FormFaces

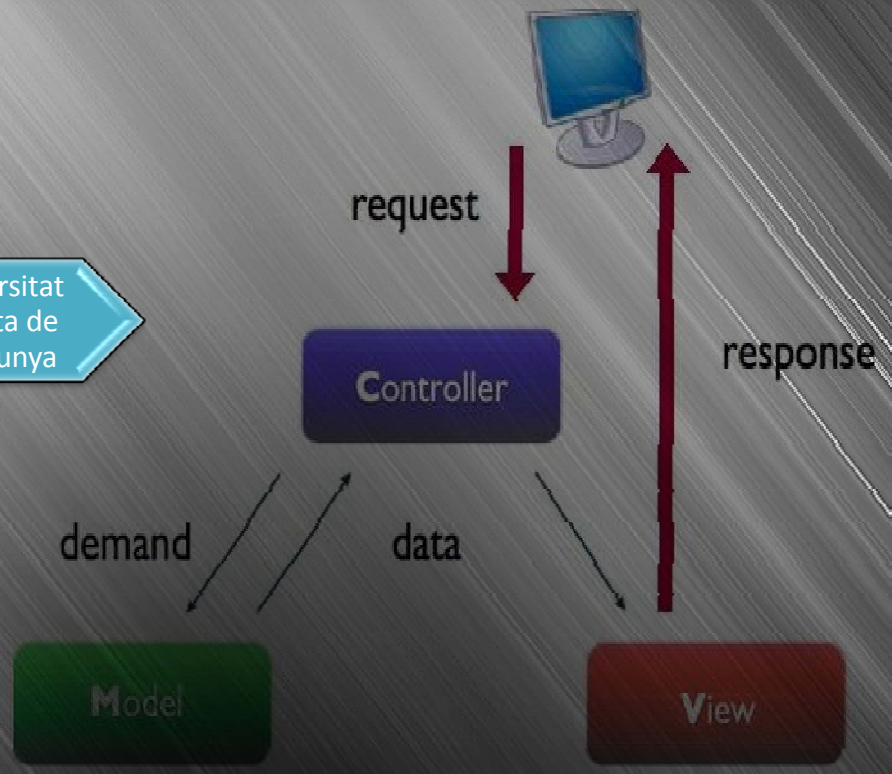
Disseny d'un marc de treball de presentació per a aplicacions J2EE

Autor: Jaume Martí Muixí

Consultor: Josep Maria Camps Riba

PFC (Projecte Fi de Carrera)

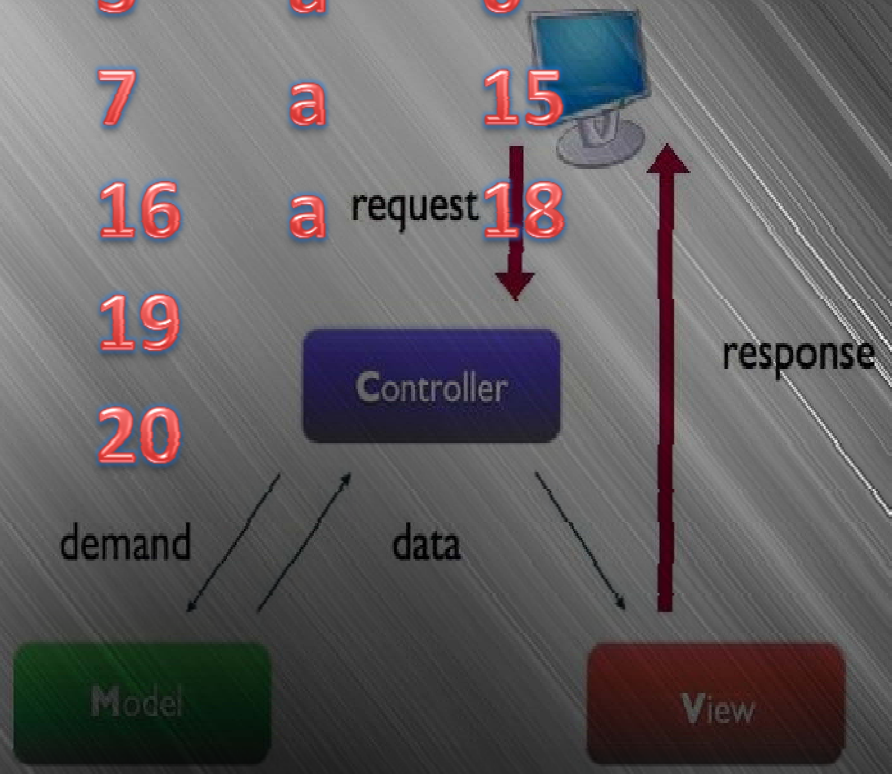
Universitat Oberta de Catalunya



FormFaces

Contingut

	Diapositives
• Introducció al tema	3 a 6
• Construcció	7 a 15
• Aplicatiu de prova	16 a 18
• Millores a fer	19
• Conclussions	20

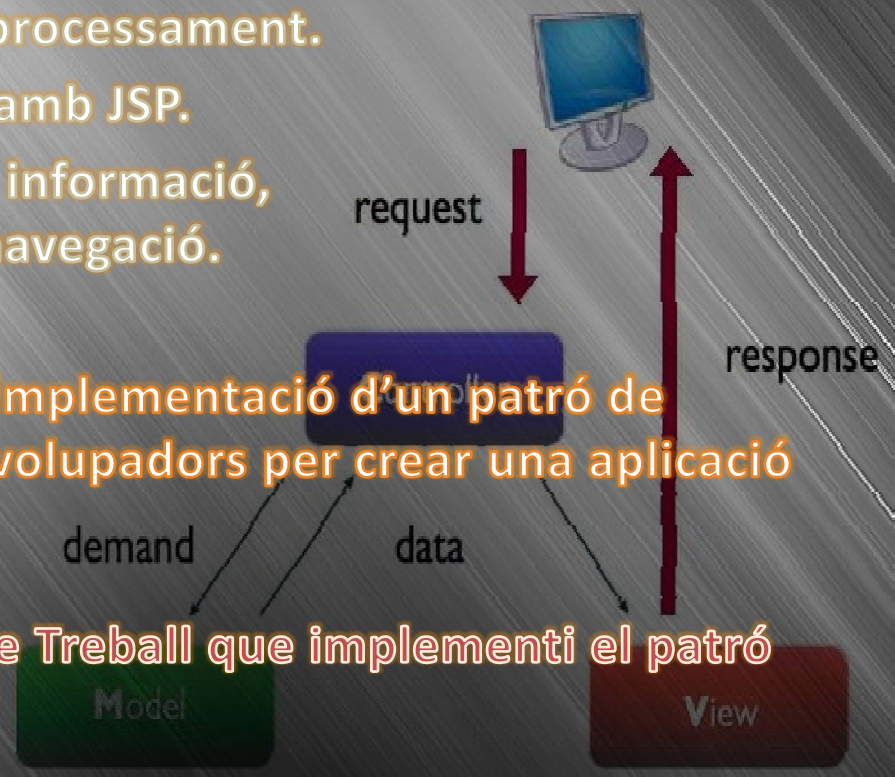


FormFaces

introducció al tema

- El *patró Model-Vista-Controlador (MVC)*

  encapsula les dades amb el seu processament (model) i les aïlla de la manipulació (controlador) i de la presentació (vista):
- Model . Encapsula les dades i el seu processament.
- Vista . Visualització de la informació amb JSP.
- Controlador. Servlet que manipula la informació, canvia d'estat el model i controla la navegació.



