Generalitat de Catalunya
Centre de Telecomunicacions
i Tecnologies de la Informació

Framework Corporatiu J2EE

Serveis de la capa de presentació

Versió 1.2

Barcelona, 13 / febrer / 2007

Històric de modificacions

Data	Autor	Comentaris	Versió
09/12/2006	Atos Origin, sae openTrends	Versió inicial del document	1.0
01/03/2006	Atos Origin, sae openTrends	<Rev 1>. Canvis format dels gràfics	1.0
13/02/2007	Atos Origin, SAE	Versió 1.2 d'OpenFrame	1.2

Llegenda de Marcadors

Índex

1. Introducció

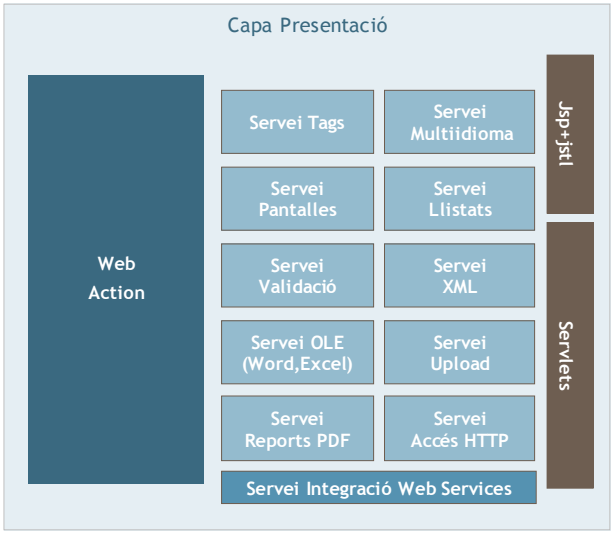
1.1. Propósit

El propósit d'aquest document és donar una visió general de quina és l'arquitectura base de la capa de presentació i els seus components, així com la configuració necessària.Es tractarà entre d'altres:

- Quins són els components principals de l'arquitectura de la capa de presentació
- Quin és el patró recomanat en el desenvolupament de les aplicacions
- Quin és el procediment que segueix l'arquitectura interna de openFrame en el tractament d'una petició
- Quina és la configuració bàsica de la capa de presentació, precondició per la utilització dels nostres serveis de presentació.

1.2. Context i Escenaris d'Ús

Els serveis de presentació es troben dins la Capa de Presentació definida a la partició en 3 capes de openFrame.



1.3. Versions i Dependències

Veure versions i dependències de cadascun dels serveis.

1.4. A qui va dirigit

Aquest document va dirigit als següents perfils:

- Programador. Per tal d'utilitzar els serveis de presentació proporcionats, l'ús de les classes bàsiques de control i els tags JSP a incorporar a les pàgines

- Arquitectes. Per conéixer quin és el procediment de tractament d'una petició, quins són els components principals i l'arquitectura genèrica del servei.

1.5. Documents i Fonts de Referència

Desavantatges de Struts	http://rollerjm.free.fr/pro/Struts11.html

1.6. Glossari

POJO (Plain Old Java Object)

S'utilitza per anomenar classes que són ordinàries, és a dir no implementa cap interfície ni extén cap classe d'un framework concret. Tot i que s'apropa molt al terme 'JavaBeans' es tracta d'un terme més atractiu pel fet que el nom "bean" dóna lloc a un conflicte entre 2 termes totalment diferents com són 'JavaBeans' i 'Enterprise Java Beans'.

DAO (Data Access Object)

Patró de disseny mitjançant un component proporciona una interfície entre l'aplicació i un o més dispositius d'emmagatzematge de dades (base de dades, fitxer, ...).

Els Data Access Objects o DAOs són un patró de disseny J2EE. L'avantatge principal del seu ús és que l'objecte de negoci (Business Objects) no necessita conéixer el destí de la informació que manipula. En Java es poden usar per aillar l'aplicació de la tecnologia de persistència utilitzada (JDBC, JDO, EJBs, Hibernate, etc.). En cas de que es canvii la tecnologia de persistència no caldrà canviar cap part de l'aplicació de negoci.

BO (Business Object)

Els Business Objects (o BO) abstrauen les entitats del domini que utilitza l'aplicació. A vegades són anomenats 'Domain Objects'. Encapsulen les dades i el comportament associat a l'entitat que representen.

2. Descripció Detallada

2.1. Arquitectura i Components

El propósit d'aquest apartat és mostrar quina és l'arquitectura del servei i els components que la composen.

En aquest apartat es detallen els següents aspectes:

- Tractament de les peticions Web. Quin és el procediment de tractament d'una petició, des de que arriba al contenidor fins que és tractat per una classe concreta.
- Components Control·lador. Components o classes de tipus control·lador dins l'arquitectura MVC (model-vista-control·lador)
- Components Vista. Components o classes de tipus vista dins l'arquitectura MVC
- Components Model. Cóm integrar els components de negoci o de model amb els components control·lador

2.1.1. Arquitectura i Patrons d'Ús

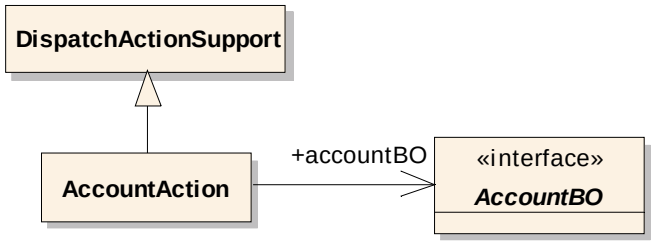
A l'hora de definir les nostres aplicacions és important que es faci un ús coherent de varis patrons de disseny.

El patró de disseny a considerar és el Model Vista Control·lador clàssic.

