maven

CubbyとMavenを使った開発のまとめスレ

株式会社ヌーラボ　縣俊貴

自己紹介

- 縣俊貴(はてなID:agt)
- 株式会社ヌーラボ
  - SI - アジャイル開発のヌーラボ
  - SaaSサービス
    - どこでもプロジェクト管理のBacklog
  - 社員募集中詳しくはWebで！

最近の活動

- OSS
  - Seasar Project
  - Cubby/S2Pager/S2XML-RPC
- 執筆
  - WEB+DB PRESS/gihyo.jp
  - 連載：良いコードへの道
- tenjin.web@福岡
  - Webクライアント技術勉強会
  - #3 State,#4 OOP, #5 MVC

アジェンダ

1. Cubbyって何？
2. Cubby 2.0の紹介
3. CubbyとMavenを使った開発のTips集

Cubbyって何？

What's Cubby?

- Webアプリケーションフレームワーク
- シンプル＆スモール
  - カスタムタグ7個（メインは４つ）
  - アノテーション9個(メインは３つ)
  - 6,390行、JAR 197K(1.1.14)
- CoolなURIをサポート
  - @Path("/todo/{id,0-9}+")
- JSP 2.0 Love!
  - JSPはロストテクノロジー化

いろいろと
おもしろいWebのサービスに
使われはじめています。

Newsgraphy

http://newsgraphy.com/

http://eatspot.jp/

「ちょい」っと学べる choistudy ちょいスタディ

http://www.choistudy.jp/

TopHatenar

http://tophatenar.com/

Hatenar Maps

http://hatenarmaps.com/

いろいろと
おもしろいWebのサービスに
使われはじめています。

S2Swingの浜本さんが
開発されたサービス

Newsgrapl

http://newsgraphy.com/

http://eatspot.jp/

choistudy
ちょいスタディ

http://www.choistudy.jp/

TopHatenar

http://tophatenar.com/

Hatenar
Maps

http://hatenarmaps.com/

CubbyとStruts(1.3)との違い

脳内変換不要カスタムタグ

Cubby

```
<t:input type="text"
  name="userId" />
```

Struts

```
<h:text
  property="userId"/>
```

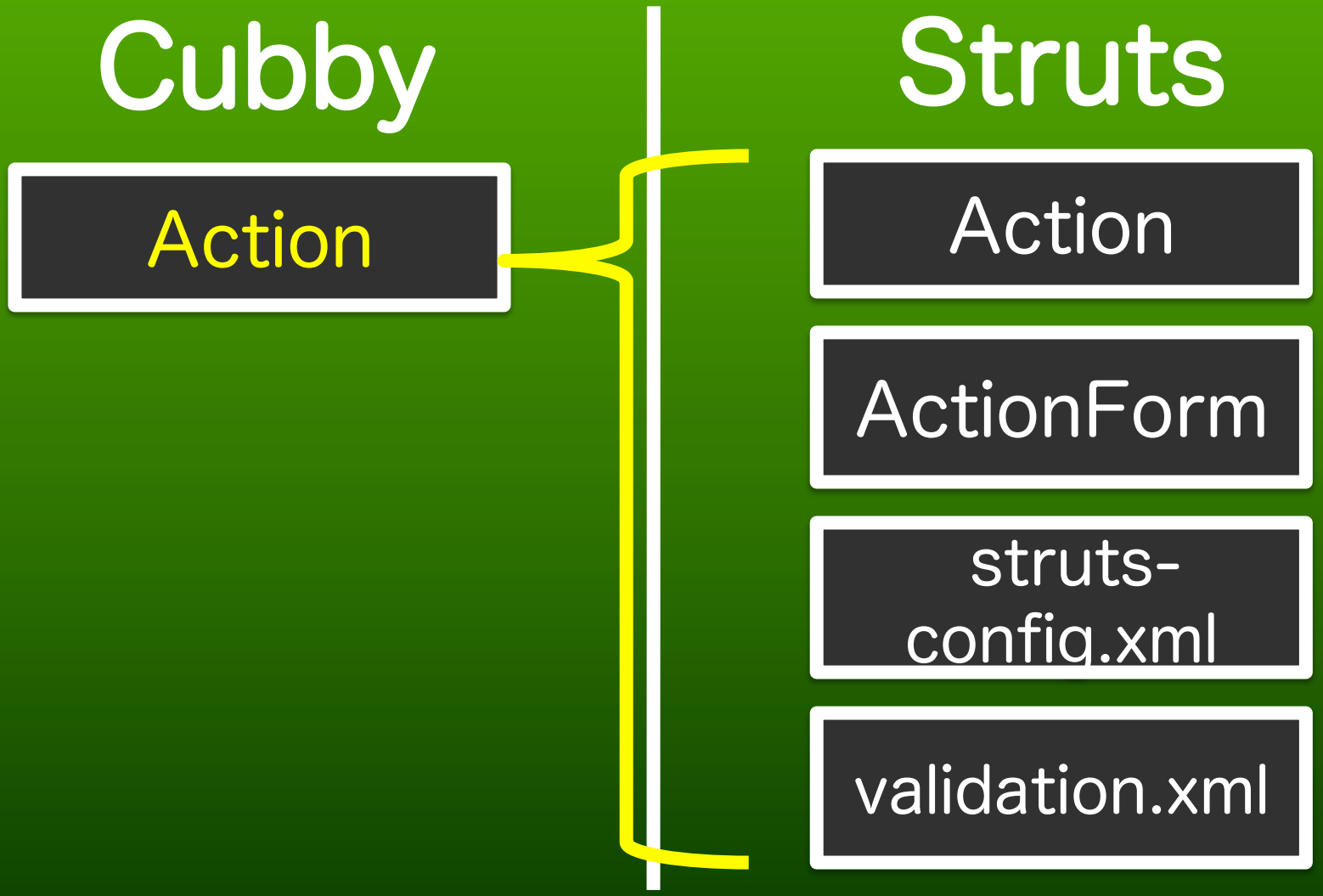
HTML

```
<input type="text" name="userId" />
```

