmailValidator validator = new
EmailValidator(message);
            validator.validate(context, address);
        }
    }
```



改行区切りで複数バリデーション！

使い方のポイント

- 基本コンポーネントの拡張 (続)
  - ActionResult を拡張する
    - テンプレート (velocity,free marker)
  - Choistudy での利用例
    - JAXB 2.0 (Java6) を使った XML
    - MobileRedirect

使い方のポイント

- JAXB 2.0 を使ったXML

```
public class Xml implements ActionResult {
    public void execute(...) throws Exception {
        JAXBContext ctx =
JAXBContext.newInstance(bean.getClass());
        Marshaller marshaller = ctx.createMarshaller();
        response.setCharacterEncoding(this.encoding);
        response.setContentType(...);
        final Writer writer = response.getWriter();
        marshaller.marshal(bean, writer);
        writer.flush();
    }
}
```

JSON/XML出力スタイルの同一化

```
public class QuestionAction extends Action{
    public ActionResult xml(){
        QuestionDto dto = ...;
        return new Xml(dto);
    }
}
```

使い方のポイント

- 秘密の機能コンバータ
  - RequestParameter のエンティティの変換の仕組
    - **/tksmd/entries/123**
      - tksmd => Account
      - 123 => Entry
  - Choistudy での利用例
    - QuestionConverter

エンティティとパスのマッピングの効率化

使い方のポイント

```
public class QuestionDtoConverter extends AbstractConverter
{
    public QuestionService questionService;

    public Object convertToObject(Object value,...) {
        Integer questionId =
Integer.parseInt(value.toString()));
        return questionService.find(questionId);
    }
:
}
```

```
public class QuestionAction extends Action {
    public QuestionDto questionDto;
    @Path({questionDto,[0-9]+})
    public ActionResult detauk() {
        return new Json(questionDto);
    }
}
```

前半まとめ

Cubby != フルスタックフレームワーク

作りたいものとの間にギャップが。。。

前半まとめ

o あえていえば

RESTfulWeb 時代において
Struts を最解釈したフレームワーク

Here Comes Cubby 2.0!

カビー2.0がやってきた！

ヤァ!ヤァ!ヤァ!

New Feature

- Spring / Guice Integration
  - Seasar2 に加えて Spring Framework / Google Guice との統合
- POJO Action
  - POJO によるアクションクラス
- JUnit4
  - JUnit4 形式の単体テスト
- Validation with Oval
  - Oval によるアノテーションベースのバリデーション
- Improve of type conversion
  - リクエストパラメータ変換のためのコンバータを指定
  - バリデーション時に型変換のエラーを検出

Spring / Guice Integration

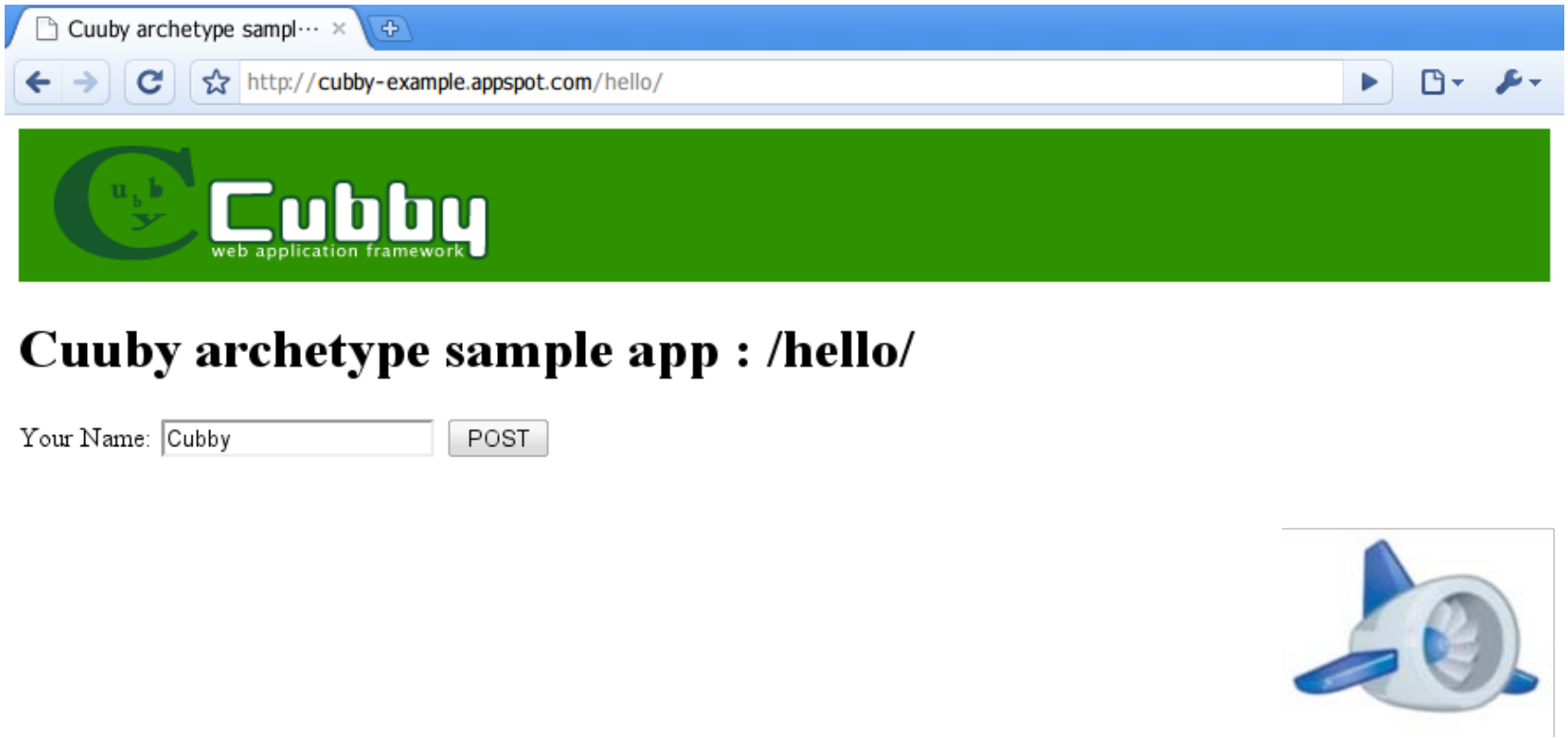
Cubby ～1.1

Seasar 2.4 との組み合わせのみ

Cubby 2.0

- プラグイン機構の導入によって、コンテナ部分の差し替えが可能に。
  - Spring Framework 2.5+
  - Google Guice 2.0+
  - Seasar 2.4+
- Maven Archetype も用意しているのでプロジェクトの作成が簡単。

Cubby + Guice on GAE/J



site: http://cubby-example.appspot.com/
source: https://www.seasar.org/svn/cubby/branches/2.0.x/cubby-apps/cubby-example-gae/

POJO Action