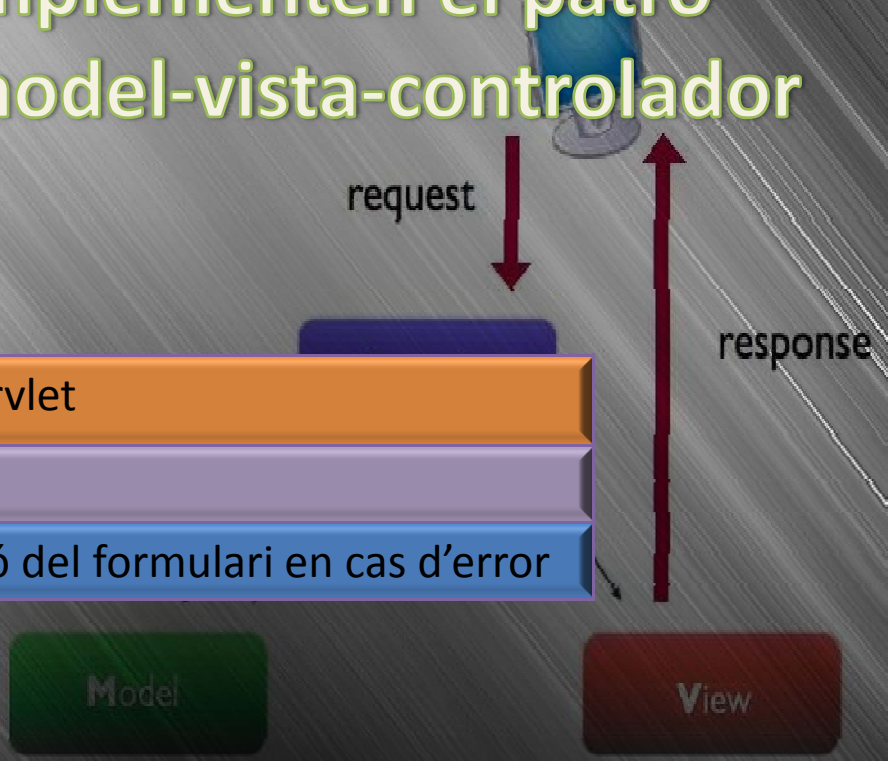
Un Framework o Marc de Treball es una implementació d'un patró de disseny que serveix d'esquelet als desenvolupadors per crear una aplicació o un aspecte d'ella.

El projecte consisteix en crear un Marc de Treball que implementi el patró MVC.

FormFaces

introducció al tema

Hem analitzat els millors Marcs de Treball que implementen el patró model-vista-controlador



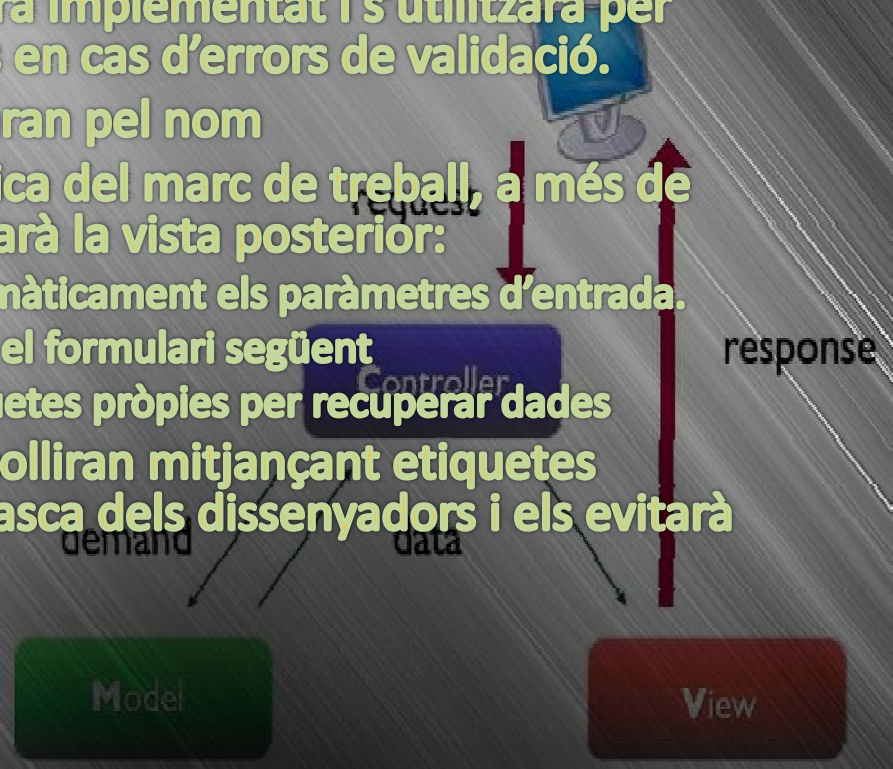
N'hem aprofitat:	
Struts	Accions, Formularis, Servlet
Struts 2	Filtres Interceptors
JSF	Validació i reconstrucció del formulari en cas d'error

FormFaces

introducció al tema

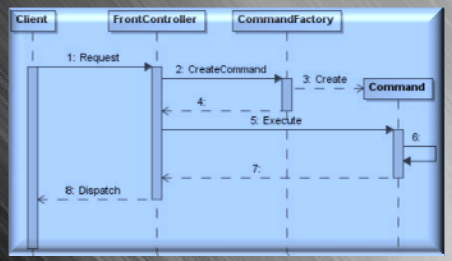
Característiques del Marc de Treball:

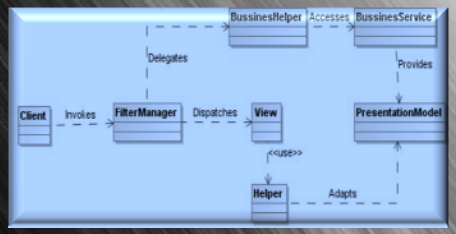
- Rebrà les peticions un Servlet que utilitzarà intercepció de filtres.
- El primer filtre per defecte vindrà implementat i s'utilitzarà per poder reconstruir els formularis en cas d'errors de validació.
- S'utilitzaran accions que es crearan pel nom
- Els formularis seran la peça bàsica del marc de treball, a més de contenir les dades que necessitarà la vista posterior:
  - Recuperarà i validarà automàticament els paràmetres d'entrada.
  - Calcularà automàticament el formulari següent
  - Donarà facilitats a les etiquetes pròpies per recuperar dades
- Les dades dels formularis es recolliran mitjançant etiquetes pròpies que facilitaran molt la tasca dels dissenyadors i els evitarà pràcticament tot el codi