Dynamic Attribute

```
<t:input type="text" name="userId" empId="1"/>
```

設定ファイルレス


Cubby
Action
Struts
Action
ActionForm
struts-config.xml
validation.xml

Actionクラス

```
// URIは「/login/process」
public class LoginAction extends Action {
  @RequestParameter
  public String userId;
  @RequestParameter
  public String password;
  public ActionResult process() {
    System.out.println(
       userId + “/” + password)
    reutrn new Forward(“/home.jsp”);
  }
}
```

ActionFormに相当

struts-config.xmlに相当

Actionクラス

```
public class LoginAction extends Action {
  public ValidationRules rules =
  new DefaultValidationRules() {
    public void initalize() {
     add(“userId”, new RequiredValidator());
     add(“password”, new RequiredValidator());
    }
  }
  @Validation(rules=“rules”,errorPage=“login.jsp”)
  public ActionResult process() { … }
}
```

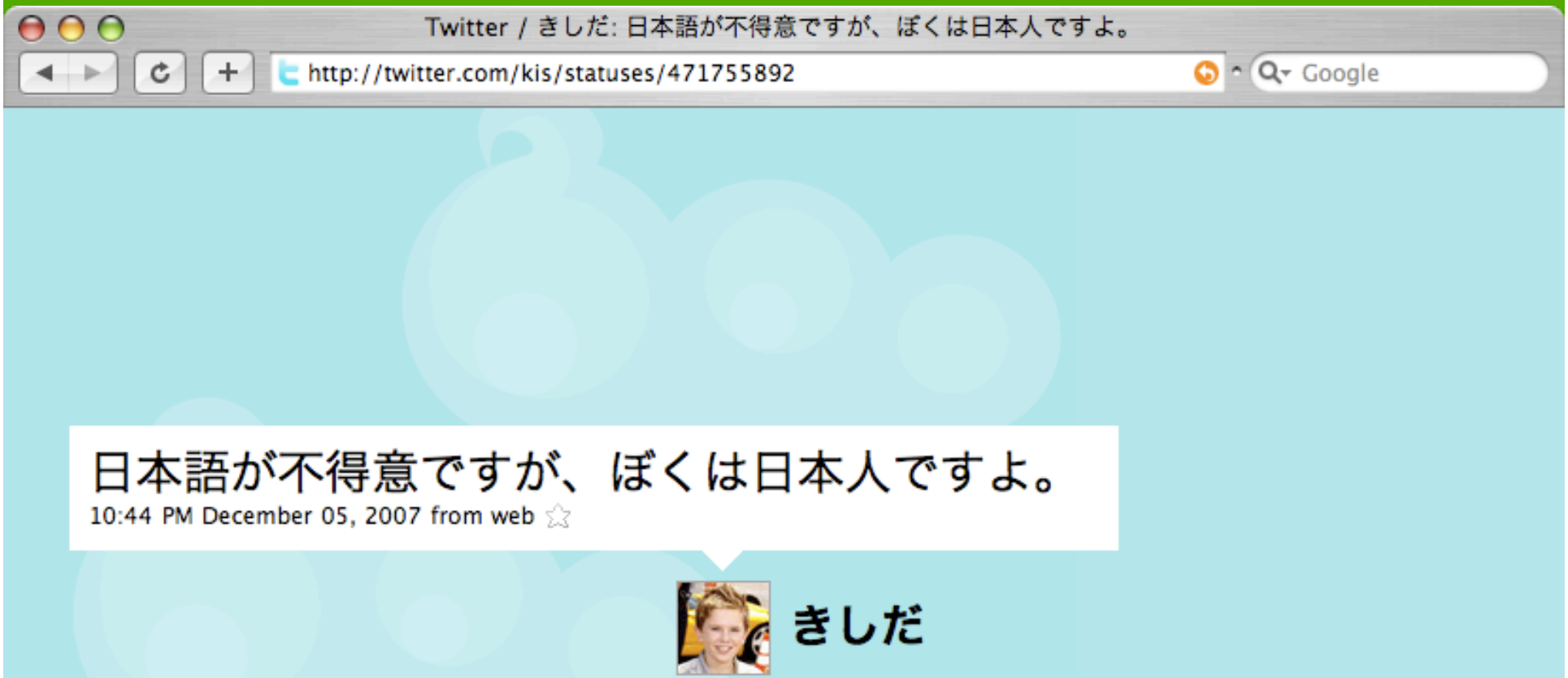
validation.xml に相当

struts-config.xmlに相当

@Path

```
@Path(“todo”)  // TodoActionの場合省略可
public class TodoAction extends Action {
  // /todo/new
  public ActionResult new() { … }
  // /todo/save
  @Path(”save")
  public ActionResult post() { … }
}
```

http://twitter.com/kis/statuses/471755892



@Path（パステンプレート）

```
@Path("todo")  // TodoActionの場合省略可
public class TodoAction extends Action {
  public String id;
  // /todo/{id}
  @Path("{id}")
  public ActionResult index() {
    System.out.println(id);
  }
}
```

正規表現を使った柔軟なURI指定

@Path(”/todo/{id}”) -> [a-z][A-Z][0-9]+
「/todo/0fabd3f」にマッチ

@Path(”/todo/{id,[0-9]+}”)
「/todo/10001」にマッチ

JSR-311の仕様もだいたい同じ！

Path(”/icon/{width,[0-9]+}x
{height, [0-9]+}.{ext,png|jpg}”)
「/icon/100x200.png」にマッチ

Cubby 2.0の紹介

Cubby 2.0

- Cubby 1.0 2008年２月
  - 最初の正式リリース版