Així, el Servei de Presentació està enfocat a la resolució dels components Vista i Control·lador, propis d'una capa de presentació. El model no ha de fer servir cap tecnologia concreta i es poden desenvolupar mitjançant classes planes o POJOs. La interacció entre l'acció i el negoci ha de ser totalment transparent dels objectes de presentació usats. És a dir, el negoci no ha de tractar amb paràmetres de tipus 'HttpServletRequest' o 'ActionForms', pel que és tasca de la capa de presentació fer la transformació entre els objectes de presentació i els de negoci.

Qualsevol petició rebuda haurà de ser gestionada per classes de tipus 'Action'. Aquestes han d'heretar de la classe 'net.opentrends.openframe.services.web.struts.DispatchActionSupport'.

L'acció no ha de realitzar cap tractament de negoci, només aspectes de presentació. Així no es permet que l'acció pugui accedir a la base de dades de forma directa. Tota delegació l'ha de fer sobre objectes de negoci que gestionaran el tractament. Aquests objectes de negoci, poden, de forma adicional fer ús de classes DAO d'accés a la base de dades per a persistir els objectes de negoci o obtenir-los.



2.1.2. Vinculació de l'Objecte de Negoci

Una de les parts més important de openFrame és la utilització dels objectes de negoci com a models de la nostra presentació (al contrari que Struts on es fan servir 'ActionForms' per a representar el nostre model i cal fer el pas entre aquests objectes i els objectes de negoci).

Per a que les nostres accions sàpiguen quin tipus de classe és el que han de utilitzar cal configurar la propietat 'pojoClass').

```
   <bean name="/accounts"

   class="net.opentrends.openframe.samples.jactionServlet.struts.action.AccountAct
ion">
          <property name="pojoClass"

   value="net.opentrends.openframe.samples.jactionServlet.model.Account" />
```

2.1.3. Tractament de les peticions Web

En aquest apartat s'explica quin és el flux que segueix una petició des de que arriba al contenidor del servidor d'aplicacions fins que és tractat per un component control·lador específic.

El tractament d'una petició des d'un navegador inclou els següents pasos:

1) Tota petició HTTP dirigida a l'aplicatiu és capturada per un servlet principal (DispatcherServlet), tal i com s'ha configurat al fitxer de deployment 'web.xml'.

```
<servlet>
   <servlet-name>mainServlet</servlet-name>
   <servlet-class>
          org.springframework.web.servlet.DispatcherServlet❶
   </servlet-class>
   <init-param>
          <param-name>contextConfigLocation</param-name>
          <param-value>
                 /WEB-INF/classes/spring/apl-servlet.xml❷
          </param-value>
   </init-param>
   <load-on-startup>2</load-on-startup>
```

```
</servlet>
```

❶ Servlet principal que rep les peticions
❷ Fitxer que defineix els mappings de gestors per patró de url

2) El Servlet consulta la variable 'urlMapping' en el fitxer especificat al paràmetre 'contextConfigLocation'. En aquesta variable s'especifiquen patrons de tractament de peticions i quins són els seus gestors.

```
<bean id="urlMapping"
class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
   <property name="interceptors">
          <list>
                 <ref bean="openSessionInViewInterceptor" />
                 <ref bean="exportInterceptor" />
          </list>
   </property>
   <property name="mappings">
          <props>
                 <prop key="*.do">strutsWrappingController❶</prop>
                 <prop key="pdfView.pdf">pdfController</prop>
                 <prop key="excelView.xls">excelController</prop>
          </props>
   </property>
</bean>
```

❶ En rebre una petició amb el patró final '*.do' (patró utilitzat per Struts) detecta que el gestor associat és 'strutsWrappingController'

3) Una vegada detectat el gestor associat (en el cas més comú la classe 'ActionServlet', de Struts) li cedeix la petició

```
<bean id="strutsWrappingController"
class="org.springframework.web.servlet.mvc.ServletWrappingController">
   <property name="servletClass">
          <value>org.apache.struts.action.ActionServlet❶</value>
   </property>
   <property name="servletName">
          <value>action</value>
   </property>
   <property name="initParameters">
          <props>
                 <prop key="config">
                        /WEB-INF/classes/struts/struts-config.xml❷
                 </prop>
          </props>
   </property>
</bean>
```

❶ Servlet principal de Struts

❷ Fitxer de mappings de Struts. Aquest fitxer serà utilitzat per determinar on anirà cada petició (una classe 'Action'), quins beans de formulari usar i on anar una vegada la petició s'hagi resolt.

NOTA: Es recomana una lectura prèvia de 'http://struts.apache.org/struts-doc-1.2.8/userGuide/index.html' per conéixer en detall cóm funciona Struts.

4) El 'ActionServlet' de Struts, segons el fitxer de mappings definit (veure 'config' a la propietat 'initParameters') cedeix la petició al 'processorClass' configurat.

```
<controller>
   <set-property property="processorClass"
value="net.opentrends.openframe.services.web.struts.ExtendedDelegatingTilesRequest
Processor" ❶/>
</controller>
```

❶ RequestProcessor de Struts al que es delega la petició. Cal recordar que Struts permet l'extensió del seu comportament mitjançant l'extensió del seu bàsic RequestProcessor.

5) La classe 'RequestProcessor' consulta al fitxer 'struts.config.xml' quina és la classe 'Action' que tractarà la petició

```
<action path="/accounts" name="actionForm" scope="request"
          parameter="reqCode">
          <forward name="error" path="pages.accounts" />
          <forward name="success" path="pages.accounts"/>
</action>
```

6) La classe 'ExtendedDelegatingTilesRequestProcessor', en heretar de la classe de Spring 'org.springframework.web.struts.DelegatingTilesRequestProcessor' redefineix la creació de la classe 'Action' per tal d'injectar-li les propietats que haguem configurat (aquest és un comportament diferenciat del clàssic Struts).