FormFaces introducció al tema

S'han seguit entre d'altres els patrons de disseny:





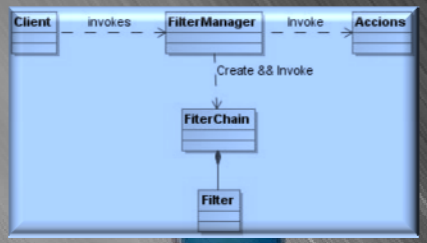
Intercepting Filter

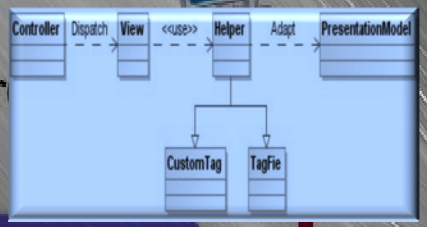
Front Controller

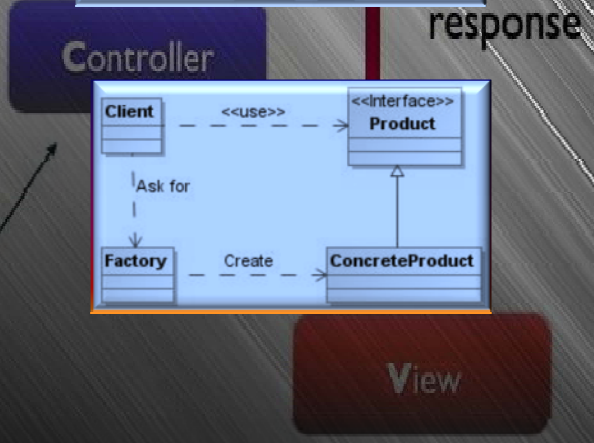
View Helper

Service to Worker

Factory



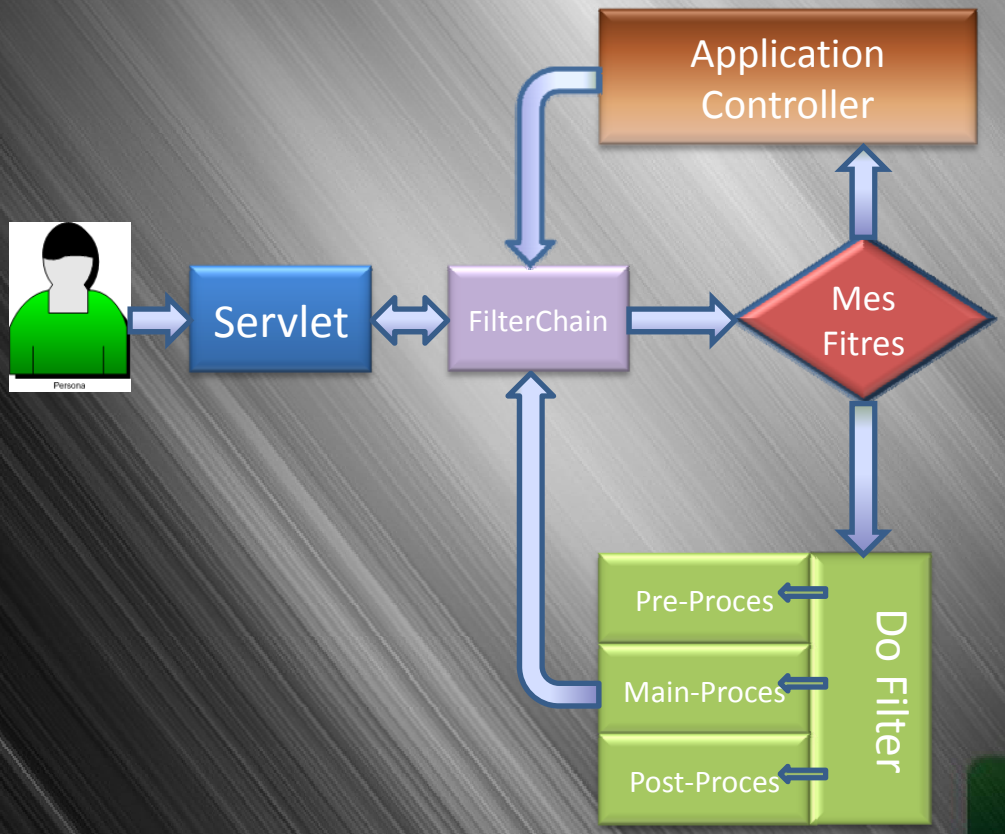




FormFaces

construcció

Intercepting Filter



El Servlet rep la petició i la transfereix a un FilterChain nou al que li passa la llista de filtres.

El FilterChain invoca al filtre següent mentre en queden. Quan s'acaben invoca al Application controller.

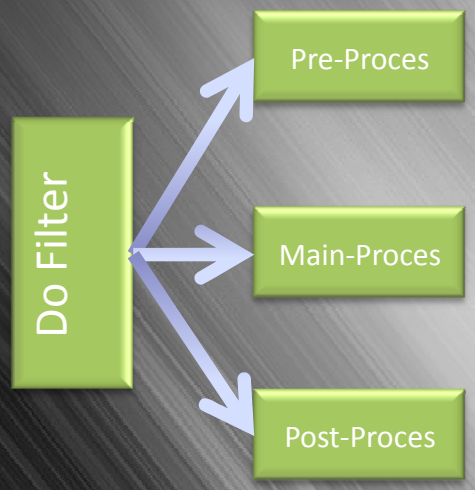
Al invocar als filtres, el FilterChain es passa a ell mateix com a paràmetre.

Els filtres executen primer el pre procés després el main procés i finalment el post procés.
El main procés demana al FiterChain que invoqui al següent filtre. Així primer s'executen tots el pre processos, després la lògica de la aplicació i després els post processos.

FormFaces

construcció

FacesFilter



Recupera el formulari

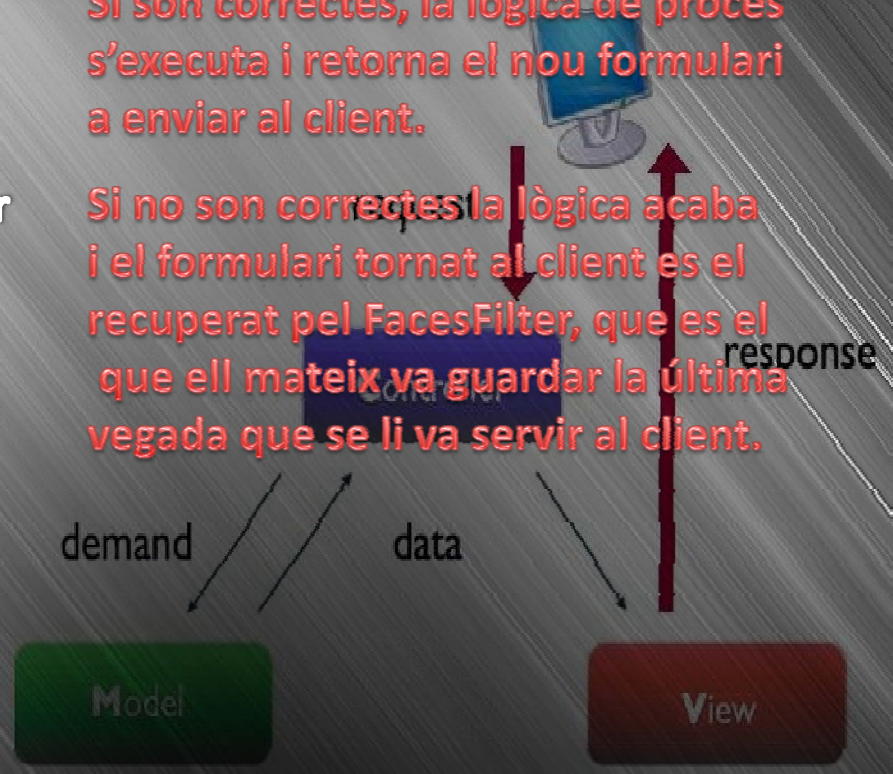
Passa el formulari recuperat al chainFilter

