Taccuino:	Agile		
Creato:	09/10/2019 09.24	**Aggiornato:**	03/01/2020 10.35
Autore:	Domenico Fico		

Session State Patterns

Vediamo qualcosa che è connesso con il layer dell'organizzazione e della presentazione web.
Ci sono tre pattern intorno i quali sono organizzati i Session State Patterns:

- Client Session State;
- Server Session State;
- Database Session State;

Questi pattern ci vengono incontro quando vogliamo simulare la presenza di una sessione che tiene traccia degli utenti e delle loro richieste.
Come implementare questa sessione continua? Con queste tre principali soluzioni.

Client Session State

"Memorizza lo stato della sessione sul client"
Tutte le informazioni riguardo la sessione sono memorizzate sul client. Il server che gestisce le nostre richieste non sa nulla a nostro riguardo. Il server è detto stateless. C'è un modo largamente usato per implementare il CST: *Cookies*.
Anche il sistema più *server-oriented* possibile necessita di almeno un piccolo CTS che memorizza solamente l'id della sessione.
Come funziona CST?
Immaginiamo di avere ogni volta una variabile il cui valore è un url che riporta le nostre informazioni memorizzate sulla pagina (come le credenziali o il nostro id), questa variabile viene rispedita al server e il server genera la pagina in base alle nostre informazioni.
Dal punto di vista del server, quella variabile corrisponde al nostro stato. E grazie a questo il server sarà completamente stateless.
Si sposa bene con il DTO, usato per gestire il trasferimento dei dati.
Nota: implementata già dal Web 1.0.
Quando usare CST?
Quando il sistema prevede l'utilizzo di rich-clients, il CST può essere implementato senza difficoltà.
Quali sono i vantaggi?
Il server può essere stateless senza problemi, ciò implica che possiamo avere un grande numero di server che possono gestire lo stesso tipo di richieste e possono essere interscambiati. E questa architettura è molto scalabile proprio per questo motivo.
Quali sono gli svantaggi?
I problemi nell'uso del CST aumentano all'aumentare della mole di dati da interscambiare. Fino a quando sono pochi byte da memorizzare, tutto filerà liscio. Ma con dati di grandi dimensioni sorgeranno i primi problemi: "dove

memorizzare i dati?" e "quanto costerà in termini di tempo il trasferimento dei dati?". Sarà proibitivo.
Un altro grosso problema riguarda la sicurezza. Ogni dato in viaggio tra client e server può essere intercettato e manomesso. L'unico modo per risolverlo è quello usare la crittografia, ma criptare e decriptare i dati costerà molto in termini di performance.

Server Session State

"Memorizza lo stato della sessione sul server in forma serializzata"
Lo stato della sessione è memorizzato nella memoria del server, ciò implica che il client non deve avere requisiti particolari.
Se il server crasha la sessione è persa. E' poco scalabile, poiché visto che le info sono tutte nel server e il server deve essere abbastanza potente da gestire tutte le richieste.
Come funziona SST?
Può essere implementato come una mappa memorizzata in memoria che mantiene tutti gli oggetti sessione con chiave *session-id*.
Il client farà richiesta al server fornendo il proprio session-id e il server recupererà la sessione dalla mappa.
Il server deve avere abbastanza memoria perché in caso di fallimento del server la sessione sarà persa.
Come implementare la persistenza e la storia delle sessioni? Tramite il memento. Il memento può essere serializzato come JSON (o in altri modi) e salvato sul server.
Quando usare SST?
Il più grande appeal del SST è la sua semplicità, anche perché la sua implementazione non necessita di programmazione. Il suo funzionamento è molto legato alle caratteristiche hardware del server, una su tutte, la memoria.

Database Session State

"Memorizza lo stato della sessione in dati affidati al database"
Migliora memorizzazione della sessione: questo compito ora può essere svolto da più server.
Le nostre informazioni riguardo la sessione sono memorizzate su un database.
Quando arriva una chiamata dal client al server bisogna prendere i dati richiesti dal db, fare il lavoro richiesto, risalvarli sul db.
Ovviamente c'è un prezzo da pagare: la latenza.
Vantaggio: se il lato server crasha, la nostra sessione resterà intatta e memorizzata sul db.
Come funziona DST?
Qualche informazione aggiuntiva richiede di essere salvata altrove, infatti il *session-id* è memorizzato sul client.
Come separare i dati della sessione? Di solito i dati sono un mix di dati in locale utili per le interazioni correnti e dati che sono già stati affidati al db e che sono utili per ogni interazione.
Esempio: se sto lavorando su un ordine in una sessione e ho bisogno di salvare lo stato intermedio sul db, questi dati devono essere trattati in modo diverso rispetto a quelli generati alla conferma dell'ordine.