Aquesta injecció es realitza amb el mecanisme genèric de Spring com si d'una classe normal es tractés.

```
<bean name="/accounts" parent="accountBaseDefinition">
   <property name="valueListActionHelper">
          <bean parent="valueListActionHelper">
                 <property name="listName">
                 <value>accountList</value>
                 </property>
                 <property name="tableId" value="ACCOUNT"/>
          </bean>
   </property>
```

```
    <property name="servletRequestDataBinderFactory"
           ref="servletRequestDataBinderFactory"/>
    <property name="logService" ref="loggingService"/>
    <property name="pojoClass"
    value="net.opentrends.openframe.samples.jactionServlet.model.Account" />
    <property name="accountBO" ref="accountBO"/>
</bean>
```

NOTA: Per a injectar-hi les propietats, el processador ha de conéixer en quin fitxer es realitzen les injeccions, per això es defineix en el fitxer 'web.xml' el paràmetres de context 'contextConfigLocation'

```
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>
           /WEB-INF/classes/spring/applicationContext.xml
           /WEB-INF/classes/spring/action-servlet.xml
    </param-value>
</context-param>
```

Consultar 'http://static.springframework.org/spring/docs/1.2.x/reference/beans.html' per més referència del mecanisme d'injecció.

Mitjançant aquest mecanisme hem eliminat la necessitat de que les nostres Action siguin 'thread-safe' (una de les restriccions de Struts).

Una vegada obtinguda la instància 'Action' que tracta la petició, el processador consulta si s'ha definit la propietat 'pojoClass' de la Action (en la injecció l'haurem incorporat). Si és així, i aquí radica una de les diferències més importants amb el tractament tradicional de Struts, el processador openFrame realitza la traducció dels paràmetres rebuts a la petició en atributs de l'objecte definit al 'pojoClass'. Per a fer aquesta traducció s'ajuda del 'servletRequestDataBinderFactory' que s'ha definit al Action.

```
<bean name="/accounts" parent="accountBaseDefinition">
    ...

    <property name="servletRequestDataBinderFactory"
           ref="servletRequestDataBinderFactory"/>

</bean>
```

Mitjançant aquesta característica, eliminem un dels desavantatges de Struts (comentat prèviament) pel qual que no s'establia la política de translació des dels paràmetres de la petició (HttpServletRequest) a objectes Java comuns.

Per a que des dels nostres Action poguem accedir a l'objecte populat, es fa ús de formularis de tipus 'SpringBindingActionForm'. L'ús d'aquests tipus, enlloc del tradicional 'ActionForm' de Struts, permet l'ús de objectes des dels tags clàssics de Struts (i per tant de l'extensió realitzada a openFrame) sense necessitat de definir formularis cópia dels objectes de negoci.

```
<form-beans>
   <form-bean name="actionForm"
type="net.opentrends.openframe.services.web.struts.SpringBindingActionForm"/>
</form-beans>
```

La nostra acció, per tant farà ús d'aquest formulari:

```
<action path="/accounts" name="actionForm" ...
```

En aquest moment ja hem eliminat un altre dels desavantatges de Struts comentat prèviament en el que es feia palés que els ActionForms són estructures innecessàries per expressar el model.

7) Finalment, es delega al Action el tractament de la petició. Aquest Action té injectats tots els objectes necessaris.

   El Action pot accedir a l'objecte populat mitjançant el mètode 'getTarget' del form:

```
SpringBindingActionForm actionForm = (SpringBindingActionForm) form;
Account vo = (Account) actionForm.getTarget();
```

8) A partir d'aquest moment, és important que es segueixi el patró de MVC pel que les nostres Action no haurien de realitzar cap funcionalitat de negoci i delegar tot el que no sigui própiament presentació a una classe manager o BO (Business Object). A aquesta classe li passarà l'objecte de negoci populat internament per l'arquitectura i obtingut en el pas anterior.

```
vo = accountBO❶.load(vo);
```

- Aquest objecte ha estat declarat mitjançant injecció de Spring a la nostra Action

9) Una vegada finalitzat el tractament, la classe Action només ha de retornar la pantalla de finalització de la petició.

```
        return mapping.findForward("success");
```

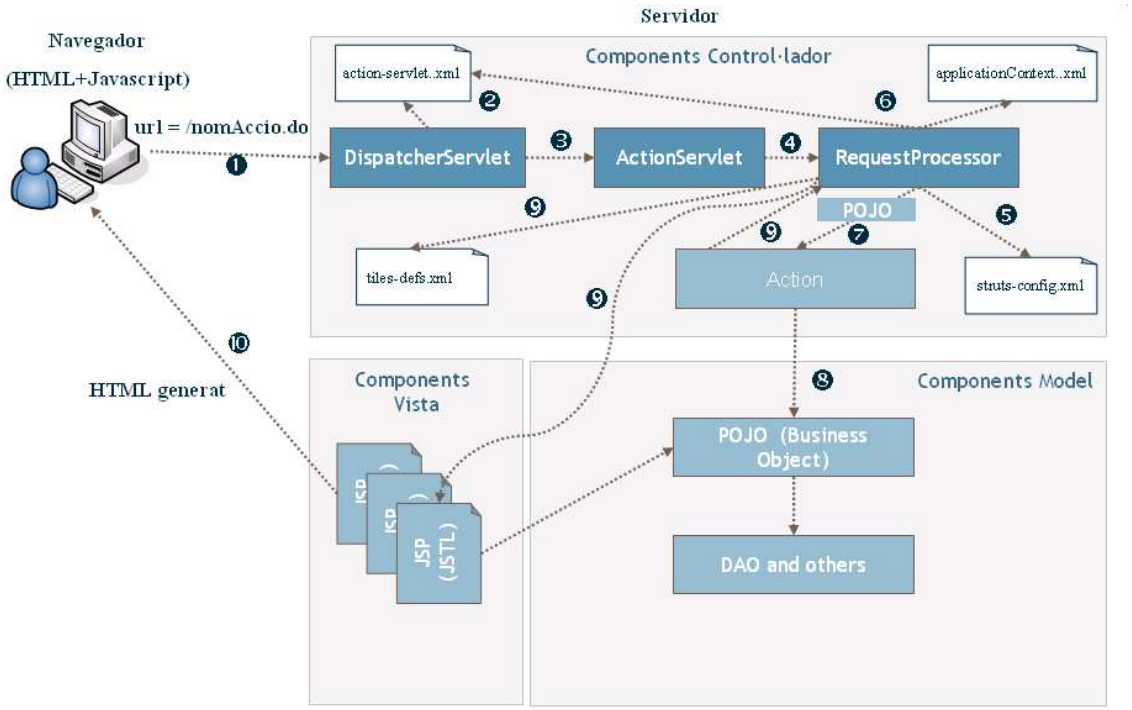
El nom de la pantalla és un nom lógic que correspon a un dels forwards definits al fitxer de configuració de l'acció (struts-config.xml).

```
<forward name="success" path="pages.accounts"/>
```