- Cubby 1.1 2008年8月
  - 実用的な機能を追加
- Cubby 2.0 2009年4月
  - 本日2.0-beta1をリリース！
  - http://cubby.seasar.org/2
  - メインコミッタはBABAさん

Cubby2 – DIコンテナ非依存

- Seasar2 ○
- Guice ○
- Spring△
- EJB△
- Slim△

○は現時点でサポート済み
△は今後サポート予定

Cubby+Guiceの例

```
public class ExampleModule extends AbstractModule {
  @Override
  protected void configure() {
    install(new AbstractCubbyModule() {
      @Override
      protected PathResolver getPathResolver() {
        final PathResolver pathResolver = new PathResolverImpl();
        pathResolver.add(IndexAction.class);
        pathResolver.add(HelloAction.class);
        return pathResolver;
        }
      });
      install(new ServletModule());
      bind(HelloService.class).to(HelloServiceImpl.class).in(Singleton.class);
    }
  }
}
```

Guiceの流儀に
合わせた設定

Cubby+Guiceの例

```
@RequestScoped
public class HelloAction extends Action {
    …
    @Inject
    private HelloService helloService;
    private String name;
    …
    public ValidationRules getValidation() {
        return validation;
    }
    @RequestParameter
    public void setName(String name) {
        this.name = name;
    }
   @Validation(rules = "validation", errorPage = "index.jsp")
    public ActionResult message() {
        this.message = this.name + " " + helloService.getMessage();
        return new Forward("hello.jsp");
    }
}
```

Guiceの流儀に
合わせた記述

Cubby2- 内部構造の変化

- 利用者からはわかりにくい変化
- 公開パッケージと非公開パッケージの明確化
  - org.seasar.cubby.internal.*
- モジューラブルな設計
  - org.seasar.cubby.spi.*
    - ActionHandlerChainProvider
    - BeanDescProvider
    - ContainerProvider
    - ConverterProvider
    - JsonProvider
    - PathResolverProvider
    - RequestParserProvider
- いろいろとリファクタリングしました