Modificando la struttura delle tabelle nel db aggiungendo campi booleani (*isInPending*) o altri tipi di informazione sarebbe una soluzione costosa ed invasiva che richiederebbe una quasi completa reingegnerizzazione del sistema perché andrà a toccare tabelle importanti che sono già molto usate nel sistema.
Una migliore soluzione può essere quella di inserire una nuova tabella che salverà solamente i dati che sono in pending e che verranno cancellati quando diventeranno dati completi e definitivi (salvandoli nella tabella reale). Queste *pending-tables* dovrebbero essere tabelle cloni di quelle reali che variano per il campo *session-id*.
Ovviamente c'è bisogno di un metodo che vada a pulire le pending-tables dalle sessioni abbandonate o scadute/perse.
La sessione può essere serializzata tramite Lob o Json e salvata sul db.
Quando usare DST?
E' un'alternativa per gestire lo stato della sessione.
Dipende dalla performance: si guadagnerà l'uso di oggetti stateless, ma si pagherà il tempo necessario a gestire i dati del db. I tempi possono ridursi utilizzando il caching per la lettura, ma non per la scrittura.

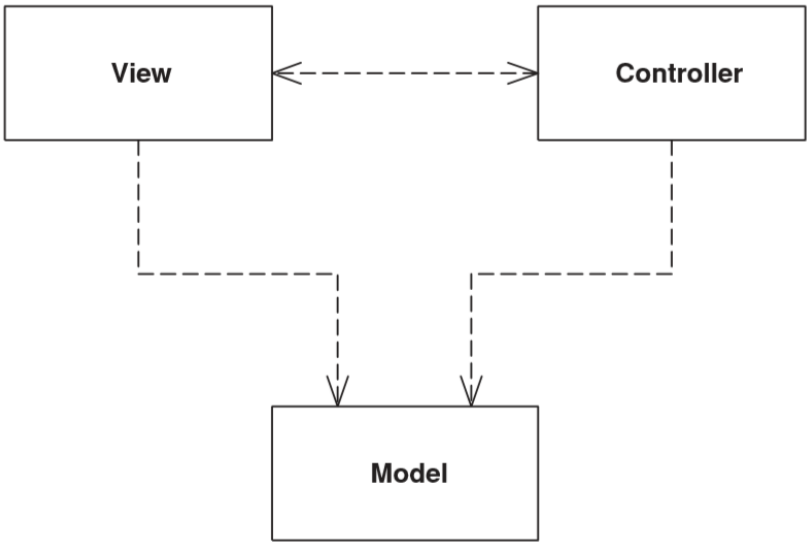
Nota: Nelle applicazioni moderne per gestire lo stato della sessione avviene una combinazione di queste tre tecniche.

Web Presentation Patterns

Model View Controller
"*Suddivide l'interazione dell'interfaccia utente in tre ruoli distinti*"
L'idea è di separare la vista dai modelli e dal controller.

- *View* si occupa di mettere a display l'ui.
- *Model* corrisponde al domain logic.
- *Controller* è l'insieme delle procedure che controllano il flusso delle interazioni dell'utente con il sistema.

Questa separazione lavora molto bene con il web (*View* -> html; *Model* -> db; *Controller* -> servlets).
Come funziona il MVC?
Il Model è un oggetto che rappresenta le informazioni riguardo il dominio
La View rappresenta la visualizzazione dei modelli nell'ui
Il Controller legge gli input, manipola i modelli e aggiorna la vista
Separazione Model dalla View
In genere è implementata con l'*Observer* in cui nel momento del cambiamento del model vengono notificate le view.
Separazione View dal Controller
E' meno importante e meno comune ma può essere implementata con la *Strategy*.