En cas que el path de retorn no correspongui a una pàgina JSP, entra el mecanisme de pantalles definit pel Servei de Pantalles. En aquest cas, la pantalla és construída a partir de la definició realitzada al fitxer 'tiles-defs.xml'.

10) Finalment, la pàgina construída és retornada a l'usuari

En el següent gràfic podem veure de forma resumida el procés descrit:



Dins aquest gràfic és tasca del programador la de definir:

- Fitxer de configuració 'action-servlet.xml'. Per definir les accions per tal que puguin injectar-se atributs
- Fitxer de configuració 'applicationContext.xml'. Per definir els atributs i objectes injectables
- Fitxer de configuració 'struts-config.xml'. Per definir el fitxer de configuració de accions de Struts (formulari, path de l'acció, forwards, ...)
- Fitxer de configuració 'tiles-defs.xml'. Per definir les pantalles de l'aplicació
- Classe Action. Per definir cóm gestionar una petició web concreta i la integració amb la capa de negoci.
- Altres classes. Una vegada definida la classe Action caldrà definir classes (si no es troben ja definides) per realitzar la funcionalitat del negoci.

2.1.4. Components Control·lador

Dins els components Control·lador trobem tots aquells elements que s'encarreguen de gestionar una petició i interaccionar amb les vistes que presentaran les dades del model a l'usuari.

Component	Package	Descripció
DispatcherServlet	org.springframework.web.servlet	Només cal conéixer el seu funcionament en cas de voler modificar l'arquitectura proporcionada

Component	Package	Descripció
ServletWrappingController	org.springframework.web.servlet.mvc	Controlador de Spring que redirigeixla petició a d'altres Controlador Spring que encapsula una instància de Servlet. Permet encapsular per exemple diferents tipus de controladors segons el tipus de petició. En el cas de '.do' podem per exemple encapsular el 'ActionServlet' de struts. Veure secció 'Instal·lació i Configuració'
ActionServlet	org.apache.struts.action	Servlet principal Struts de recepció de les peticions. Delega tot el tractament en el 'RequestProcessor' definit (en el cas de openFrame sobre 'ExtendedDelegatingTilesRequestProcessor')
ExtendedDelegatingTilesRequestProcessor	net.opentrends.openframe.services.web.struts	Processador de les peticions de Struts de openFrame.
DispatchActionSupport	net.opentrends.openframe.services.web.struts	Classe base Struts de les accions de l'aplicació

- **ExtendedDelegatingTilesRequestProcessor**

openFrame ha definit aquesta classe per gestionar les peticions rebudes. Hereta de la classe 'DelegatingTilesRequestProcessor' de Spring per a incorporar els següents tractaments:

1) Tractament de les excepcions rebudes des de les capes inferiors (negoci, presentació, etc.). Encapsula en objectes de tipus 'ActionMessage' de Struts els missatges a mostrar a l'usuari
2) Preparació del context dels tags. Per a que els tags puguin obtenir la seva configuració de forma externa des de Spring deixa a l'atribut del request 'CONFIGURATION_TAG' la informació dels tags per l'acció actual
3) Realitzar el binding entre els paràmetres rebuts i el POJO configurat a l'acció

En el següent gràfic es mostra la jerarquia d'herència de la classe.

2.1.5. Components Vista

Els components vista de openFrame están formats per la conjunció entre diferents elements:

- JSP (Java Server Pages) per representar les pàgines
- JSTL i Tags com a helpers des de les pàgines
- SpringBindingActionForm per fer ús dels objectes de negoci com a formularis de Struts

La definició dels tags i el seu ús es troba en el document 'Servei de Presentació: Tags'.
l'usuari.

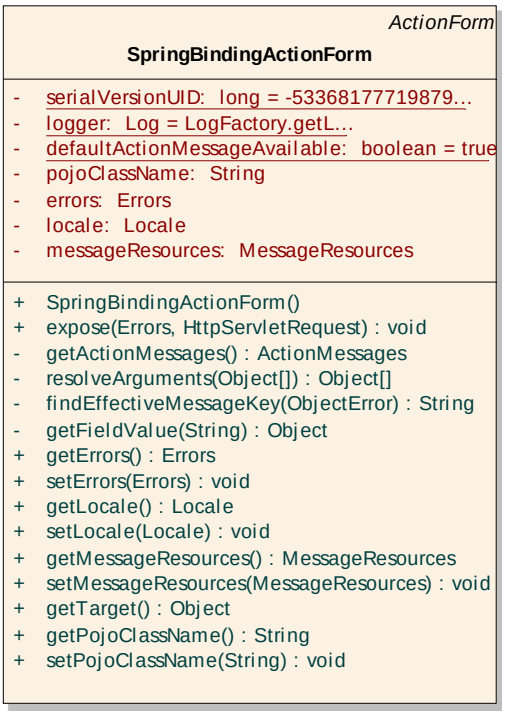
Component	Package	Descripció
SpringBindingActionForm	net.opentrends.openframe.services.web.struts	Formulari Struts que permet eliminar la necessitat de definir formularis Struts

- **SpringBindingActionForm**

Classe adaptador dels ActionForm de Struts que permet l'ús de objectes de forma directa des de les pàgines.