Si el procés ha anat be guarda el formulari Nou.

Al començar la lògica de procés es validen els paràmetres entrats.

Si son correctes, la lògica de procés s'executa i retorna el nou formulari a enviar al client.

Si no son correctes la lògica acaba i el formulari tornat al client es el recuperat pel FacesFilter, que es el que ell mateix va guardar la última vegada que se li va servir al client.



FormFaces

construcció

Application Controller

- Recupera el nom de la acció
- Recupera el ActionForm
- Afegeix els paràmetres d'entrada a l'ActionForm i els valida
- Recupera la acció
- Executa la Acció
- Resol quin es el proper Jsp a enviar al cient
- Recupera el nou JspForm
- Passa les dades del ActionForm al JspForm
- Retorna el nou JspForm i el nou nom de pàgina al ChainFilter

Un error en qualsevol pas fa que el ChainFilter retorni al Servlet el Jsp recuperat per Facesfilter

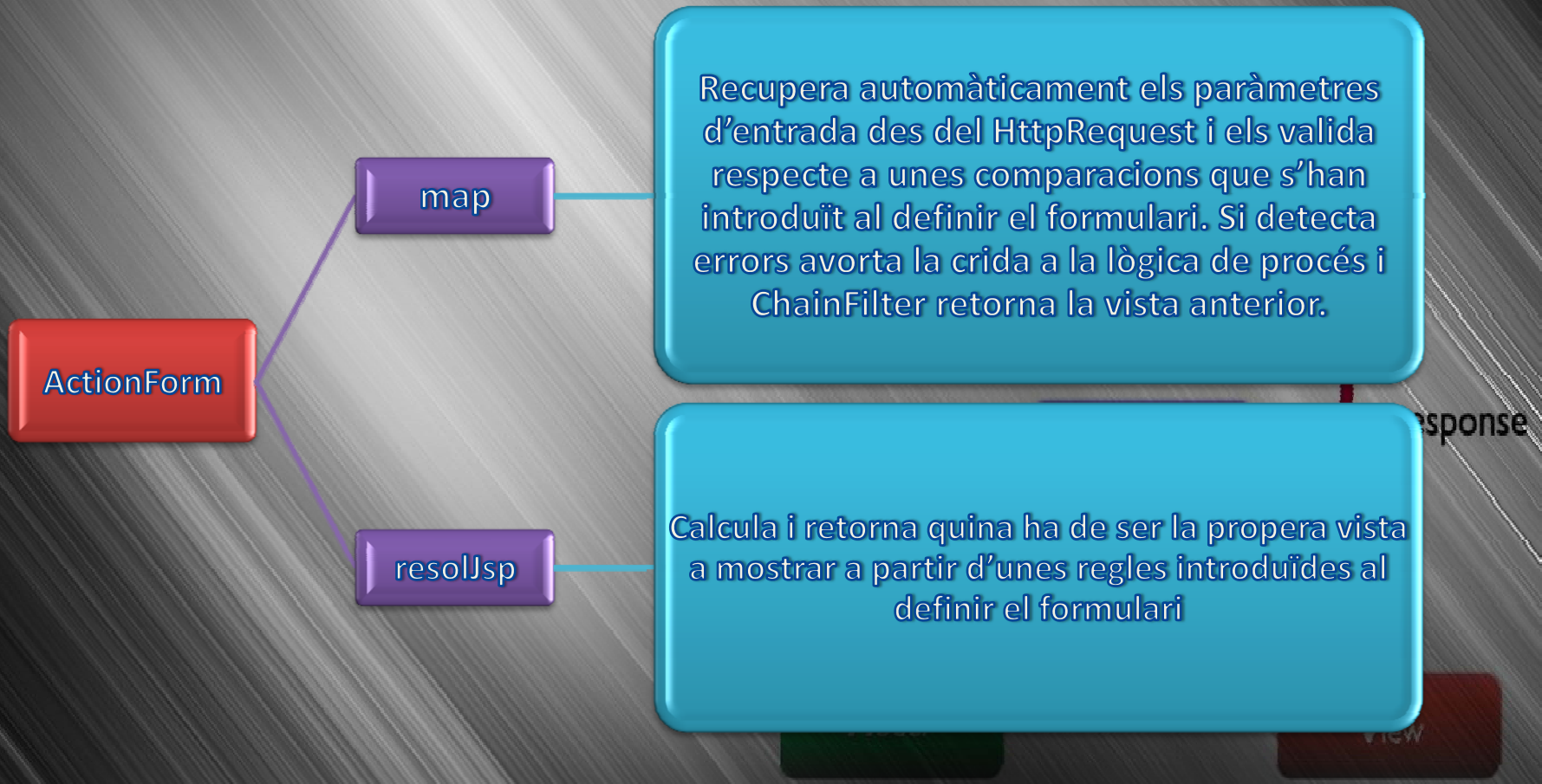
Model

View

FormFaces

construcció

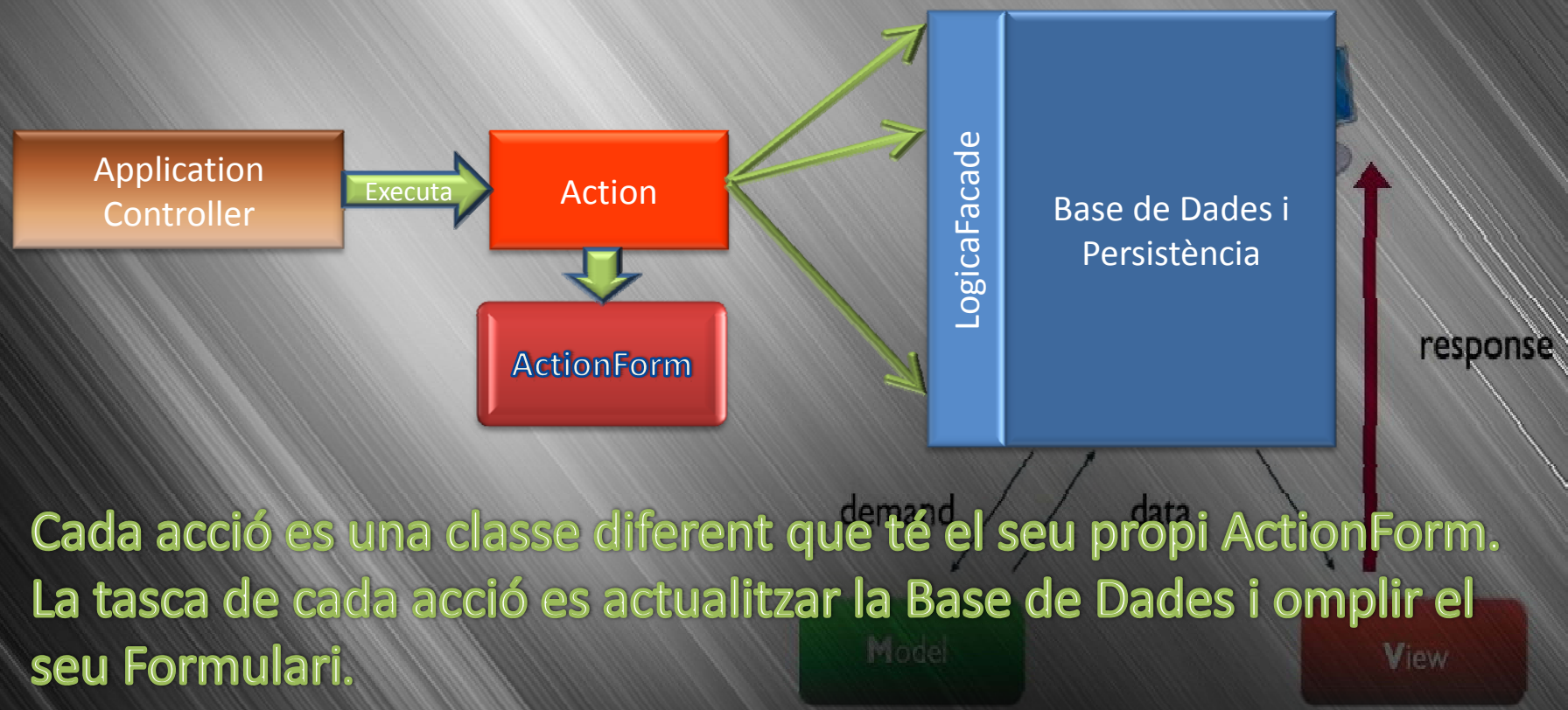
ActionForm es la classe més important: Fa les dues funcions bàsiques del Marc de Treball



FormFaces

construcció

Action es el model, executa en part la lògica de negoci i delega a una classe Facade l'accés a les dades.

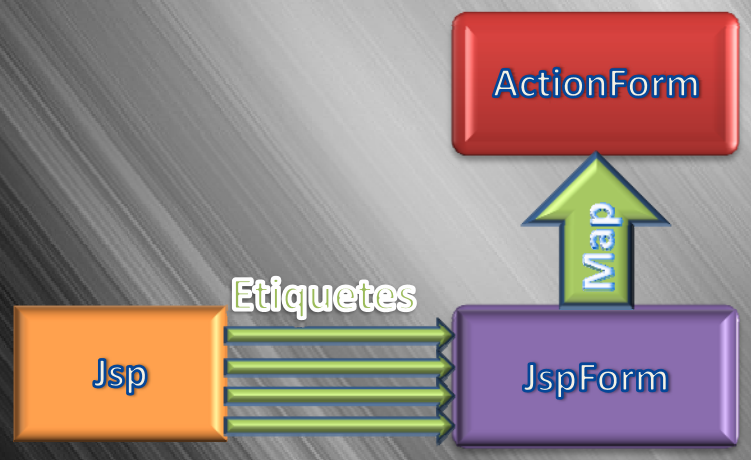


Cada acció es una classe diferent que té el seu propi ActionForm. La tasca de cada acció es actualitzar la Base de Dades i omplir el seu Formulari.

FormFaces

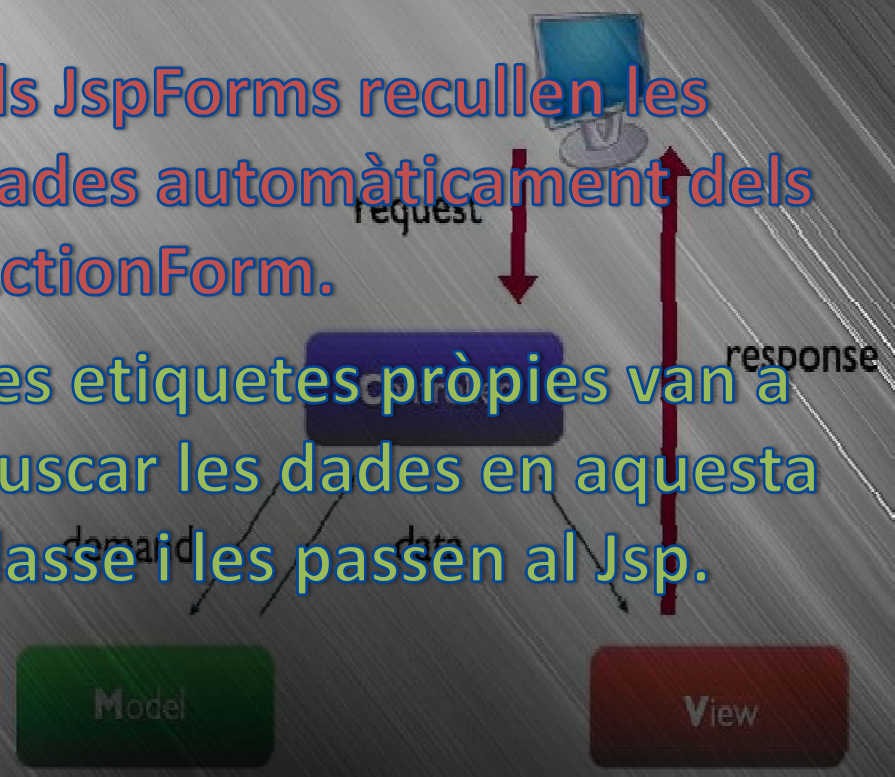
construcció

JspForm es la classe que conté les dades que necessita el Jsp. Cada Jsp té el seu JspForm.



Els JspForms recullen les dades automàticament dels ActionForm.