View
Controller
Model

Taccuino:	Agile		
Creato:	15/10/2019 10.55	**Aggiornato:**	03/01/2020 12.48
Autore:	Domenico Fico		

Page Controller

"E' un oggetto che gestisce un'azione o una richiesta per una specifica pagina di un sito"

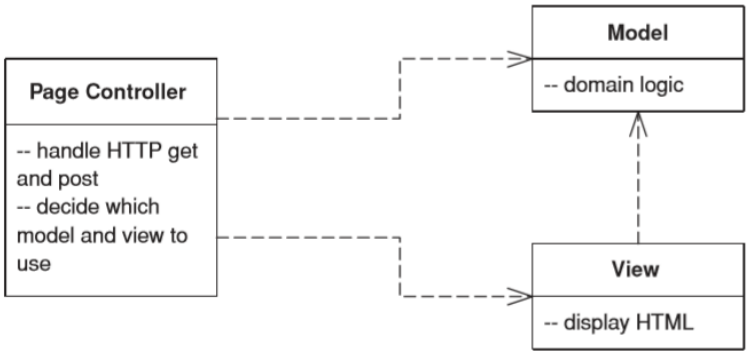
Per ogni evento generato, cattura l'input, decide quale procedura usare e manda il risultato alla ui. Per ogni pagina logica del sito esisterà un controller che può essere posto nella pagina stessa o essere un oggetto separato.

Come funziona PC?

L'idea di base è quella di decidere qual è la prossima pagina.
Riceve l'input, analizza i parametri, applica le eventuali modifiche al business logic e le ritorna alla UI.
Il PC può essere strutturato o come uno script in una servlet o come una pagina JSP sul server, o un mix di questi.

Quando usare PC?

E' il modo più semplice di preparare il controller di una applicazione MVC ed è usatissimo per siti molto semplici con poche pagine.
Un altro metodo molto utilizzato è il *Front Controller* e spesso vengono mixati.



Front Controller

"Un controller che gestisce tutte le richieste per un sito web"

