



Usare Spring a pieno regime con WebFlow

Davide Del Vecchio (d.delvecchio@k-tech.it)



Agenda

- Che cos'è Spring
- Spring nelle applicazioni enterprise
- Definizione di flussi web
- Deploy del flusso nel contesto di una webappl
- Integrazione con tecnologie preesistenti
- Conclusioni



Questo talk è indirizzato a tutti i gli sviluppatori alla ricerca di un framework più razionale per le applicazioni enterprise

Si propone di mostrare l'utilizzo di Spring prendendo ad esempio lo sviluppo di applicazioni web



Spring ancora un altro framework....

- Spring è un framework “leggero” per applicazioni enterprise java
- L’aggettivo è riferito al minimo impatto architetturale/implementativo necessario per accedere alle funzionalità del framework e alla possibilità di utilizzo in qualsiasi fase del ciclo di vita di un progetto



J2EE != EJB

- I componenti EJB hanno degli aspetti decisamente svantaggiosi
 - Obbligano lo sviluppatore a legarsi ad un particolare insieme di API modificando la struttura delle classi di business al fine di ottenere vantaggi infrastrutturali
 - Sono difficili da testare



J2EE != EJB

- Non è indispensabile ricorrere ad EJB per realizzare applicazioni java enterprise
- Spring mette a disposizione dello sviluppatore tutti i vantaggi di un container EJB ma lo rende libero di sviluppare semplici classi java (Plain Old Java Objects) POJO



I due elementi fondamentali

Lo sviluppatore viene incoraggiato ad
avvalersi di due aspetti della
programmazione OO

Programmazione per interfacce
Dependency Injection



Dependency Injection

- Il runtime di Spring associa ad ogni oggetto le sue dipendenze sulla base delle definizioni contenute in un documento di configurazione.
- L'oggetto è fin da subito legato al contesto in cui opera e non deve preoccuparsi di istanziare tutte le risorse a cui fa riferimento.