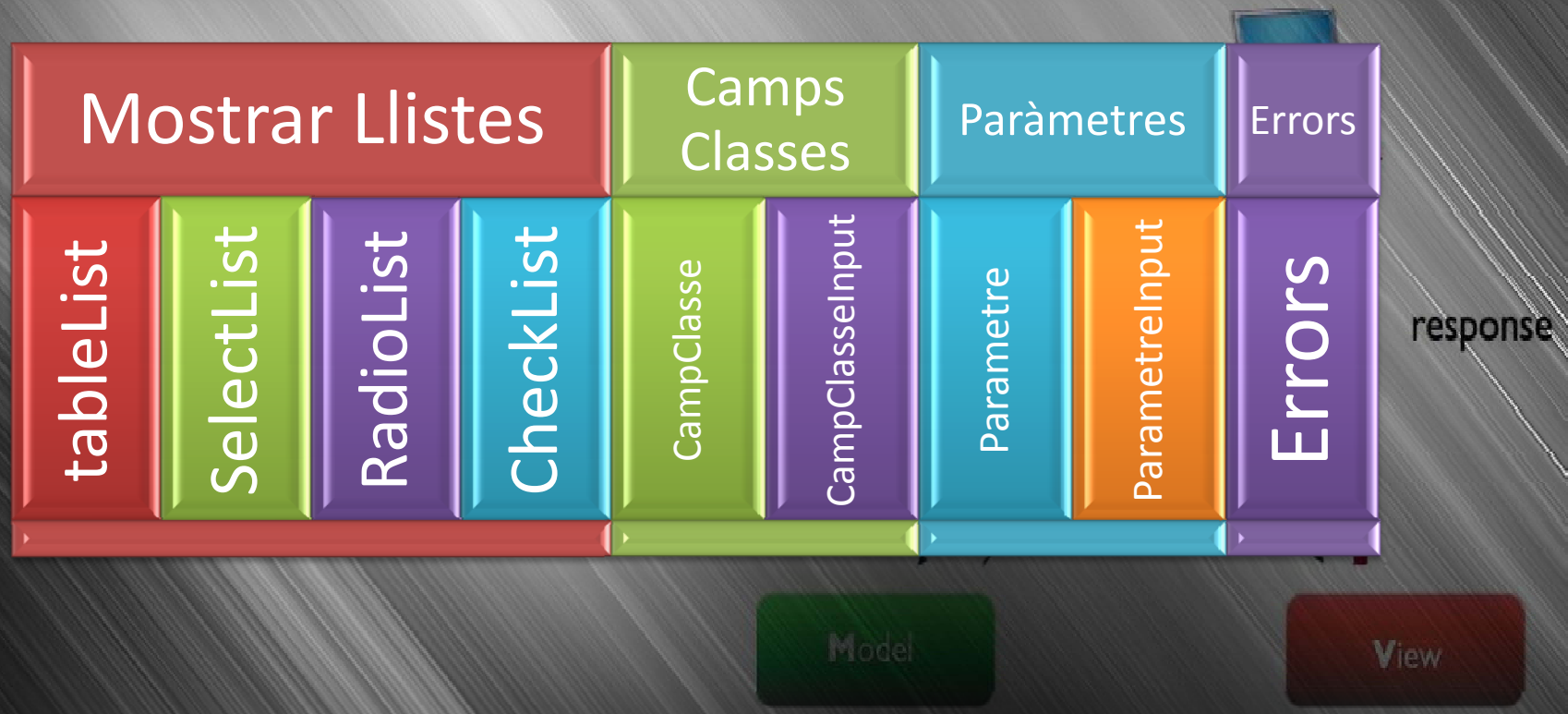
Les etiquetes pròpies van a buscar les dades en aquesta classe i les passen al Jsp.



FormFaces

construcció

Les **etiquetes pròpies** recullen la informació dels JspForm i les posen en els Jsp. Tenim les següents etiquetes:

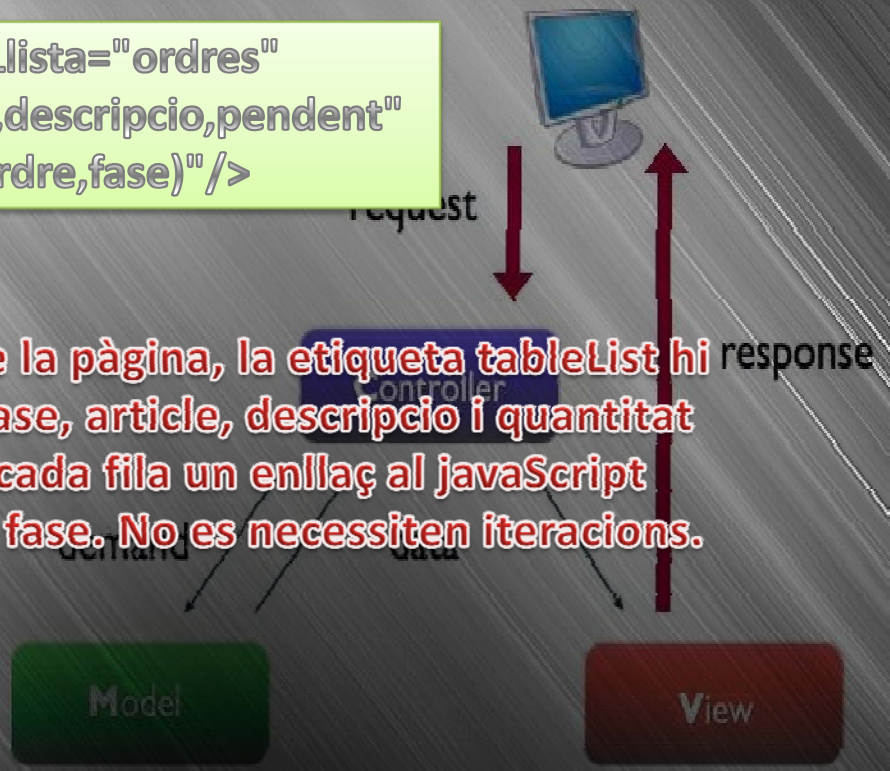


FormFaces

construcció

Les etiquetes pròpies s'han creat per estalviar molta feina als dissenyadors de les pàgines

```
<formfaces:tableList nomLlista="ordres"
camps="ordre,fase,article,descripcio,pendent"
funcio1="selec=seleccio(ordre,fase)"/>
```



Amb aquest codi posat en alguna zona de la pàgina, la etiqueta tableList hi dibuixa una taula amb els camps ordre, fase, article, descripcio i quantitat pendent de la llista "ordres" i afegeix en cada fila un enllaç al javaScript seleccio passant-li els paràmetres ordre i fase. No es necessiten iteracions.

FormFaces

construcció

FormFacesExcepcio es la excepció que utilitza el Marc de Treball per emmagatzemar els errors i poder-los mostrar.