Struts requereix de l'ús de FormBeans per cada formulari amb el conseqüent ús de classes extra que en realitat no són necessàries. A més, les propietats d'aquests formularis són normalment Strings, pel que cal sempre la utilització de conversions de tipus (ho solventa mitjançant l'ús de Commons BeanUtils).

Mitjançant aquesta classe, les nostres accions de Struts no definiran cap formulari, sinó que es farà referència a un formulari d'aquest tipus de classe.



2.2. Instal·lació i Configuració

2.2.1. Configuració Bàsica

Paràmetres de context

Fitxer de configuració: web.xml

Ubicació proposada: <PROJECT_ROOT>/src/main/webapp/WEB-INF/web.xml
NOTA: Aquesta ubicació és la proposada en cas d'utilització de Maven

En aquest apartat definirem els següents paràmetres:

```
<web-app>
       <context-param>
```

- contextConfigLocation

Indicar les referències als fitxers de configuració 'action-servlet.xml', on definirem les configuracions de les accions de Struts, i el fitxer 'applicationContext.xml' on definirem la injecció dels altres objectes (negoci, daos, ...)

```
<context-param>
   <param-name>contextConfigLocation</param-name>
   <param-value>
          /WEB-INF/classes/spring/applicationContext.xml
          /WEB-INF/classes/spring/action-servlet.xml
   </param-value>
</context-param>
```

Filtres

Fitxer de configuració: web.xml

Ubicació proposada: <PROJECT_ROOT>/src/main/webapp/WEB-INF/web.xml
NOTA: Aquesta ubicació és la proposada en cas d'utilització de Maven

- Struts Locale Filter

```
<web-app>
       <filter>
       ...
```

Aquest filtre s'encarrega de integrar el Servei d'Internacionalització per tal que des de Struts es pugui fer ús de fitxers d'internacionalització.

Atributs:

Atribut	Requerit	Valor

Atribut	Requerit	Valor
filter-name	Sí	Nom del filtre Exemple: Struts Locale Filter
filter-class	Sí	net.opentrends.openframe.services.web.i18n.StrutsLocaleFilter

```
<filter>
   <filter-name>Struts Locale Filter</filter-name>
   <filter-class>
   net.opentrends.openframe.services.web.i18n.StrutsLocaleFilter
   </filter-class>
</filter>
```

- JSTL Locale Filter

  Aquest filtre s'encarrega de integrar el Servei d'Internacionalització per tal que es pugui fer servir des de tags JSTL de tipus fmt

  Atributs:

Atribut	Requerit	Descripció
filter-name	Sí	Nom del filtre Exemple: Struts Locale Filter
filter-class	Sí	Usar 'net.opentrends.openframe.services.web.i18n.JSTLLocaleFilter'

```
<filter>
   <filter-name>JSTL Locale Filter</filter-name>
   <filter-class>
          net.opentrends.openframe.services.web.i18n.JSTLLocaleFilter
   </filter-class>
</filter>
```

- Acegi Filter Chain Proxy

  Aquest filtre s'ha definir per integrar el **Servei de Seguretat** de forma que qualsevol petició sigui filtrada prèviament abans d'arribar a l'Action.

  Atributs:

Atribut	Requerit	Descripció
filter-name	Sí	Nom del filtre Exemple: Acegi Filter Chain Proxy
filter-class	Sí	Usar

Atribut	Requerit	Descripció
		'net.opentrends.openframe.services.web.i18n.JSTLLocaleFilter'

Paràmetres d'inicialització:

```
<web-app>
      <filter>
            <init-param>
```

Atribut	Requerit	Valor
param-name	Sí	targetClass
param-value	Sí	'net.sf.acegisecurity.util.FilterChainProxy"

```
<filter>
   <filter-name>Acegi Filter Chain Proxy</filter-name>
   <filter-class>
          net.sf.acegisecurity.util.FilterToBeanProxy
   </filter-class>
   <init-param>
          <param-name>targetClass</param-name>
          <param-value>
                net.sf.acegisecurity.util.FilterChainProxy
          </param-value>
   </init-param>
</filter>
```

Mapeig dels Filtres

Fitxer de configuració: web.xml

Ubicació proposada: <PROJECT_ROOT>/src/main/webapp/WEB-INF/web.xml
NOTA: Aquesta ubicació és la proposada en cas d'utilització de Maven

En aquest apartat configurarem en quins casos s'apliquen els filtres

```
<web-app>
      <filter-mapping>
```

- Struts Locale Filter

  Definirem que s'activi el filtre 'Struts Locale Filter' en els següents casos:

  - *.do. Per activar el filtre quan es facin peticions als Action de Struts
  - *.pdf. Per activar el filtre per generar els fitxers pdf de sortida dels llistats
  - *.xls. Per activar el filtre per generar els fitxers xls de sortida dels llistats
  - /dwr/*. Per activar el filtre des de Ajax

```
<filter-mapping>
   <filter-name>Struts Locale Filter</filter-name>
```

```
    <url-pattern>*.do</url-pattern>
</filter-mapping>
<filter-mapping>
    <filter-name>Struts Locale Filter</filter-name>
    <url-pattern>*.pdf</url-pattern>
</filter-mapping>
<filter-mapping>
    <filter-name>Struts Locale Filter</filter-name>
    <url-pattern>/dwr/*</url-pattern>
</filter-mapping>

<filter-mapping>
    <filter-name>Struts Locale Filter</filter-name>
    <url-pattern>*.xls</url-pattern>
</filter-mapping>
```

- JSTL Locale Filter

  Definirem que s'activi el filtre 'JSTL Locale Filter' en els mateixos casos anteriors:

```
<filter-mapping>
    <filter-name>JSTL Locale Filter</filter-name>
    <url-pattern>*.do</url-pattern>
</filter-mapping>
<filter-mapping>
    <filter-name>JSTL Locale Filter</filter-name>
    <url-pattern>/dwr/*</url-pattern>
</filter-mapping>
<filter-mapping>
    <filter-name>JSTL Locale Filter</filter-name>
    <url-pattern>*.pdf</url-pattern>
</filter-mapping>
<filter-mapping>
    <filter-name>JSTL Locale Filter</filter-name>
    <url-pattern>*.xls</url-pattern>
</filter-mapping>
```

