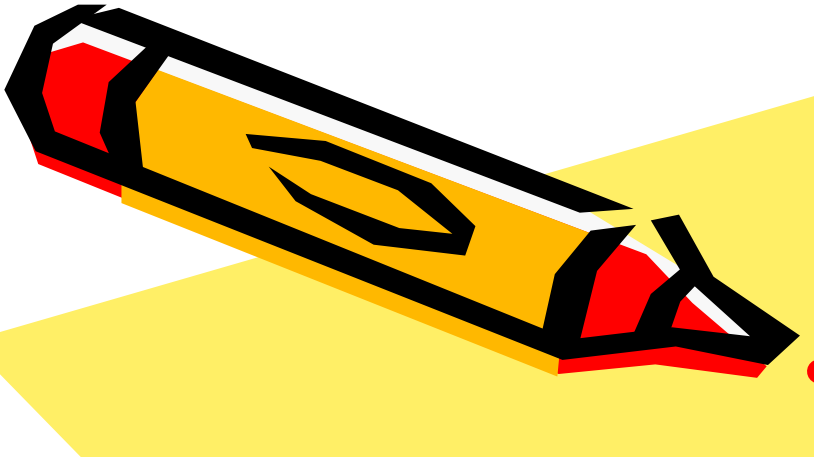
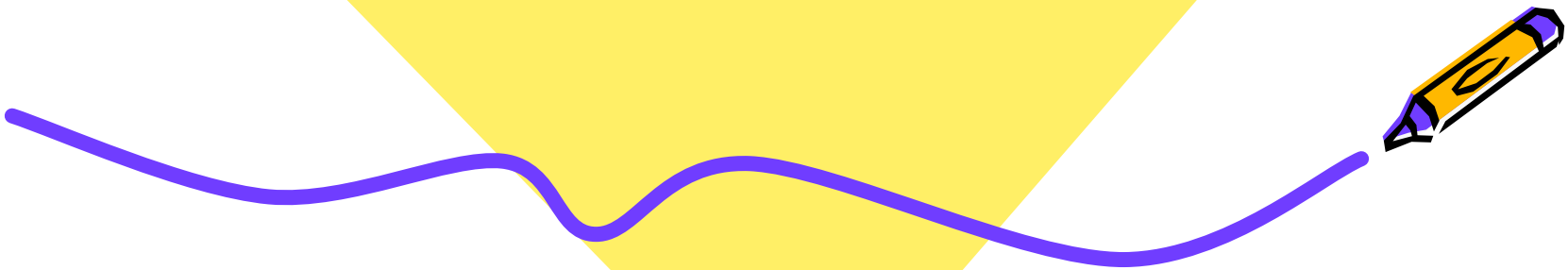


SpringとStruts連携



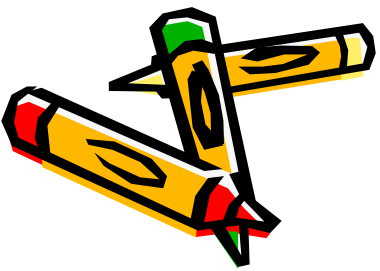
㈱トラストサービス　2006/05/27

DIコンテナ

前回ご説明したSpring→DIコンテナに共通することは、依存を注入することによってシステム内に存在するオブジェクト同士の結びつきを緩くすることであり、そのための仕組み提供を意味する。

[Spring のDIコンテナはBean定義ファイルに基づきJavaBeansの管理と構成を行う（Beanファクトリと、Beanファクトリの上に構成されるApplicationコンテキストによってDIコンテナは構成されている）]

Bean定義ファイル(デフォルト：applicationContext.xml)



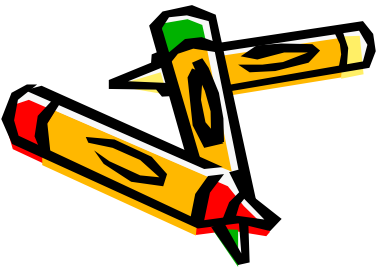
DIコンテナ

- またSpringは、

  Setter Injectionと呼ばれる手法(オブジェクト間の関連がXMLファイルに記述され、DIコンテナがオブジェクトのSetterメソッドを利用して参照するオブジェクトを設定すること)

  Constructor Injectionと呼ばれる手法(DIコンテナがオブジェクトのコンストラクタのパラメータとして参照するオブジェクトを設定する。参照するオブジェクト、参照されるオブジェクトの作成は開発者が行い、コンテナには関連付けだけを依頼する。

(Seasar2もSpringと同様の上記2つをサポートしている)



SpringはなぜWEBアプリケーション開発に必要なのか

システム構築を検討していく上で
開発者のためのアーキテクチャとして
開発効率・テスト精度・保守（デバッグ）や拡張（二次開発）のしやすい設計が必要である。

■開発効率
→　意図を把握しやすく、理解しやすい構造を設計(5000ページもあるドキュメントを読まないとわからないのはよくない)
→　テストが容易に行える構造を設計
(テストをするのにライブラリにクラスパスを通さなければならないとかWEBコンテナを用意したり、テストのために実装変更をするのはよくない)
■柔軟性
→保守しやすく、拡張しやすい構造を設計(システムに対するユーザ要求が変化しやすいため)
→将来の環境の変動に耐える頑健な構造を設計