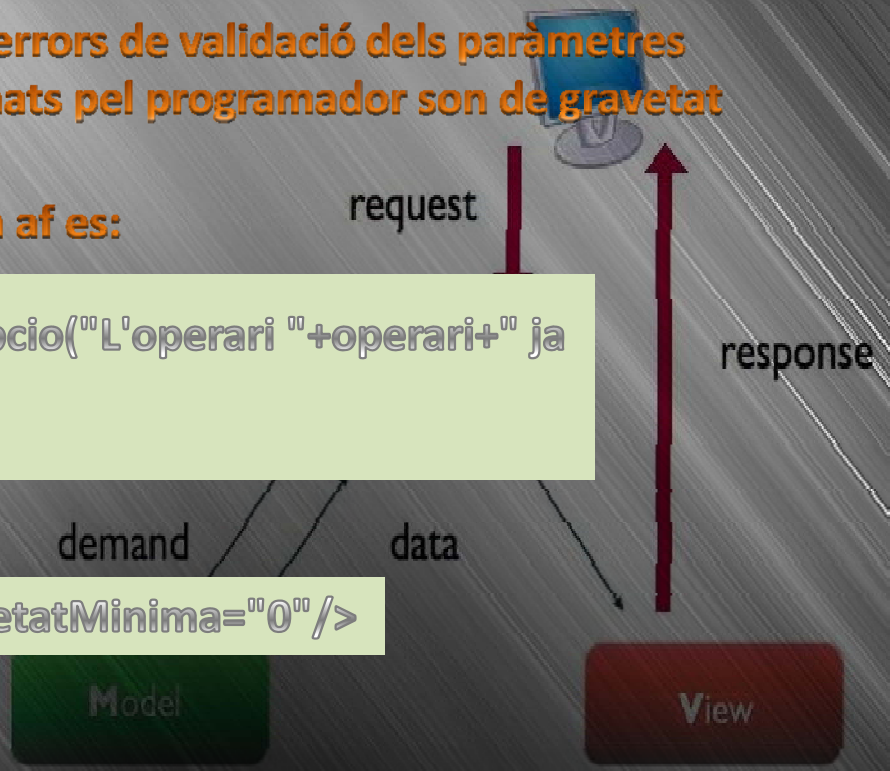
Els errors de disseny son de gravetat 4 i 5, els errors de validació dels paràmetres d'entrada son de gravetat 3 i els errors gestionats pel programador son de gravetat 1, 2 o 3.

El codi per introduir un error en el ActionForm af es:

```
af.novaExcepcio(new FormFacesExcepcio("L'operari "+operari+" ja està assignat a la màquina "+maq,3));
```

El codi per llistar els errors en un Jsp es:

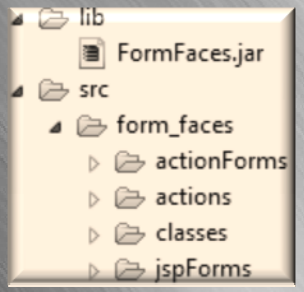
```
<formfaces:Errors gravetatMinima="0"/>
```



FormFaces

Aplicatiu de prova

Per connectar l'aplicatiu de prova al Marc de treball només cal fer:



Incloure la llibreria FormFaces.jar en la que hi ha el Marc de Treball.

Crear la estructura de carpetes form_faces per guardar les classes ActionForm, Action, Classe i JspForm creades.

```
<servlet>
<servlet-name>ServletFilter</servlet-name>
<servlet-class>formFaces.facade.ServletFilter</servlet-class>
<load-on-startup>0</load-on-startup>
</servlet>
<servlet-mapping>
<servlet-name>ServletFilter</servlet-name>
<url-pattern>/ServletFilter/*</url-pattern>
</servlet-mapping>
```

Definir al fitxer web.xm de la aplicació, que el Servlet que utilitzarem es el que ve en el Marc de Treball

Es millor extreure el fitxer formFaces-taglig.tld de la llibreria i ficar-lo dins la carpeta jsp junt amb les pàgines.

FormFaces

Aplicatiu de prova

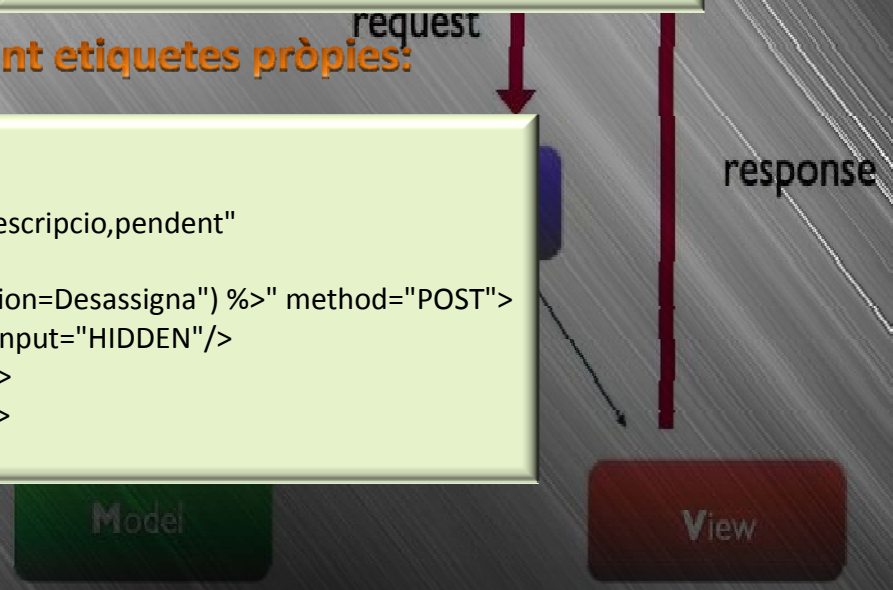
Jsp i JspForm

Per crear els JspForm només cal afegir en el mètode definició: els paràmetres, les classes i les llistes que necessita el Jsp, utilitzant els mètodes addParametre, addClasse i addList

```
public class veureOrdresForm extends JspForm {
        @Override
        protected void definicio() {
                addParametre("maquina", "String");
                addList("operaris", "Operari");
                addList("ordres", "Treball");
        }
}
```

Els jsp agafen les dades dels JspForm mitjançant etiquetes pròpies:

```
<%@ taglib uri="formFaces-taglib.tld" prefix="formfaces" %>
<formfaces:tableList nomLlista="operaris" camps="codi,nom"/>
<formfaces:tableList nomLlista="ordres" camps="ordre,fase,article,descripcio,pendent"
funcio1="selec=seleccio(ordre,fase)"/>
<form action="<%= response.encodeURL("/Captura/ServletFilter?action=Desassigna") %>" method="POST">
<table><formfaces:ParametreInput nomParametre="maquina" tipusInput="HIDDEN"/>
        <input type="HIDDEN" name="idForm" value="veureOrdres">
        <input type="submit" border="0" value="Finalitzar"> </table>
</form>
```