- Acegi Filter Chain Proxy

  Definirem que s'activi el filtre per qualsevol url

```
<filter-mapping>
          <filter-name>Acegi Filter Chain Proxy</filter-name>
          <url-pattern>/*</url-pattern>
</filter-mapping>
```

Listeners

```
<web-app>
      <listener>
```

Fitxer de configuració: web.xml

Ubicació proposada: <PROJECT_ROOT>/src/main/webapp/WEB-INF/web.xml
NOTA: Aquesta ubicació és la proposada en cas d'utilització de Maven
Es definiran els següents listeners:

```
<listener>
   <listener-class>
          org.springframework.web.context.ContextLoaderListener
   </listener-class>
</listener>
<listener>
   <listener-class>
net.opentrends.openframe.services.web.context.AspectWerkzContextListener
   </listener-class>
</listener>
```

Per a més referència consultar
'http://www.springframework.org/docs/api/org/springframework/web/context/ContextLoaderListener.html'

Servlets

Fitxer de configuració: web.xml

Ubicació proposada: <PROJECT_ROOT>/src/main/webapp/WEB-INF/web.xml
NOTA: Aquesta ubicació és la proposada en cas d'utilització de Maven

```
<web-app>
      <servlet>
```

En aquest apartat configurarem en quins casos s'apliquen els filtres definits prèviament

- Servlet Central de les Peticions Struts

  Atributs:

Atribut	Requerit	Descripció
servlet-name	Sí	Nom del servlet Exemple: actionServlet
servlet-class	Sí	Usar 'org.springframework.web.servlet.DispatcherServlet'
init-param	Sí	Inicialització dels paràmetres: • contextConfigLocation Ubicació del fitxer que lliga els Action de Struts definits a 'struts-config.xml' amb la injecció dels seus atributs Valor: /WEB-INF/classes/spring/action-servlet.xml

```
<servlet>
   <servlet-name>actionServlet</servlet-name>
   <servlet-class>
          org.springframework.web.servlet.DispatcherServlet
   </servlet-class>
   <init-param>
          <param-name>contextConfigLocation</param-name>
          <param-value>
                 /WEB-INF/classes/spring/action-servlet.xml
          </param-value>
   </init-param>
   <load-on-startup>2</load-on-startup>
</servlet>
```

- Servlet de comunicació Ajax amb el Servidor

  Atributs:

Atribut	Requerit	Descripció
servlet-name	Sí	Nom del servlet Exemple: dwr-invoker
servlet-class	Sí	Usar 'uk.ltd.getahead.dwr.DWRServlet'
init-param	Sí	Inicialització dels paràmetres: • config Ubicació del fitxer de definició dels objectes Spring exposats. Valor: /WEB-INF/classes/dwr/dwr.xml

```
<servlet>
   <servlet-name>dwr-invoker</servlet-name>
   <servlet-class>uk.ltd.getahead.dwr.DWRServlet</servlet-class>
   <init-param>
          <param-name>config</param-name>
          <param-value>/WEB-INF/classes/dwr/dwr.xml</param-value>
   </init-param>
</servlet>
```

Mappings de urls i Servlets

Fitxer de configuració: web.xml

Ubicació proposada: <PROJECT_ROOT>/src/main/webapp/WEB-INF/web.xml
NOTA: Aquesta ubicació és la proposada en cas d'utilització de Maven

```
<web-app>
       <servlet-mapping>
       ...
```

En aquest apartat configurarem quins servlets s'usaran per rebre les peticions segons el patró de url rebut.

Definirem els mapejos dels patrons següents al servlet 'actionServlet':

- *.do, *.doc, *.report, *:xml, *.xls i *.pdf

Tal i com es mostra a continuació:

```
<servlet-mapping>
   <servlet-name>actionServlet</servlet-name>
   <url-pattern>*.do</url-pattern>
</servlet-mapping>
<servlet-mapping>
   <servlet-name>actionServlet</servlet-name>
   <url-pattern>*.doc</url-pattern>
</servlet-mapping>
<servlet-mapping>
   <servlet-name>actionServlet</servlet-name>
   <url-pattern>*.report</url-pattern>
</servlet-mapping>
<servlet-mapping>
   <servlet-name>actionServlet</servlet-name>
   <url-pattern>*.xml</url-pattern>
</servlet-mapping>
<servlet-mapping>
   <servlet-name>actionServlet</servlet-name>
   <url-pattern>*.xls</url-pattern>
</servlet-mapping>
<servlet-mapping>
<servlet-name>actionServlet</servlet-name>
   <url-pattern>*.pdf</url-pattern>
</servlet-mapping>
```

Configuració de Struts

Fitxer de configuració: struts-config.xml

Ubicació proposada: <PROJECT_ROOT>/src/main/resources/struts/struts-config.xml

Per a la integració amb Struts definirem els següents elements:

- Control·lador

```
<controller>
    <set-property property="processorClass"
value="net.opentrends.openframe.services.web.struts.ExtendedDelegatingTilesRequest
Processor"/>
</controller>
```

Com a processorClass cal indicar la classe 'ExtendedDelegatingTilesRequestProcessor' del package 'net.opentrends.openframe.services.web.struts'

- Recursos de missatges

  Referència al fitxer base de internacionalització.

```
<message-resources parameter="i18n.applicationResources" null="false"/>
```

- Plugins

En aquest apartat es defineixen els plugins d'integració amb Tiles (Servei de Pantalles) i la integració amb Spring.

```
<plug-in className="org.apache.struts.tiles.TilesPlugin">
    <set-property property="definitions-config" value="/WEB-
INF/classes/struts/tiles-definitions.xml"/>
</plug-in>

<plug-in className="org.springframework.web.struts.ContextLoaderPlugIn">
    <set-property property="contextConfigLocation" value="/WEB-
INF/classes/spring/action-servlet.xml"/>
</plug-in>
```

Configuració del mapeig entre paràmetres Web i objectes negoci

Dins la classe 'ExtendedDelegatingTilesRequestProcessor' es realitza un mapeig entre els paràmetres rebuts i l'objecte configurat a la propietat 'pojoClass' de l'acció. Els valors dels paràmetres rebuts són de tipus 'String' (dins el protocol HTTP).