Java <-> JSON変換

- 特にJavaではオブジェクトに変換したい
  - JSONには日付・時刻型の規定がない
  - Serializer/Desrializerの挙動・機能の違い

JsonProvider

- JavaでJSON
  - Gson
  - JSON-lib
  - Jsonnic
  - JsonSerializer(Seasar2)
- JavaScriptでJSON
  - eval
  - json2.js
  - JSON(ECMA Script 3.1, FF 3.5)

Cubby2 – public field

- JavaEE5のUnifiedELで実現
  - Servlet2.5+JSP2.1
  - Tomcatだと6.0以上で利用可能
  - Intertypeと比較してパフォーマンスが向上
- 現在はS2を選択した場合のみ使用可能

CubbyとMaven2を使った開発のTips集

Cubbyで基底クラスの活用

- Cubbyのポリシー
  - フレームワークががんばりすぎない
  - アプリケーション作成者に相違・工夫はゆだねたい
- AbstractAction
  - ログインユーザ情報の取得
  - 頻出するDB情報の取得メソッド
- AbstractValidationRules
  - 良く使用するValidationを定義
  - 検証エラー発生時の共通的な挙動

AbstractValidationRules

```
public abstract class AbstractValidationRules extends DefaultValidationRules {
    private static final RequiredValidator REQUIRED_VALIDATOR = new
    RequiredValidator();
     …
    protected Validator required() {
        return BaseValidationRules.REQUIRED_VALIDATOR;
    }
    @Override
    public ActionResult fail(String errorPage) {
        HttpServletRequest request = ThreadContext.getRequest();
        if ("XMLHttpRequest".equals(request.getHeader("X-Requested-With"))) {
          Map<String, Object> result = new HashMap<String, Object>();
          result.put("status", "error");
          result.put("errors", action.getErrors());
          return new Json(result);
        }
        return super.fail(errorPage);
    }
}
```

ビューの部品化

- JSTLのインポートタグ(<c:import>)
- タグファイル(<tag:hoge>)
- カスタムタグ
  - SimpleTag
- インクルードディレクティブ (<%@ include %>)
- インクルードアクション (<jsp:include>)
- クライアントサイドテンプレート
  - jTemplates

Maven2のすすめ

- Maven2に乗っかることで・・・
  - ディレクトリ構成に悩まない
  - ビルド手順が統一される
  - 依存関係の管理ができる
    - pom.xmlを見ればOK
- 最初は苦労も（すごく）多いが飼いならせば得るものも大きい

プロジェクトの作成

```
# プロジェクトの雛形
mvn archetype:generate
    -DarchetypeCatalog=http://cubby.seasar.org
mvn cd {artifactId}

# Tomcatを単体で起動
mvn tomcat:run

# ブラウザで開く
open http://localhost:8080/{artifactId}
```

クラスパスの解決①

- Maven Eclipse Plugin
  - mvn eclipse:clean eclipse:eclipse
  - .project、.classpath、WTPの設定ファイルなどを作成
  - ソースアーカイブ、JavaDocのアタッチ設定付き
  - マルチモジュール構成の場合、親を取り込んでから子を個別にEclipseインポート

クラスパスの解決①

- はまりやすい
  - 依存ライブラリ追加時
    - WTPの設定ファイルがうまく更新されない
    - .settings/org.eclipse.wst.common.component
    - mvn eclipse:clean eclipse:eclipse

クラスパスの解決②

- m2eclipse
  - Eclipse Plugin
    - 0.9.8 - 最近安定してきた
  - マルチモジュールに対応
    - M2elipse -> Enable Nested Modules -> Update Project Configration
  - 時々重い
    - インデックスファイルの取得時とか
  - 将来的にはこれが完璧に動作するのが理想

リリースビルド

- 設定ファイルの変更
  - ログ出力設定の変更　DEBUG -> ERROR
  - JDBC接続先の変更
  - SMTPサーバの接続先
  - env.txtの書き換え ut -> 空文字
- 方法
  - 手作業・・・xだめ！ぜったいだめ！
  - resources-pluginのフィタリング
  - ant-runプラグイン
  - profileでリソースの置き換え

プロファイル

- 環境毎の設定やビルドプロセスを切り替える（フックする）仕組み
- CubbyのArchetypeプラグインではリリース用の設定(production)が設定済み
- -Pオプション
  mvn package -Pproduction