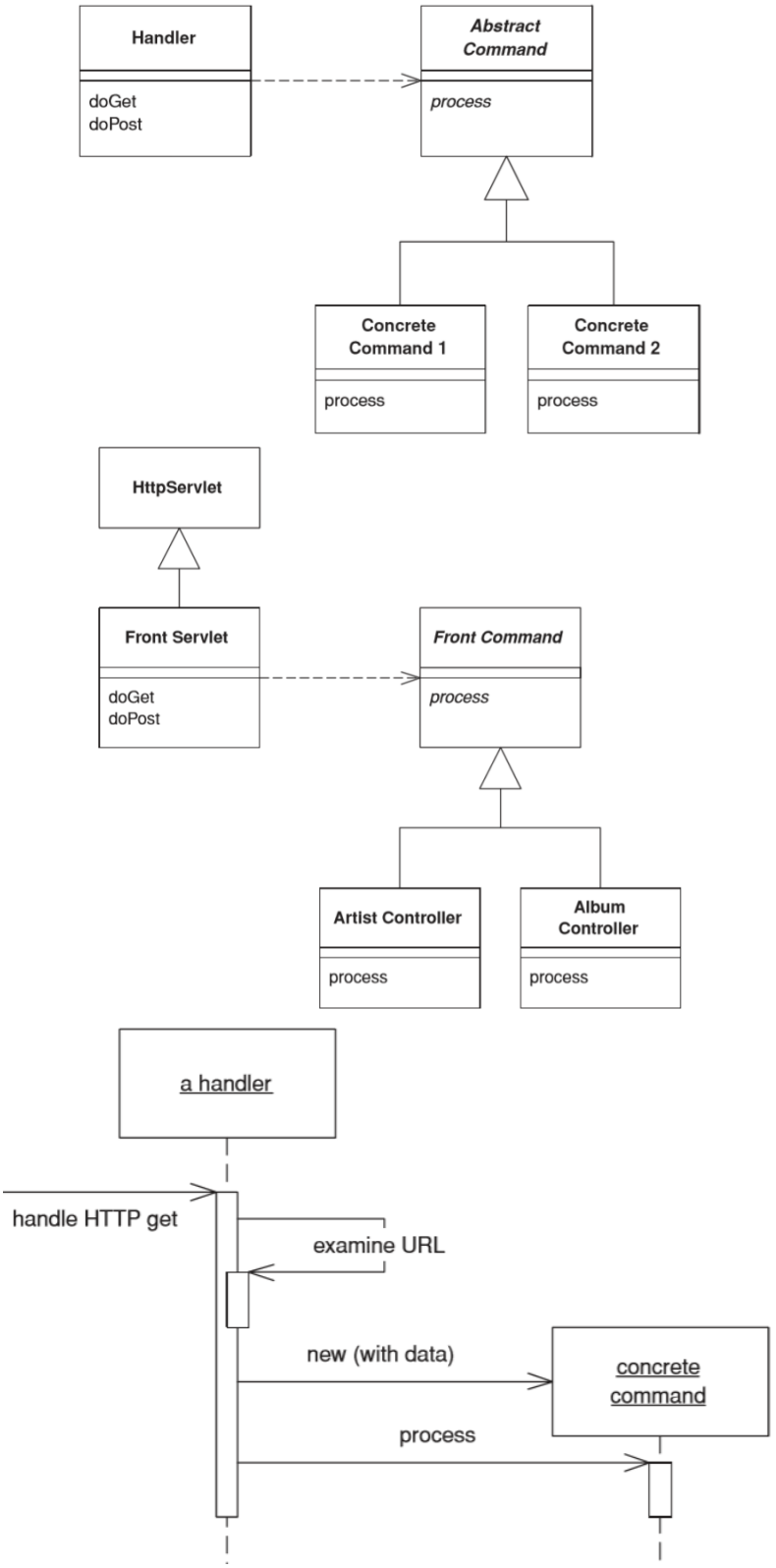
Consolida la gestione delle richieste canalizzandole verso un singolo oggetto gestore.
Questa soluzione risulta essere meno dipendente dalla configurazione del server e molto più scalabile, ma come al solito è molto più complicato da implementare.
Non c'è bisogno di creare un controller per ogni richiesta come il PC perché il FC gestisce tutte le richieste del website fornendo un unico punto d'accesso per ogni pagina.
Nel Handler vengono integrate delle istanze del Command che usa per ogni azione diversa.

Come funziona FC?

E' strutturato in due parti:

- Un gestore delle richieste web;
- Una gerarchia di Commands.

L'Handler è l'oggetto a cui effettivamente arriva la richiesta e delega un *Command* per portarla a compimento.
FC è molto spesso una classe e non una web page dove è presente un grande switch-case (o magari un handler dinamico come *Strategy*).
E' anche possibile un'implementazione a due livelli basata su *Decorator*.
Quando usare FC?
E' molto complicato rispetto al PC, ma è molto flessibile grazie ai parametri che riceve dall'esterno.
Complica l'implementazione, ma semplifica il comportamento del server quando c'è da gestire un qualsiasi tipo di richiesta.
Riduce il codice ridondante presente nel PC.



Template View

"*Renderizza le informazioni incorporando tag in una pagina html*"

Il TV è usato per organizzare la vista ed è una soluzione abbastanza semplice specialmente lavorando con persone che sono molto brave con la grafica e meno con il codice. E' possibile inserire un tag all'interno dell'html che quando invocato "esplode" invocando il codice a cui punta. "*E' il sogno dei grafici*" *[Ricca docet]*.

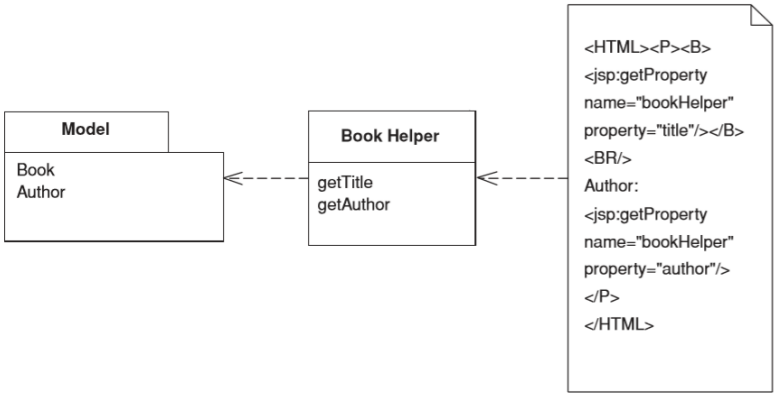
Come funziona il TV?