Fitxer de configuració: property-editors.xml

Ubicació proposada: <PROJECT_ROOT>/src/main/resources/spring/property-editors.xml

- Configuració dels editors

```
<beans>
   <bean id=”customEditors”...>
   </bean>
```

En aquest apartat es poden definir editors específics de transformació entre tipus. Per defecte no cal definir els editors de transformació entre tipus bàsics (String a boolean, String a Integer, etc.)

S'ofereixen els següents editors:

1) CustomDateEditor

Package:net.opentrends.openframe.services.i18n.spring.beans.propertyeditors

Cal definir les següents propietats:

Propietat	Requerit	Descripció
i18nService	Sí	Referència al Servei d'Internacionalització El fa servir per obtenir el llenguatge de l'usuari
localeDatePatternsMap	Sí	Mapa amb la següents característiques: • La clau correspon a un idioma • El valor correspon al patró de transformació (basat en SimpleDateFormat)

Segons el llenguatge de l'usuari, cerca si es troba com aaclau en el mapa. Si el troba, realitza el pas de tipus segons el patró definit al valor de la clau del mapa.

Exemple:

```
<bean id="customEditors" class="java.util.HashMap">
  <constructor-arg>
       <map>
           <entry key="java.util.Date">
         <bean
class="net.opentrends.openframe.services.i18n.spring.beans.propertyeditors.CustomD
ateEditor">
          <property name="i18nService" ref="i18nService"/>
<property name="localeDatePatternsMap">
             <map>
               <entry>
                 <key><value>es</value></key>
                 <value>dd/MM/yyyy</value>
               </entry>
               <entry>
                 <key><value>es_ES</value></key>
                 <value>dd/MM/yyyy</value>
               </entry>
```

```
                <entry>
                  <key><value>ca_ES</value></key>
                   <value>dd/MM/yyyy</value>
                </entry>
                <entry>
                  <key><value>ca</value></key>
                  <value>dd/MM/yyyy</value>
                </entry>
                <entry>
                  <key><value>en</value></key>
                  <value>MM/dd/yyyy</value>
                </entry>
                <entry>
                  <key><value>en_GB</value></key>
                  <value>MM/dd/yyyy</value>
                </entry>
              </map>
            </property>
   </bean>
      </entry>
      </map>
       </constructor-arg>
</bean>
```

- Configuració del configurador d'editors

```
<beans>
    <bean id=”customEditorConfigurer”...>
    </bean>
```

Per tal que funcioni correctament la transformació definirem el configurador d'editors.

Classe: CustomEditorConfigurer
Package: org.springframework.beans.factory.config.CustomEditorConfigurer

I definirem les següents propietats:

Propietat	Requerit	Descripció
customEditors	Sí	Referència als editors definits en el pas previ

```
   <bean id="customEditorConfigurer"
class="org.springframework.beans.factory.config.CustomEditorConfigurer">
      <property name="customEditors"><ref local="customEditors"></ref></property>
   </bean>
</beans>
```

2.2.2. Configuració de les Accions

La configuració de cadascuna de les nostres accions inclou 2 pasos:

- Configuració Struts. Definició de les accions segons Struts amb algunes particularitats específiques
- Configuració d'injecció de dependències. Definició de la injecció dels atributs de l'acció, com per exemple la implementació de negoci amb la que interactuarà

Configuració Struts

Fitxer de configuració: struts-config.xml

Ubicació proposada: <PROJECT_ROOT>/src/main/resources/struts/struts-config.xml

- Configuració dels formularis

  A diferència de Struts, en la que hem de definir un formulari per cada pantalla, en openFrame s'ofereix una classe 'SpringBindingActionForm' a utilitzar per les accions.

```
<form-beans>
   <form-bean name="actionForm"
type="net.opentrends.openframe.services.web.struts.SpringBindingActionForm"/>
</form-beans>
```

- Configuració de l'acció

  La configuració de l'acció es realitza de la mateixa forma que Struts, pel que es recomana una lectura prèvia de Struts per a més detalls.

```
<action path="/categories" name="actionForm" scope="request" parameter="reqCode">
      <forward name="error" path="pages.categoryEdit"/>
      <forward name="edit" path="pages.categoryEdit"/>
      <forward name="create" path="pages.categoryEdit"/>
      <forward name="list" path="pages.categories"/>
      <forward name="errorList" path="pages.categories"/>
      <forward name="success" redirect="true"/>
</action>
```

Fitxer de configuració: action-servlet.xml

Ubicació proposada: <PROJECT_ROOT>/src/main/resources/spring/action-servlet.xml

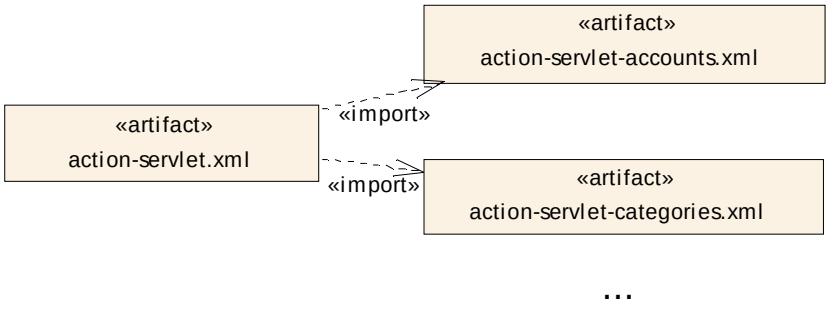
Per cadascuna de les accions de Struts definides al fitxer 'struts-config.xml' injectarem propietats adicionals que podran ser usades des de l'acció (tal i com es realitza entre qualsevol objecte segons defineix el Servei de Configuració).