```
public class HelloAction {

    @Inject
    private FlashMap flash;

    @Inject
    private ActionErrors errors;

    public ActionResult index() {
        return new Forward("index.jsp");
    }

}
```

JUnit4

```
@Test
public void index1() throws Exception {
  MockServletContext ctx = new MockServletContext();
  ctx.addInitParameter(
    MODULE_INIT_PARAM_NAME,
    ApplicationModule.class.getName());
  MockHttpServletRequest req
    = new MockHttpServletRequest();
  req.setMethod("GET");
  req.setServletPath("/hello/");
  MockHttpServletResponse res
    = new MockHttpServletResponse();
  ActionResult result
    = CubbyRunner.processAction(
      ctx, req, res, new GuiceFilter());
  CubbyAssert.assertPathEquals(
    Forward.class, "index.jsp", result);
}
```

Validation with Oval

```
@RequestParameter public String text;
@RequestParameter public String memo;
```

```
public void initialize() {
    add("text", new RequiredValidator(), new
MaxLengthValidator(10));
    add("memo", new MaxLengthValidator(100));
}
```

1.1

```
@RequestParameter @NotNull @NotEmpty @MaxLength(10)
public String text;
@RequestParameter @MaxLength(100)
public String memo;
```

```
public void initialize() {
    add(new OvalValidationRule());
}
```

2.0

Improvement of type conversion

1. バリデーション時に型変換のエラーを検出
   - バリデータの記述がシンプルになります。
2. @RequestParameter にコンバータを指定
   - 入力されたデータをトリムするなど、入力値の加工に使えます。

Improvement of type conversion

1. バリデーション時に型変換のエラーを検出

```
@RequestParameter public int age;
@RequestParameter public Date startDate;
```

```
public void initialize() {
    add("age", new NumberValidator());
    add("startDate", new DateFormatValidator());
}
```

1.1

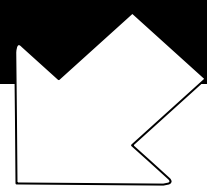
```
public void initialize() {
    add(new ConversionValidationRule());
}
```

2.0

Improvement of type conversion

2. @RequestParameter にコンバータを指定

```
@RequestParameter(converter = TrimConverter.class)
public String address;
```