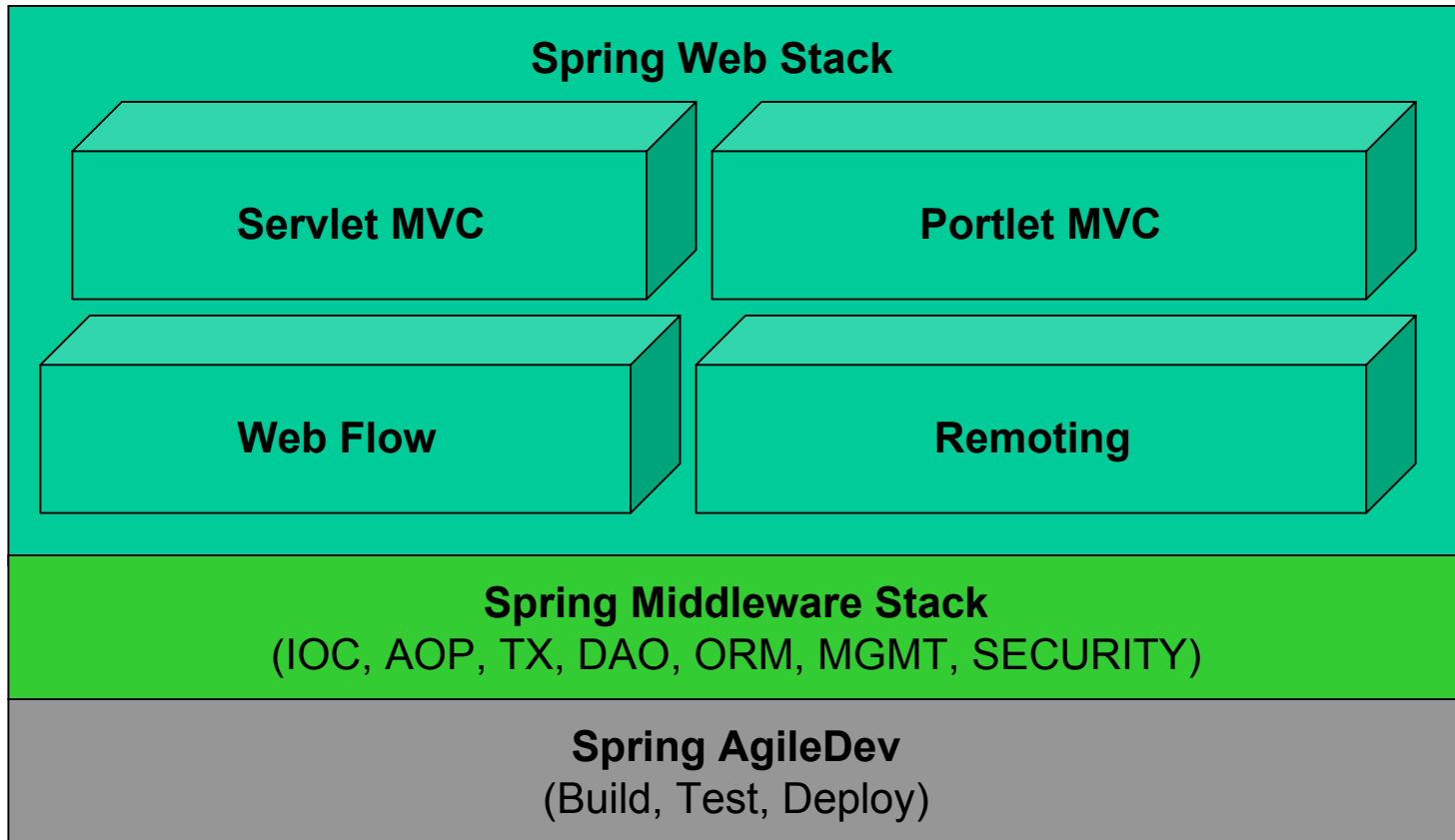


Inversion of control

- La D. I. è un particolare caso di Inversion of Control.
- Tale espressione viene generalmente illustrata dal principio di Hollywood “non chiamare noi saremo noi a chiamarti”
- Tutti i framework adoperano una qualche forma di IoC non solo Spring



Uno stack completo per applicazioni enterprise





Un caso di studio SWF





Definizione del problema

- Voglio definire un flusso di navigazione attraverso le pagine
 - Transazione applicativa che si estende a più pagine
 - Alcuni nodi richiedono l'intervento di un utente
 - Altri nodi eseguono della logica decisionale
 - La navigazione è controllata



Come viene realizzato in Struts

1. Creazione di una `ActionForm` per ospitare i dati utente
2. Definire una `JSP` per ciascun passo
3. Definire una `Action` per ciascun passo
4. Esporre ciascuna `Action` ad una URL della request
5. Renderizzare la form mediante la `JSP`
6. Delegare l'effettivo commit della transazione ad un servizio di back-end



Problemi per questo tipo di approccio

- Centrato sulla Request (non si osserva nessuno stato conversazionale)
- Controllo dello stato programmatico (non dichiarativo)
- Difficile aggiungere altre finestre
- Difficile gestire il back
- Navigazione debolmente controllata
- Difficile rendersi conto del ciclo di vita del processo
- Si può avere accoppiamento tra il controller e le operazioni da eseguire
- Legato alle specifiche del protocollo



E' un caso tipico di uso di Spring

- Abbiamo trovato degli elementi insoddisfacenti negli strumenti che adoperiamo.
- Utilizziamo Spring per estenderli e solo in caso lo ritenessimo utile possiamo sostituirli del tutto
- Non implica un cambio radicale