SpringはなぜWEBアプリケーション開発に必要なのか

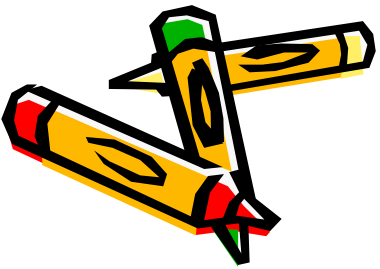
アスペクトの注入

Spring Aspect Injection を利用することでビジネス層で業務ロジックを実現するオブジェクトはフレームワークやコンテナに依存しないPOJOで作成することができる。

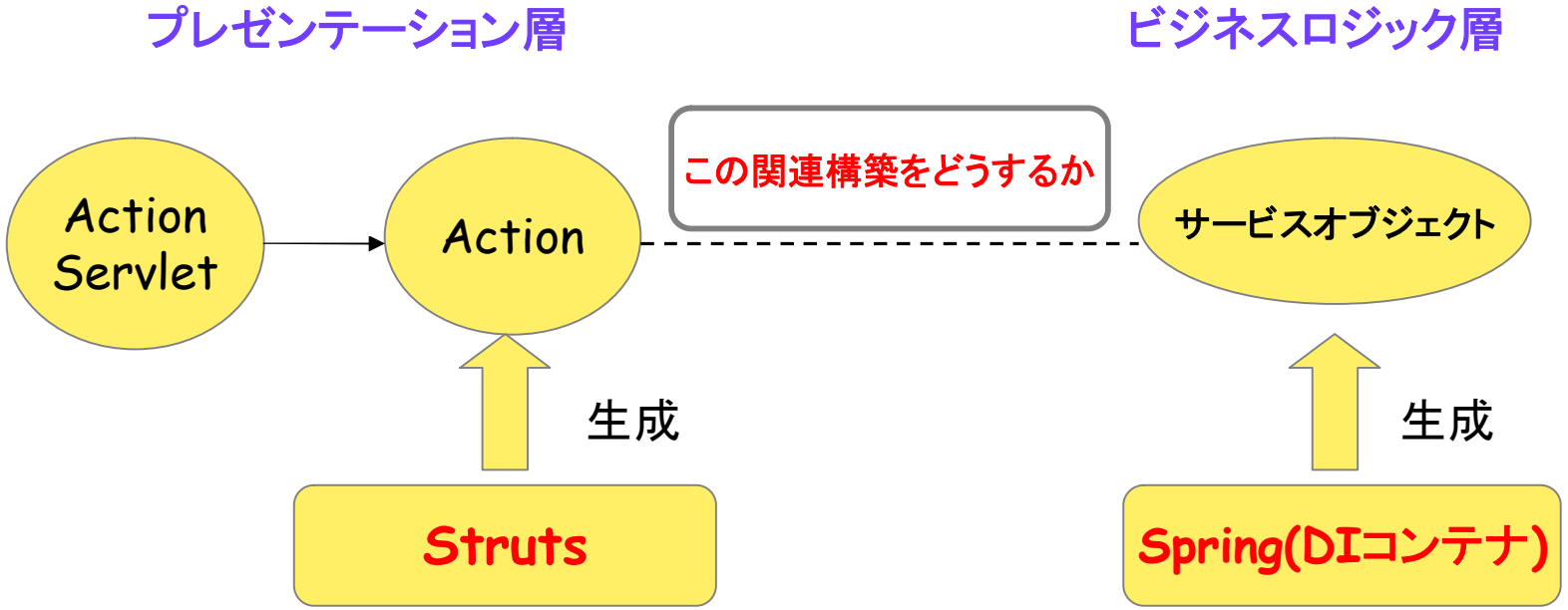
独立した業務ロジックを作れることはSpringの設計上の最大の魅力点だと思われる。