```
public class TrimConverter extends AbstractConverter {
    public Object convertToObject(Object value,
Class<?> objectType, ConversionHelper helper) throws
ConversionException {
        if (value == null) {
            return null;
        }
        return value.toString().trim();
}
```

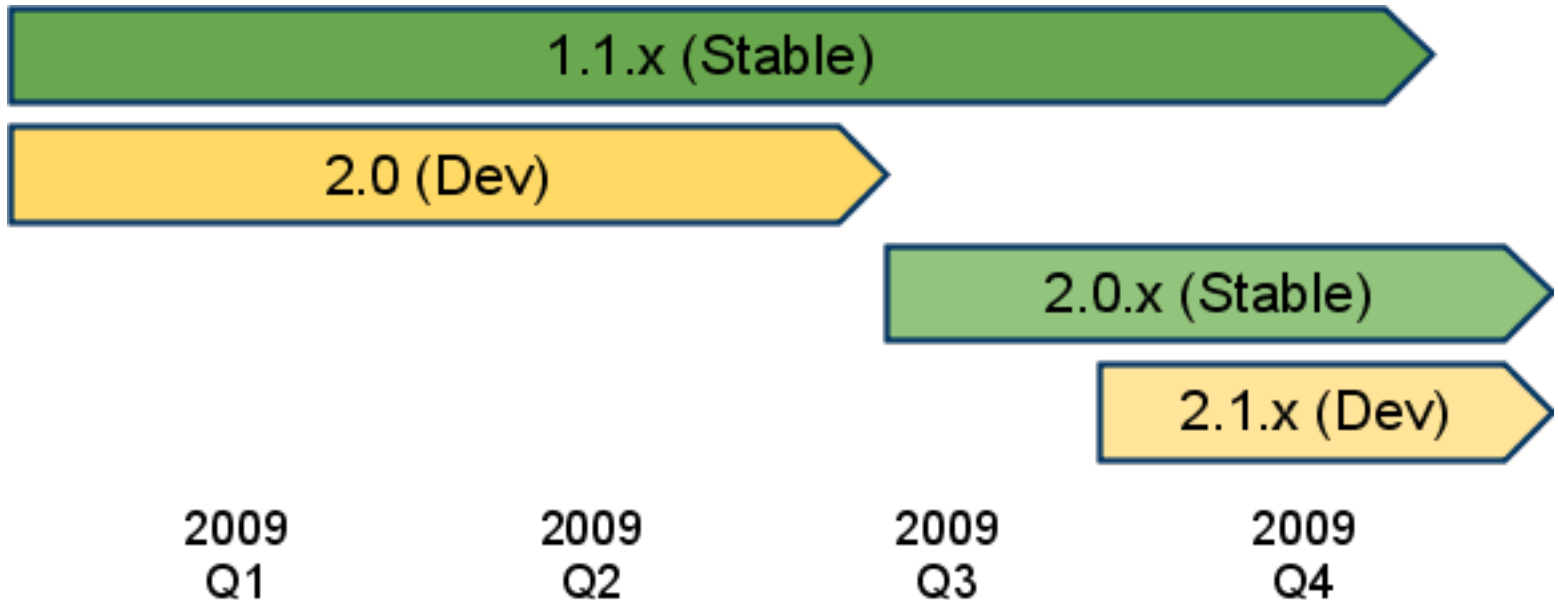
Plugin Mechanism

- プラグインを使うには
  - jar をクラスパスに追加するだけ
- どんなプラグインがある？
  - S2Container プラグイン cubby-s2.jar
  - Guice プラグイン cubby-guice.jar
  - Spring プラグイン cubby-spring.jar
  - Gson プラグイン cubby-gson.jar
  - Oval プラグイン cubby-oval.jar

Plugin Mechanism

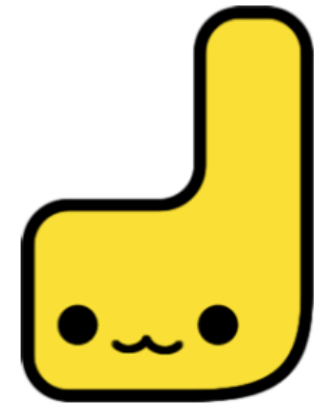
- プラグインを作るには
  - Plugin インターフェイスを実装する
    - 拡張ポイント
      - リクエストの処理
      - アクションの実行
      - アクションリザルトの実行
      - サービスプロバイダの取得
  - META-INF/service/org.seasar.cubby.plugin.Plugin にそのクラスの FQCN を書く

Roadmap



2009 Q2 Cubby 2.0.0 リリース
2009 Q3～Q4 Cubby 2.1 開発開始

Stable : バグフィックス
Dev : 機能追加



ご清聴ありがとうございました

カビじゃない
綺麗に作れる
それCubby

米林一茶