Per tal que es pugui fer la injecció el id del bean ha de coincidir amb la propietat 'path' definida a l'acció dins el fitxer 'struts-config.xml'.

```
<bean id="/editAccount"
class="net.opentrends.openframe.samples.jpetstore.struts.action.AccountAction">
          <property name="logService" ref="loggingService"/>
          <property name="serializationService" ref="xmlSerializationService"/>
          <property name="pojoClass"
                 value="net.opentrends.openframe.samples.jpetstore.model.Account"
/>
          <property name="accountBO" ref="accountBO"/>


          <property name="tagsConfiguration">
</bean>
```

Es recomana crear petits fitxers que incloguin un determinat módul d'accions relacionats (per exemple tots els que tractin amb una entitat, ...) i referenciar-los des del fitxer 'action-servlet.xml' amb la sentència import.



Exemple:

```
    <import resource="action-servlet-accounts.xml"/>
    <import resource="action-servlet-categories.xml"/>
    <import resource="action-servlet-products.xml"/>
    <import resource="action-servlet-items.xml"/>
    <import resource="action-servlet-mails.xml"/>
    <import resource="action-servlet-files.xml"/>
...
```

2.3. Utilització

2.4. Eines de Suport

2.4.1. Struts Easy Plugin

Amb la instal·lació d'aquest plugin a Eclipse podem visualitzar de forma gràfica els nostres fitxers de configuració de Struts.

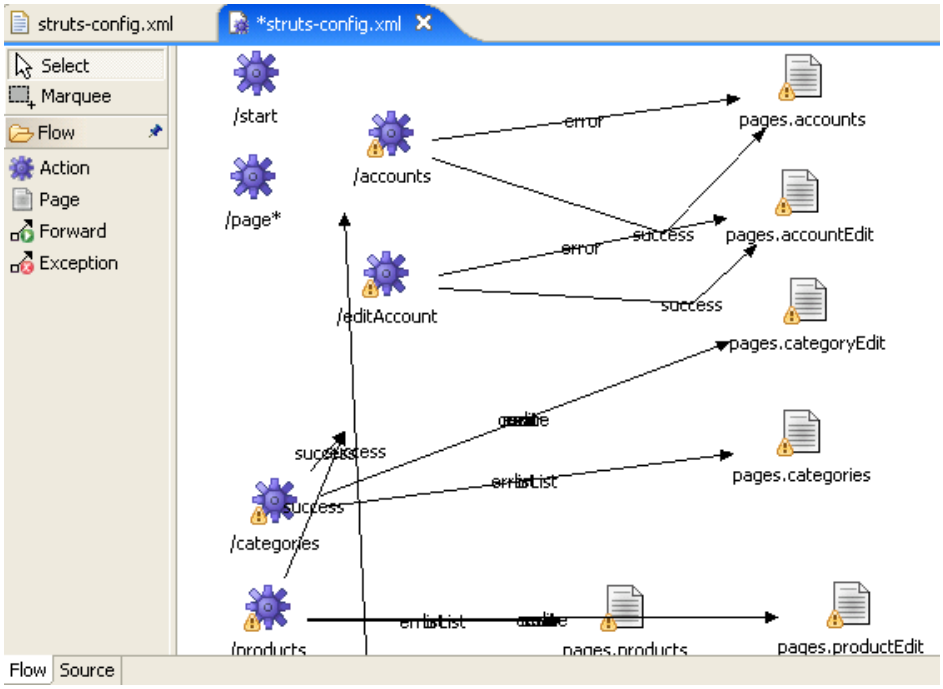
Pàgina de descàrrega:
https://sourceforge.jp/projects/amateras/files/?release_id=16539#16539

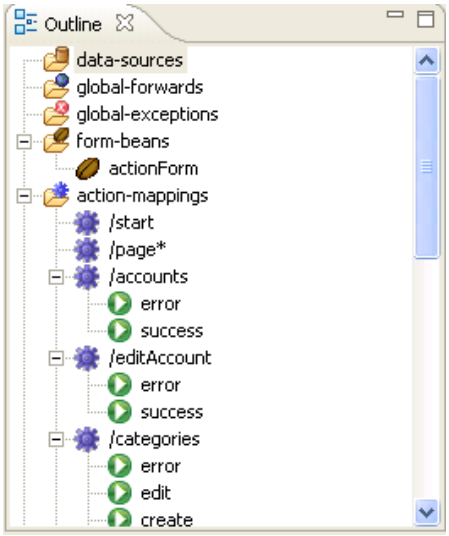
Versió actual: 1.1.7
(http://prdownloads.sourceforge.jp/amateras/16539/tk.eclipse.plugin.struts_1.1.7.zip)

Tancar Eclipse i descomprimir el fitxer al directori d'instal·lació de eclipse.

- Comprovació instal·lació

  1) Comprovar que dins el directori 'plugins' es troba un subdirectori anomenat 'tk.eclipse.plugin.struts_1.1.7'
  2) Obrir Eclipse
  3) Crear un fitxer 'struts-config.xml' si no existia prèviament
  4) Seleccionar el fitxer al navegador i amb click dret seleccionar 'Open With>struts.config.xml Editor'
  5) Veure que es representa gràficament la configuració i que a la vista 'Outline' ens apareix l'estructura jeràrquica dels elements.


struts-config.xml
*struts-config.xml
Select
Marquee
Flow
Action
Page
Forward
Exception
/start
/page*
/accounts
/editAccount
/categories
/products
error
success
pages.accounts
pages.accountEdit
pages.categoryEdit
pages.categories
pages.products
pages.productEdit
Flow
Source


Outline
data-sources
global-forwards
global-exceptions
form-beans
actionForm
action-mappings
/start
/page*
/accounts
error
success
/editAccount
error
success
/categories
error
edit
create

3. Exemples

4. Annexos