FormFaces

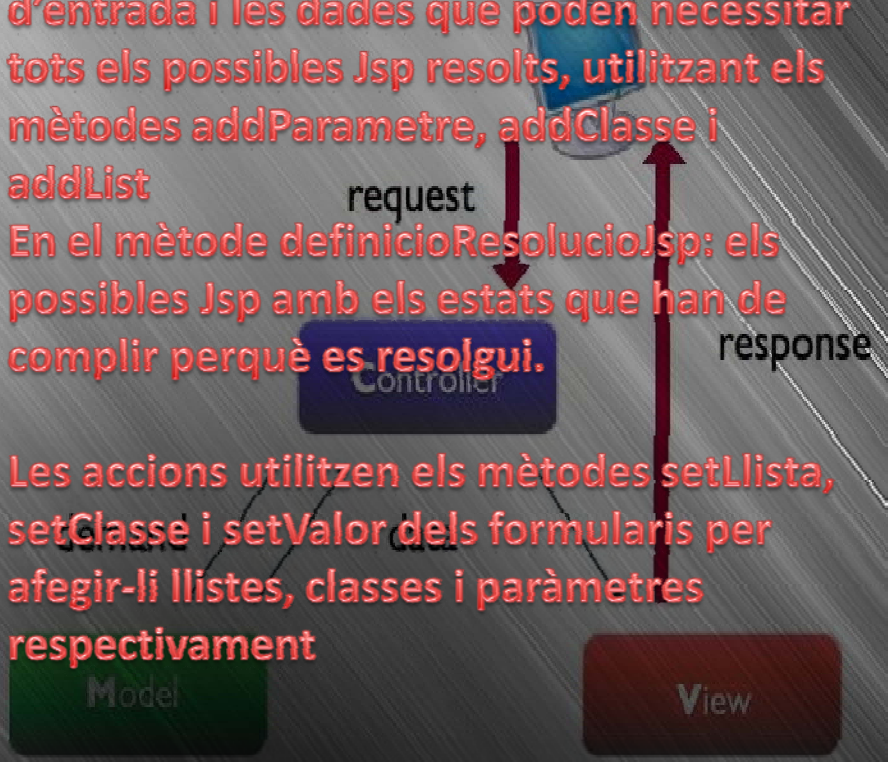
Aplicatiu de prova

Action i ActionForm

```
public class SelMaquinaForm extends ActionForm {
        @Override
        protected void definicio() {
                addParametre("maquina", "String");
                addList("operaris", "Operari");
                addClasse("treball", "Treball");
                addList("components", "Component");
                addList("ordres", "Treball");
        }
        @Override
        protected void definicioResolucioJsp() {
                addJsp("treballOrdre");
                addJspStatus("llista", "operaris");
                addJspStatus("campClasse", "treball.ordre");
                addValidacioStatus("gt", (long)0);
                addJsp("veureOrdres");
                addJspStatus("llista", "operaris");
                addJsp("assigOperaris");
        }
}
```

Per crear els ActionForm només cal afegir:
En el mètode definició: els paràmetres d'entrada i les dades que poden necessitar tots els possibles Jsp resolts, utilitzant els mètodes addParametre, addClasse i addList
En el mètode definicioResolucioJsp: els possibles Jsp amb els estats que han de complir perquè es resolgui.

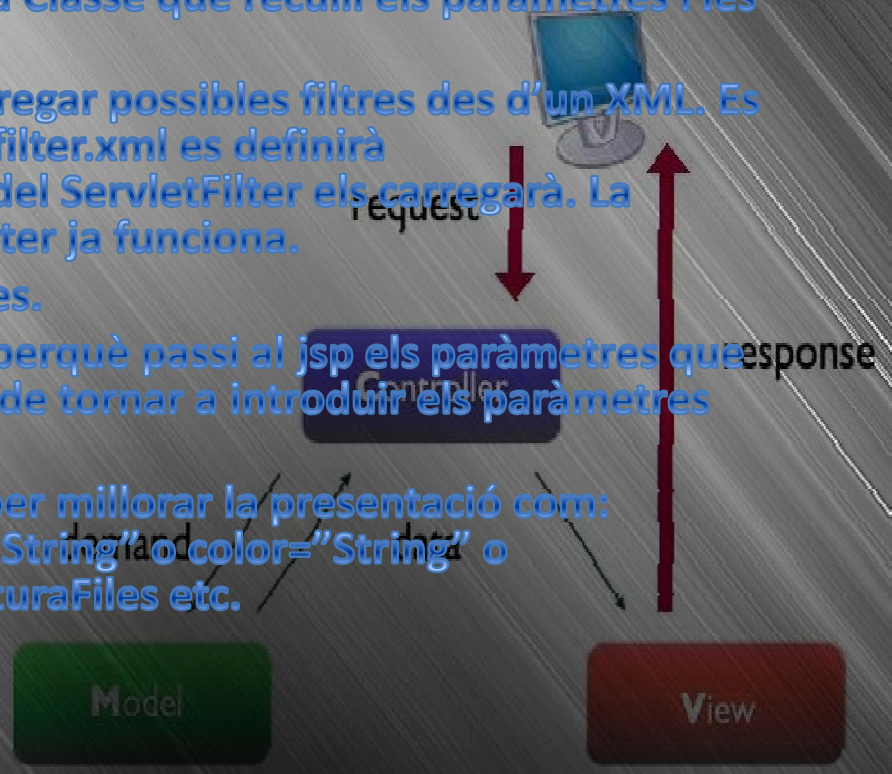
Les accions utilitzen els mètodes setLlista, setClasse i setValor dels formularis per afegir-li llistes, classes i paràmetres respectivament



FormFaces

Millores a Fer

- Queden moltes millores a fer que no dóna temps d'implementar. Enumerem algunes millores que hem anat pensant durant la construcció i les proves i que posarem en pràctica en un futur proper:
  - Tractament de classes: El framework ha de poder tractar les classes normals que s'utilitzen a la lògica de negoci. Una forma senzilla d'implementar-ho es construint un map a la classe abstracta Classe que reculli els paràmetres i les llistes de les classes de la lògica.
  - Filtres: S'ha d'acabar el sistema de carregar possibles filtres des d'un XML. Es una funcionalitat molt senzilla: en un filter.xml es definirà <filter>classe</filter> i el constructor del ServletFilter els carregarà. La funcionalitat perquè passin al ChainFilter ja funciona.
  - Ampliar els tipus de camp en les classes.
  - Donar més potencialitat al FacesFitre perquè passi al jsp els paràmetres que passen en el request, així no s'hauran de tornar a introduir els paràmetres bons.
  - Millorar les etiquetes: posar atributs per millorar la presentació com: colorColumna="String",String", ……,"String" o color="String" o ampleColumna= int, int, ……, int. O alturaFiles etc.



FormFaces

Conclusions

En aquest projecte s'ha **vist com es pot construir** un entorn per facilitar la feina dels desenvolupadors d'aplicacions. S'ha **vist amb quina facilitat** es pot construir una aplicació web utilitzant el Framework, a més, es **construeix seguint un bon disseny.**

Hem estudiat els millors **exemples de Framework de** presentació que hi ha al mercat: Struts, Struts2, Spring i JSF. **De tots n'hem tret coses interessants,** d'entre elles cal citar:

De Struts 2 n'hem **tret el concepte dels filtres que es molt** potent per donar funcionalitats generals a totes les **peticions.**

De Struts n'hem **obtingut el concepte de les accions i dels formularis** que faciliten molt el disseny del patró MVC.

De JSF n'hem tret el **concepte de la validació dels paràmetres** d'entrada i tornar el jsp sense haver de cridar la lògica per **reconstruir-lo.**

Hem aprofitat la potencia **del disseny d'etiquetes pròpies** per encapsular les classes del framework i facilitar la feina als **dissenyadors.**

Mentre l'hem dissenyat hem **estudiat patrons** de disseny, gràcies a ells hem aconseguit una arquitectura ben dissenyada i de **fàcil** manteniment. L'estudi dels patrons ens ha proporcionat maneres elegants de resoldre problemes complexos amb els que ens hem trobat.

Ha estat un treball realment interessant que anirem millorant i espero sigui de mota utilitat.