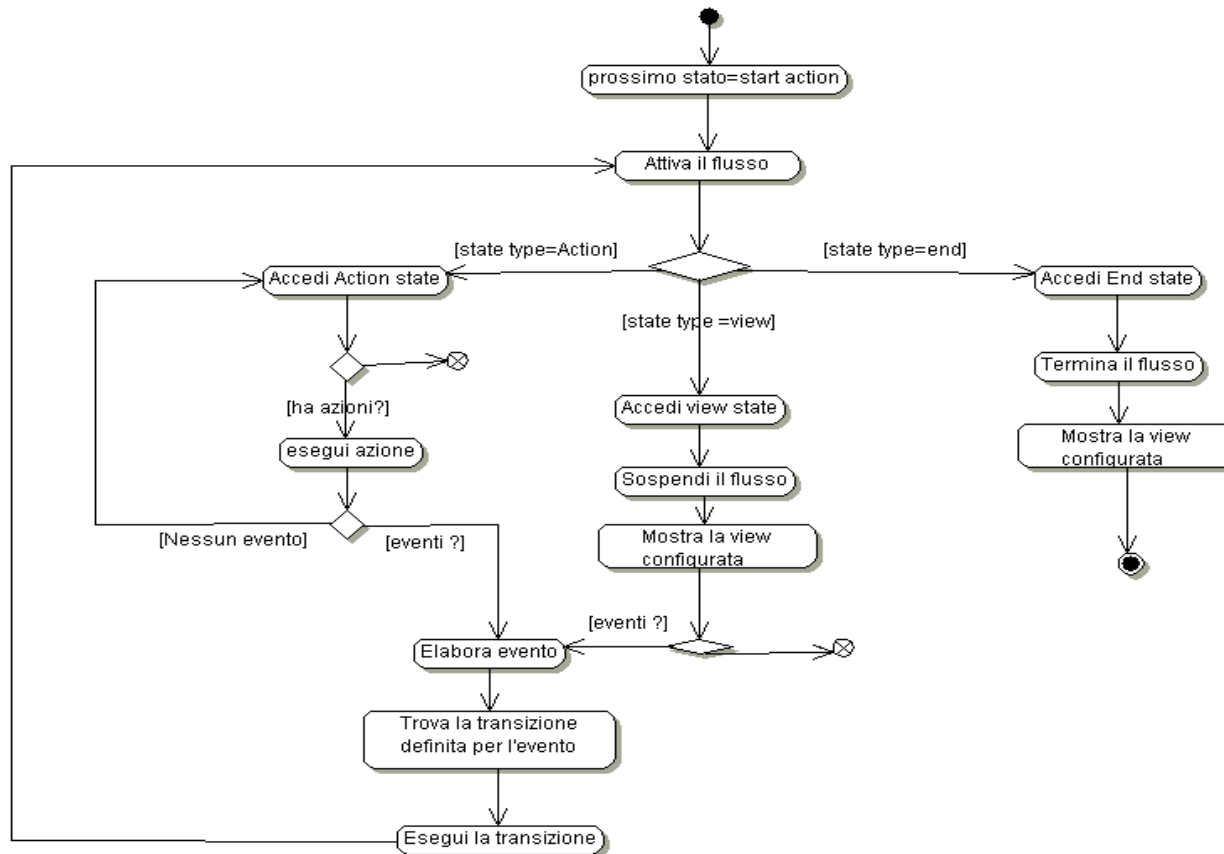


Il Web Flow

- La rappresentazione di un flusso definisce una macchina a stati finiti
- Gli elementi costitutivi sono gli stati
- I possibili stati i seguenti
 - Stati View che attendono risposta dall'utente
 - Stati Action eseguono dei comandi
 - Stati Subflow attivano altri flussi
 - Stati End terminano il processo
- Eventi predefiniti innescano la transizione tra i diversi stati



Activity Diagram per il flusso





Scegliamo il modo per rappresentarlo

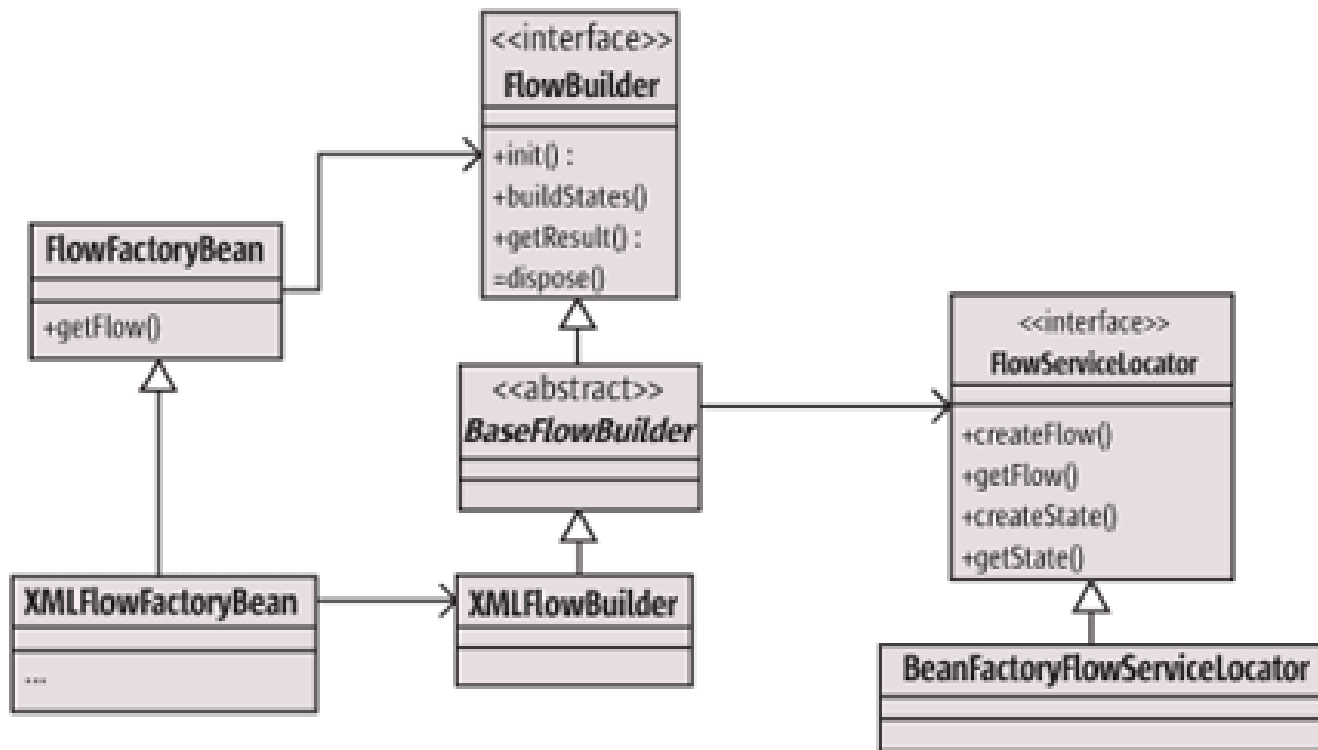
- La Dependency Injection ci consente di scegliere in quale modo rappresentare il flusso

```
<bean id="MyFlow"  
class="org.springframework.web.flow.config.XMLFlowFactoryBean">  
<property name="location" value="classpath:myFlow.xml" />
```