開発効率がよいまた、保守性や拡張性に強い設計が実現できると思われる

POJO :Plain Old Java Object の略語。「Pure JAVA、昔ながらのJAVA」

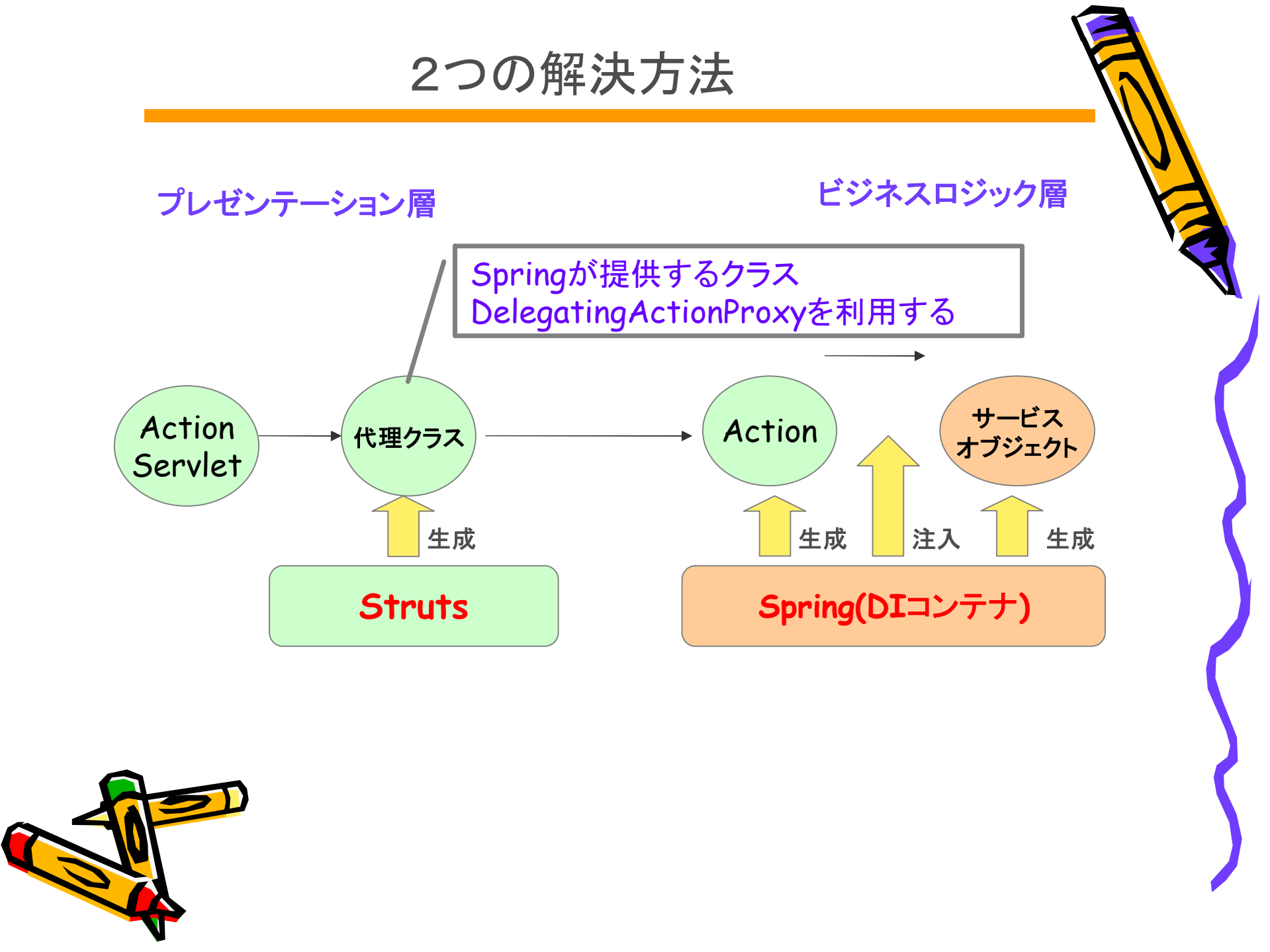


StrutsとSpringを連携させる場合の問題点

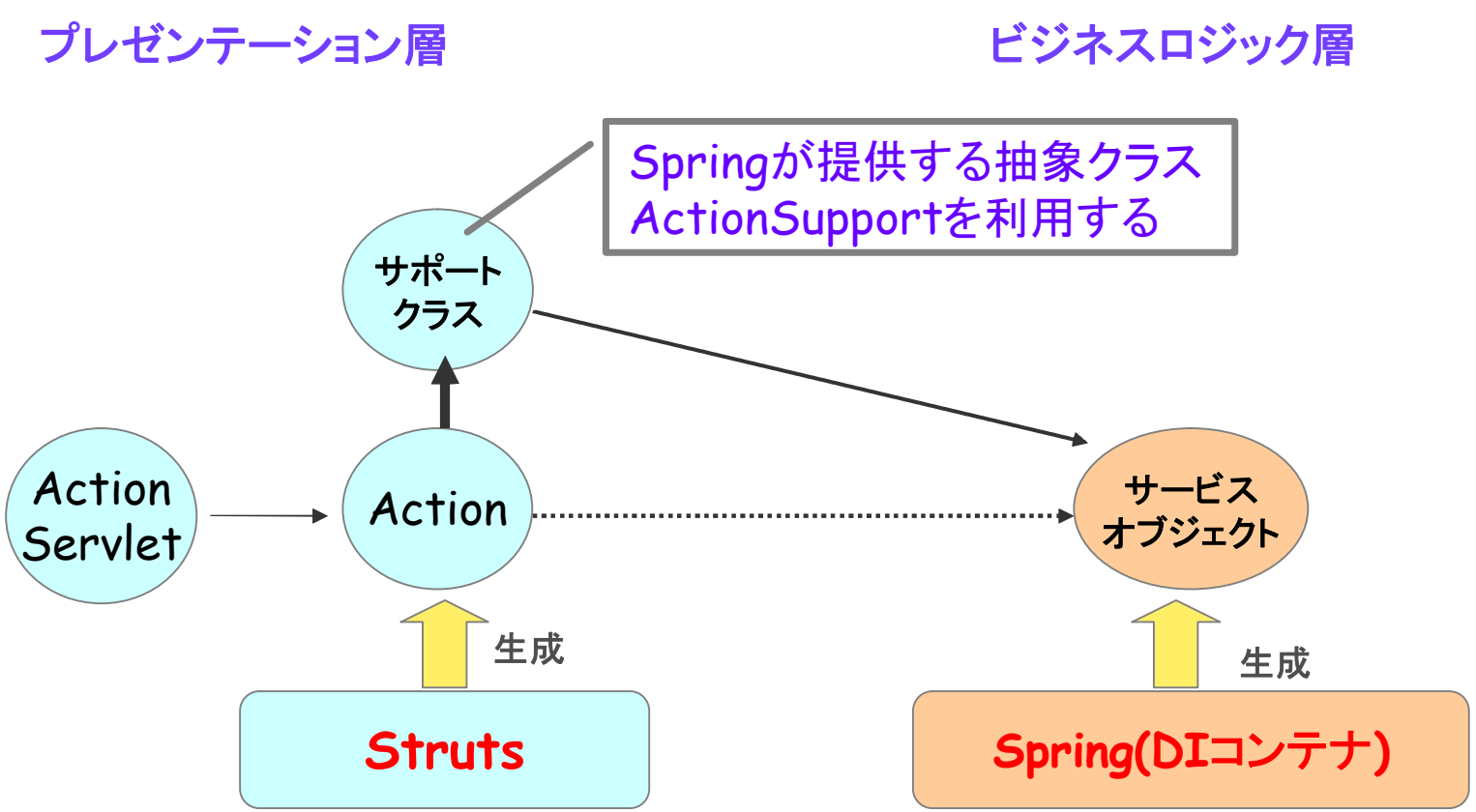


今回は説明しないが、【データアクセス層】が実装modelとしてある

2つの解決方法


プレゼンテーション層
ビジネスロジック層
Springが提供するクラス
DelegatingActionProxyを利用する
Action Servlet
代理クラス
Action
サービス オブジェクト
生成
生成
注入
生成
Struts
Spring(DIコンテナ)