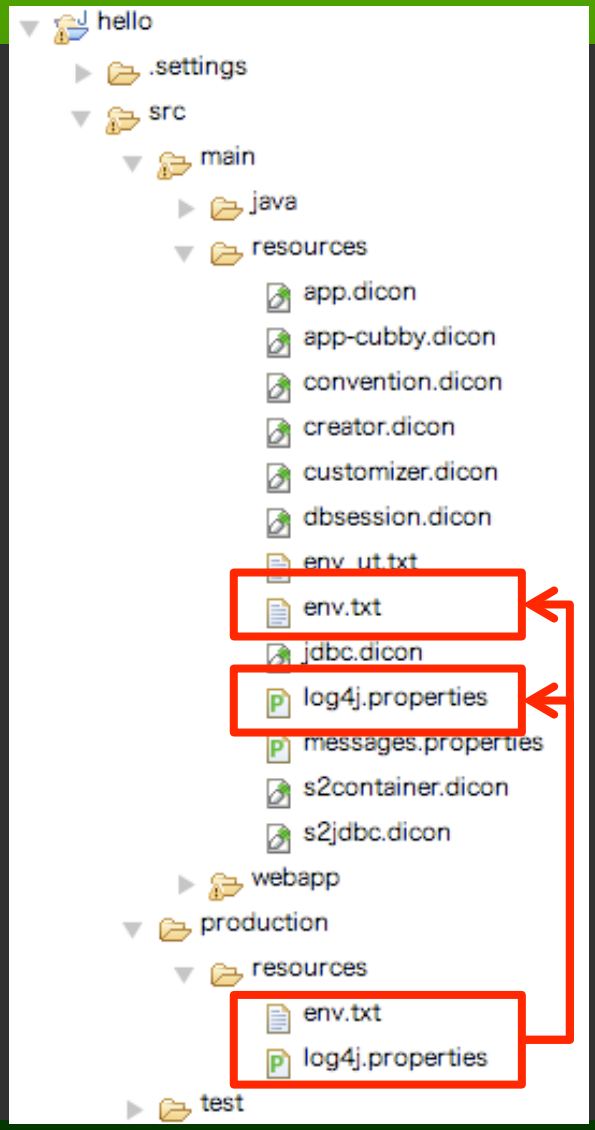
開発用
プロファイル

社内開発サーバ用
プロファイル

開発用
プロファイル

Profileでリソースの置き換え

```
<profiles>
 <profile>
    …
    <id>production</id>
    <activation>
      <activeByDefault>false</activeByDefault>
    </activation>
    <build>
      <resources>
        <resource>
          <directory>src/production/resources</directory>
        </resource>
        <resource>
          <directory>src/main/resources</directory>
        </resource>
      </resources>
    </build>
  </profile>
</profiles>
```



置き換えるリソースを最小化

config.properties

```
jdbc.url=jdbc:h2:tcp://localhost/~/cubbitter/data
smtp.host=jdbc:h2:tcp://localhost/~/cubbitter/data
```

settings.dicon

```
<components namespace=”settings">
 <component class=”java.util.Properties " name=” prop">
   <initMethod name="load”>
     <arg>@org.seasar.framework.util.ResourceUtil@getResourceAsStream(”settings.properties")</arg>
   </initMethod>
 </component>
</components>
```

jdbc.dicon

```
…
   <include path=”settings.dicon"/>
   <component name="xaDataSource"
         class="org.seasar.extension.dbcp.impl.XADataSourceImpl">
         <property name="driverClassName">settings. prop.getString(”jdbc.driverClassName")</property>
         <property name="URL">settings. prop.getString(”jdbc.URL")</property>
         <property name="user">settings. prop.getString(”jdbc.user")</property>
         <property name="password">settings.prop.getString(”jdbc.password")</property>
   </component>
```

その他Maven2トピック

- 社内リポジトリのススメ
- 外だしAntスクリプトと協力
- マルチモジュール構成はおすすめ
- CIにプロファイルを活用
- 続きはWebで
  - WEB+DB PRESS Vol.39
    構成管理　実践入門

まとめ

- CubbyはRESTfulなWebアプリケーション開発を助けるシンプルなフレームワークです。
  - クールなWebサービスから業務アプリまで。
- Cubby2.0は地味ながらも進化。他のコンテナもサポートしていきます。
- CubbyとMaven2の組み合わせで柔軟な開発、堅いリリースを実現しましょう。

ご清聴ありがとう
ございました。
ご質問があればどうぞ。