La classe XMLFlowFactoryBean è predefinita e ci consente di leggere la definizione da un file xml



Posso cambiare rappresentazione





Definizione del flusso

```
<webflow id="itemlist" start-state="displayItemlist">
  <view-state id="displayItemlist" view="itemlist">
    <transition on="add" to="createItem"/>
  </view-state>
  <action-state id="createItem">
    <action bean="createItemAction"/>
    <transition on="success" to="displayItem"/>
  </action-state>
  <view-state id="displayItem" view="item">
    <transition on="submit" to="addItem"/>
  </view-state>
  <action-state id="addItem">
    <action bean="addItemAction"/>
    <transition on="*" to="displayItemlist"/>
  </action-state>
</webflow>
```



Definizione per NewItemAction

```
import org.springframework.webflow.RequestContext;
import org.springframework.webflow.action.AbstractAction;

public class NewItemAction extends AbstractAction {
    protected Event doExecute(RequestContext context) throws Exception {
        // begin transactional processing
        context.beginTransaction();
        return success();
    }
}
```



Deploy del WebFlow

```
<beans>
  <bean name="/itemlist.htm"
    class="org.springframework.webflow.mvc.FlowController">
    <!-- serialize all operations within a session -->
    <property name="synchronizeOnSession" value="true"/>
    <property name="flow" ref="itemlist"/>
  </bean>
  <bean id="itemlist"
    class="org.springframework.webflow.config.XmlFlowFactoryBean">
    <property name="location" value="/WEB-INF/itemlist-flow.xml"/>
  </bean>
  <bean id="createItemAction"
    class="org.springframework.webflow.samples.itemlist.NewItemAction"/>
  <bean id="addItemAction"
    class="org.springframework.webflow.samples.itemlist.AddItemAction"/>
</beans>
```



Lanciare un flow da una form

```
<form method="post" action="itemlist.htm"/>
```

...

```
<input type="hidden" name="_flowId" value="itemlist"/>
```

```
<input type="submit" value="Submit"/>
```

```
</form>
```

Anchor

```
<a href="<c:url  
value="/itemlist.htm?_flowId=itemlist&src.item=MLB&  
dest.item=ATL"/>">
```

```
</a>
```



View che partecipano al flusso

```
<form method="post" action="intemlist.htm"/>  
  <input type="text" name="firstName"/>  
  <input type="text" name="lastName"/>  
  ...  
  <input type="hidden" name="_flowExecutionId"  
    value="{flowExecutionId}"/>  
  <input type="submit" name="_eventId_submit"/>  
</form>
```

```
<a href="<c:url  
value="/springair.htm?_flowExecutionId={flowExecutionId}&  
_eventId=submit&firstName=Keri&lastName=Donald/>">Enter  
Passenger Information</a>
```



Un Web Flow Transazionale

- Mediante l'introduzione del `flowExecutionId` l'applicazione Web diventa a tutti gli effetti stateful ed è possibile continuarla nel tempo
- Mediante i nodi `end` è possibile terminare la transazione e evitare submit doppi



Caratteristiche future

- Rendere JMX enabled il management del WF
- Ajax support
- Ulteriori default per la configurazione
- Definizione di sub WF asincroni paralleli



Ringraziamenti

- Desidero ringraziare Stefan Dijvievier e Stefano Usai per utili discussioni sull'argomento
- Ho beneficiato molto del lavoro in gruppo e di spiegazioni da parte di Rob Harrop