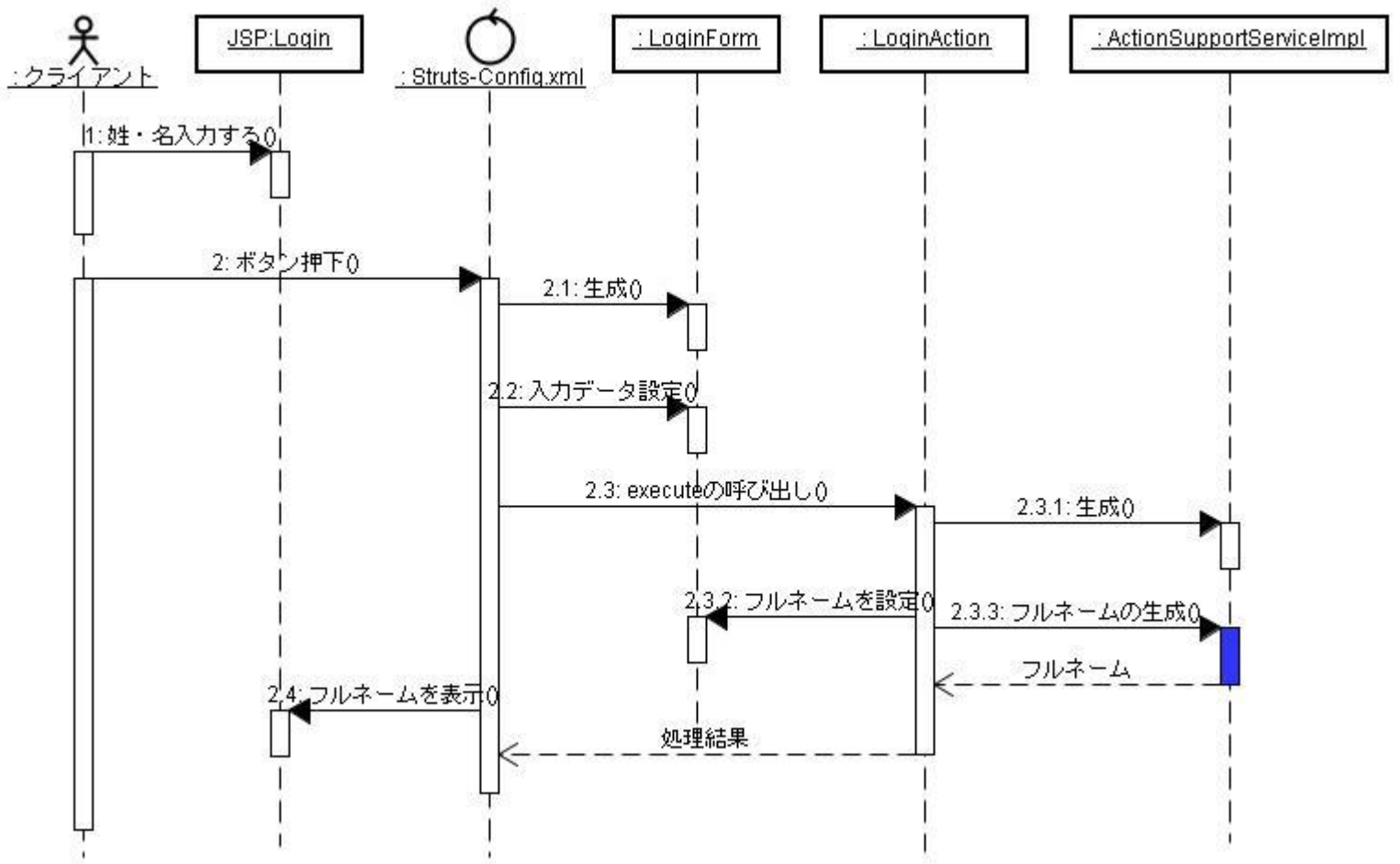
2つの解決方法



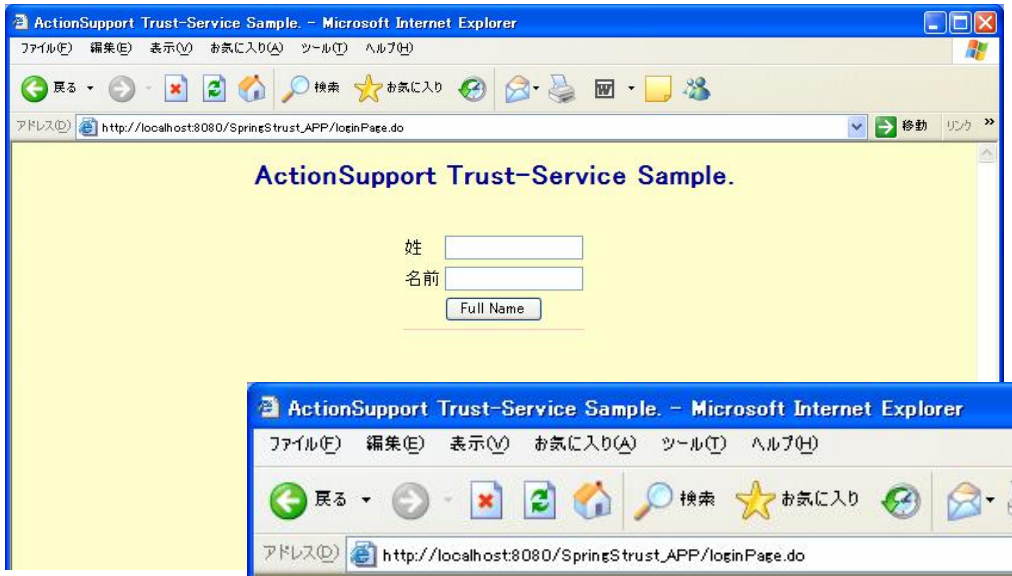
抽象クラスActionSupportはstrutsのActionを継承しているが、
Springと連携する際にActionをサポートするメソッドを幾つか
追加しているだけであることに注意したい。