Incorporando dei tag in una pagina statica HTML che quando invocati richiamano il codice (ad esempio il risultato di una query).

Implementazioni popolari di TV sono presenti in JSP, PHP, etc. Uno svantaggio di questa soluzione è che si vanno a mischiare più livelli di un sistema, gli scriptlet presenti nelle pagine possono essere modificati solamente dai programmatori.

Quando usare TV?

Viene spesso usato per implementare la Vista nel MVC. Un'altra valida alternativa è il *Transform View*.



Transform View

"*Processa i dati del dominio elemento per elemento e li trasforma in HTML*"

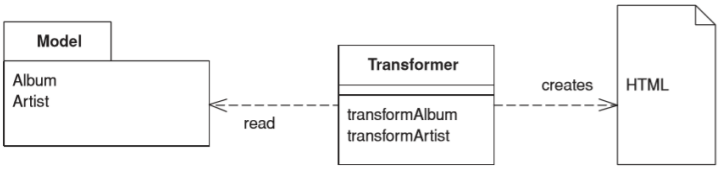
Nelle applicazioni moderne è diventato uno standard. E' pensato come un oggetto che riceve un model in input e ritorna una pagina html in output.

Come funziona TRV?

Gli oggetti del dominio vengono tradotti in XML e trasformati in html grazie a XSLT. La differenza con TV è nel modo in cui viene organizzata la Vista. TV organizza la vista in base all'output ricevuto dallo scriptlet, il TRV la organizza intorno a trasformazioni separate per ogni tipo di elemento in input.

Quando usare TRV?

Per implementare la Vista nel MVC. Molto semplice da testare e da manutenere.



Two Step View

"*Converte un oggetto del dominio in html in due passi: prima generando un qualche tipo di pagina logica, poi renderizzandola in html*".

L'idea è quella di generare le risposte in un format intermedio generando una sorta di pagina logica da sfruttare successivamente per renderizzare la pagina finale. Questo approccio prova ad affrontare il "problema"

dell'implementazione della Vista del MVC suddividendola in due step. E' usato per incrementare la flessibilità in quanto è più facile decidere ciò che deve essere visualizzato e cosa no grazie al livello logico intermedio che è possibile personalizzare in un modo più semplice.
Come funziona TSV?
Una possibile implementazione potrebbe partire da un oggetto del model codificato in XML (*domain-oriented*) sul quale si applica una trasformazione con XSLT per generare un livello (sempre un file XML) in cui sono presenti le informazioni che si vogliono renderizzare (*presentation-oriented*) sul quale si applica una seconda trasformazione XSLT per generare la pagina HTML.
Quando usare TSV?
Quando si vuole facilitare un cambiamento globale nella grafica di una o più pagine, distinguendo ciò che si renderizza da come lo si fa.
Un grosso punto a sfavore di questa soluzione è la mancanza di un tool/framework dedicato ai no-programmer per aiutarli ad usare TSV.
In questo modo è sempre necessario qualcuno che sappia programmare anche per fare un piccolo cambiamento nella grafica.

Application Controller
"E' un punto centrale per la gestione della navigazione tra le schermate e del flusso dell'applicazione"
La gestione della navigazione è gestita da un unico punto d'accesso centralizzato.
Come funziona APC?
L'APC ha due principali responsabilità:

- decidere quale parte della logica di dominio avviare;
- decidere la vista in cui visualizzare le risposte;

Per compiere questo lavoro viene dotato di due collezioni di riferimenti a classi:

- una di comandi del dominio da eseguire (possono essere oggetti Command o Transaction Script)
- una di viste (stringhe che corrispondono a nomi di pagine sul server, nomi di file XSLT o classi)

Viene implementato come un livello intermedio tra la Vista e il Model.
Quando usare APC?
Evitare di usarlo con applicazioni con poche pagine e dal flusso semplice.
E' molto consigliato quando c'è un determinato ordine in cui le pagine devono essere visitare. Anche quando la navigazione dipende dallo stato dei model.