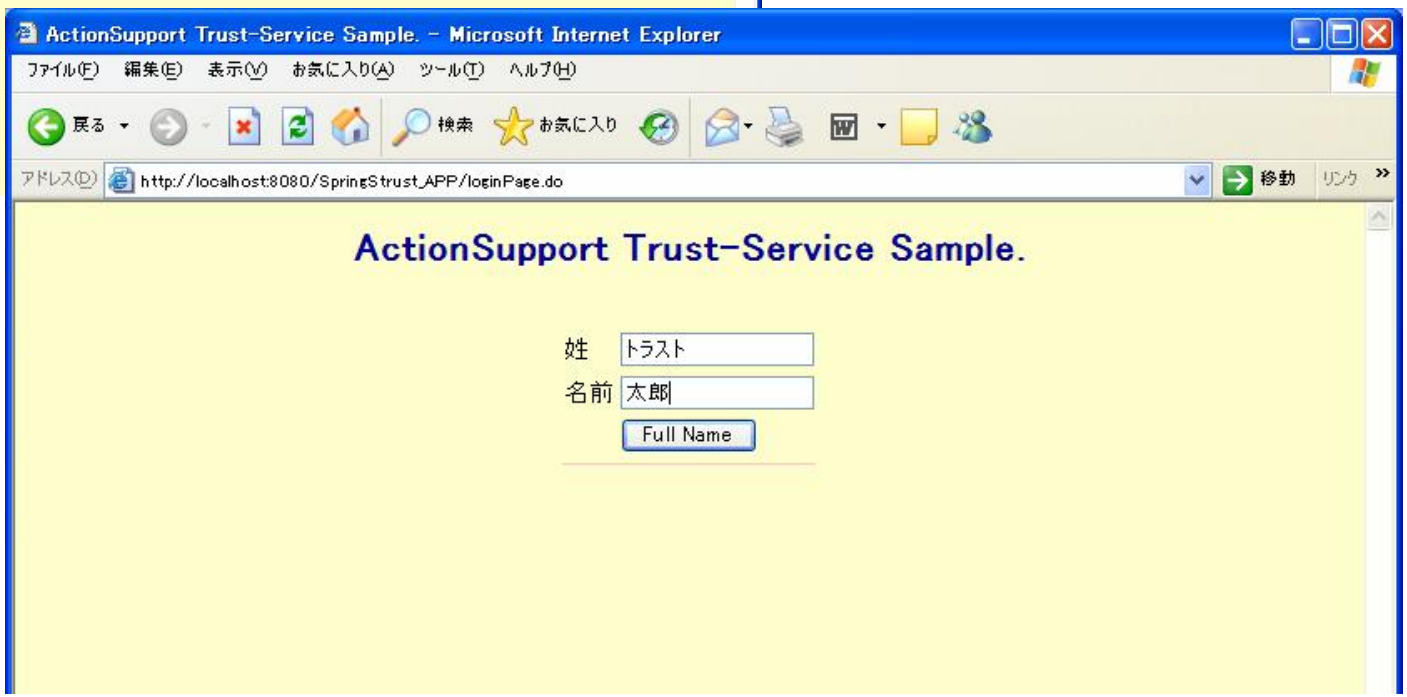
シーケンス図

入力項目:姓 ＋ 名前を 連結する

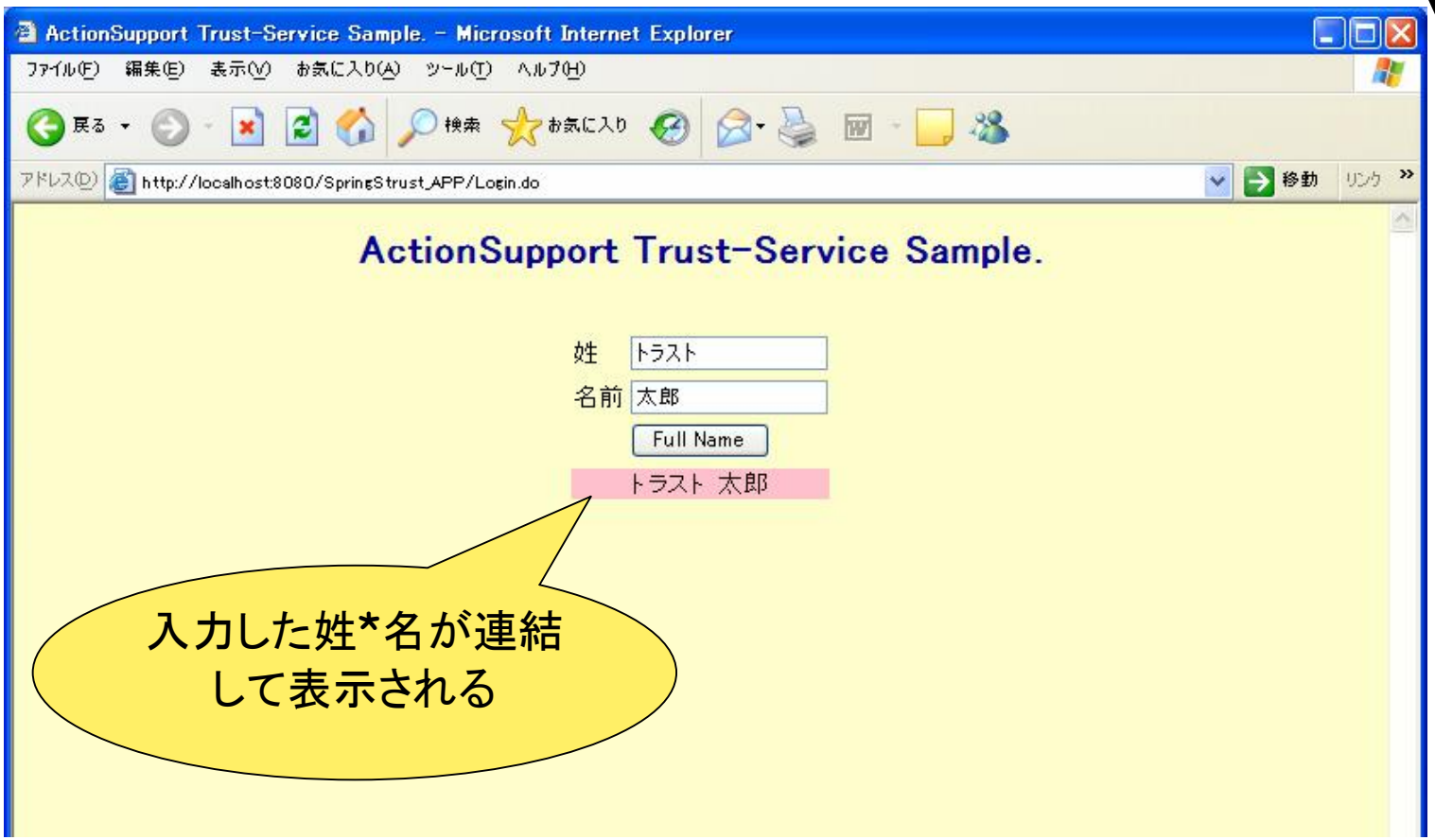


画 面 遷 移


ActionSupport Trust-Service Sample. - Microsoft Internet Explorer
ファイル(F) 編集(E) 表示(V) お気に入り(A) ツール(T) ヘルプ(H)
戻る 検索 お気に入り
アドレス(D) http://localhost:8080/SpringStrust_APP/loginPage.do
移動
ActionSupport Trust-Service Sample.
姓
名前
Full Name


ActionSupport Trust-Service Sample. - Microsoft Internet Explorer
ファイル(F) 編集(E) 表示(V) お気に入り(A) ツール(T) ヘルプ(H)
戻る 検索 お気に入り
アドレス(D) http://localhost:8080/SpringStrust_APP/loginPage.do
移動
ActionSupport Trust-Service Sample.
姓 トラスト
名前 太郎
Full Name

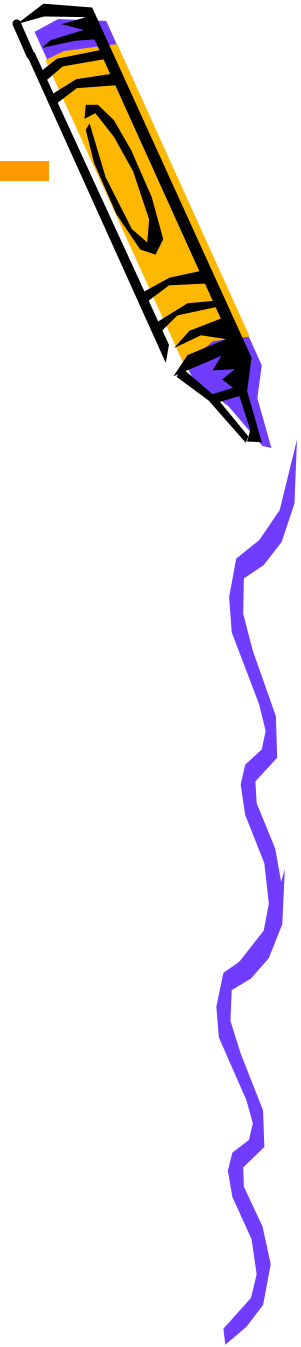
画 面 遷 移


ActionSupport Trust-Service Sample. - Microsoft Internet Explorer
ファイル(F) 編集(E) 表示(V) お気に入り(A) ツール(T) ヘルプ(H)
戻る 検索 お気に入り
アドレス(D) http://localhost:8080/SpringStrust_APP/Login.do
移動 リンク
ActionSupport Trust-Service Sample.
姓 トラスト
名前 太郎
Full Name
トラスト 太郎
入力した姓*名が連結
して表示される

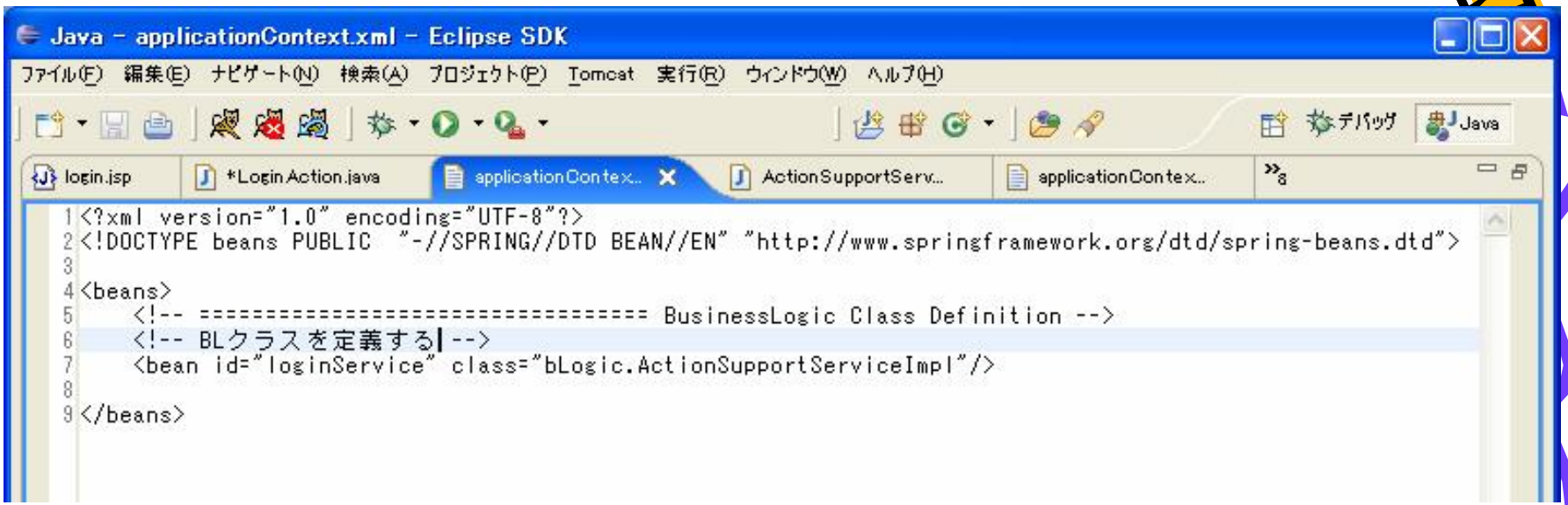
jsp

```
<%@ page language="java" pageEncoding="Windows-31J"
    contentType="text/html; charset=Windows-31J" %>

<html:form action="/Login" method="POST">
   <TABLE border="0">
    <TR><TD><bean:message key="label.firstName"/></TD>
      <TD><html:text property="firstName"/></TD> </TR>
    <TR><TD><bean:message key="label.lastName"/></TD>
      <TD><html:text property="lastName"/></TD></TR>
    <TR>  <TD colspan="2" align="center">
            <html:submit property="button">
               <bean:message key="welcome.button"/>
   </html:submit>
      </TD></TR>
    <TR><BR></TR>
    <TR><TD bgcolor="PINK" colspan="2" align="center">
        <bean:write name="loginForm" property="fullName"/>
   </TD>
    </TR>
   </TABLE>
</html:form>
```



ApplicationContext.xml



LoginActionクラス

LoginAction.java | ActionSupportServ... | applicationContex... | ActionSupportServ... | applicationContex...

```
1  package action;
2
3  import javax.servlet.http.HttpServletRequest;
13
14 /**
15  *
16  * @author k-mori
17  * @version 1.0
18  * @since 2006/05/10
19  */
20 public class LoginAction extends ActionSupport{
21
22      /**
23      * アクションクラス
24       * @param mapping
25       * @param form
26       * @param request
27       * @param response
28       * @return 画面遷移
29       */
30      public ActionForward execute(
31              ActionMapping mapping,
32              ActionForm form,
33              HttpServletRequest request,
34              HttpServletResponse response){
35
36             //Form取得
37             LoginForm fm = (LoginForm) form;
38             String firstName = (String) fm.getFirstName();
39             String lastName = (String) fm.getLastName();
40             //ビジネスロジック呼び出し(インターフェース)
41             /** getWebApplicationContext().getBean("loginService");
42              * AplicationContext.xml内のBEAN  id  を設定*/
43             ActionSupportService actionService
44                     = (ActionSupportService) getWebApplicationContext().getBean("loginService");
45             String fullName = actionService.createFullName(firstName, lastName);
46             fm.setFullName(fullName);
47              return mapping.findForward("success");
48         }
49 }
```

WebApplicationContext()からBeanを取得

ビジネスLogic


LoginAction.java
ActionSupportServ...
applicationContex...
ActionSupportServ...
package bLogic;
/**
* ビジネスロジック
* @author k-mori
* @version 1.0
* @since 2006/05/10
*/
public class ActionSupportServiceImpl implements ActionSupportService{
/**
*
*/
public String createFullName(String firstName, String lastName) {
return firstName + " " + lastName;
}
}

Struts基本処理フロー

・通常Actionクラスの実装

```
Public class SampleAction extends Action{
   ・・・(省略)・・・
    public ActionForward execute(
                    ActionMapping mapping,
                    ActionForm form,
                    HttpRequest requset,
                    HttpResponse response){

     ActionSupportService actionService =
                                new ActionSupportService();
     String fullName =
                     sampleService.createFullName(
                      firstName,lastName);
 ・・・(省略)・・・
    }
}
```

Actionクラスは通常ビジネス層のオブジェクトを生成してビジネスロジックを呼び出している。

Struts Spring基本処理フロー

・ActionSupportクラスの実装

```
Public class SampleAction extends SapportAction{
   ・・・(省略)・・・
    public ActionForward execute(
                        ActionMapping mapping,
                        ActionForm form,
                        HttpRequest requset,
                        HttpResponse response){

    ActionSupportService actionService =
(actionService) getWebApplicationContext
                                    .getBean("loginService");
   String fullName = actionService.createFullName(
                                firstName,lastName);
 ・・・(省略)・・・
   }
}
```

まとめ

- ビジネスロジック層が変更されても、Actionクラス内は変更せず、読み込むxmlファイルの<bean id="" class名="">を変更するだけでOK！！

- 上記を実現するためSpringが提供しているActionSupport を継承して実装することでSpring application contextへのリファランスが取得できていることが保守や拡張のしやすさが実現できると思われる。（getWebApplicationContext()メソッド）

- ActionSupportは簡単にSpringとStrutsの連携ができるが、デメリットとしてSpring AOPが使えないことがあげられる。

Bean定義ファイル

●Bean定義ファイル例

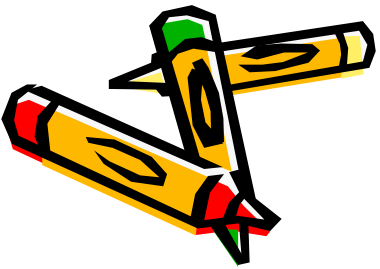
```
<beans>
<bean id =“オブジェクトA”  class=“パッケージ名.クラス名A”>
<property name=“変数名”><value>文字列</value></property>
<property name=“変数名”><value>文字列</value></property>
</bean>
<bean id=“オブジェクトB” class=“パッケージ名.クラス名B”>
</bean>
<bean id=“オブジェクトC” class=“パッケージ名.クラス名C”>
  <property name=“変数名”><ref bean=“オブジェクト名B”/>
  </property>
</bean>
<beans>
```

beanタグの属性

属性	意味
id	オブジェクト名
name	オブジェクトに別名をつける。空白、「,」、「;」で区切ることにより、複数の名前をつけることができる。StrutsプラグインなどContextLoaderPlaginを利用した場合使用する。
class	idの実装。パッケージ名+クラス名
parent	設定情報を引き継ぐオブジェトのidを指定する
singleton	**true** 属性を省略した場合のデフォルト。メソッドgetBeanで取得する。オブジェクトはシングルトン **false** メソッドgetBean取得するオブジェクトは毎回インスタンス化されたもの
lazy-init	**true** オブジェクトの生成を遅らせる **false** 属性を省略した場合のデフォルト。Beanファクトリの起動時にオブジェクトを生成する。

beanタグの属性

属性	意味
Autowire	
no	省略した場合デフォルト。<property>タグには<ref>タグで指定されたオブジェクトがプロパティに設定される。
byName	指定した名前のオブジェクトがプロパティに設定される **byName=employee** は**setEmployee()**に相当
byType	指定されたタイプのオブジェクトがプロパティに設定される
constructor	byTypeと同義。Type3 利用時に利用する。
autodetect	byTypeもしくはcustrutorのいずれかを実行する。
dependency-check	
none	属性を省略した場合デフォルト。 依存関係のチェックをしない。
simple	プロパティに基本型が設定されているかチェック
object	プロパティにオブジェクトが設定されているかチェック
all	simpleとobjectの複合



beanタグの属性

属性	意味
depend-on	依存関係の対象となるオブジェクトの存在をチェックする
init-method	メソッド名を記述することにより、プロパティの設定後に呼ばれる。 ここで指定するメソッドには引数がないこと
destroy-method	メソッド名を記述することにより、システム終了時に呼ばれる。 ここで　指定するメソッドを持つオブジェクトはシングルトンであること。