Dipartimento di Informatica - Università di Salerno
Corso di Ingegneria del Software - Prof.re Andrea de Lucia

ODD
Object Design Document

Versione	1.0
Data	09/01/2019
Destinatario	Prof. Andrea De Lucia
Presentato da	Giovanni Vassalluzzo Luca Giaffreda Alfonso Luciani Roberto De Luca

Revision History

Data	Versione	Descrizione	Autori
09/01/2019	1.0	Prima stesura	Intero Team
16/01/2019	1.1	Revisione trade-offs	Intero Team
24/01/2019	1.2	Revisione design pattern	Intero Team
01/02/2019	2.0	Modifica packaging e revisione generale	

Sommario

1. Introduzione

Il presente documento illustra l'Object Design per l'E-Commerce DressShop. Sono quindi presentate tutte le scelte che precedono la fase di implementazione, come i trade-off su cui ci si è dovuto scontrare e le linee guida a cui i membri del team dovranno attenersi nella fase implementazione.
Sarà presentata inoltre un'overview dei package e classi del sistema, e sarà infine documentato il riuso di soluzioni effettuato, come l'utilizzo di design pattern o COTS.

1.1 Object Design Trade-off

Nella fase dell'Object Design sorgono diversi compromessi di progettazione ed è necessario stabilire quali punti rispettare e quali rendere opzionali.

Per quanto riguarda la realizzazione dell'E-Commerce DressShop, sono stati individuati i seguenti trade-off:

- **Comprensibilità vs Costi di sviluppo:**

Una parte fondamentale della progettazione e realizzazione del sistema è costituita da un codice comprensibile: sia le classi che i metodi dovranno essere chiari, semplici e ben documentati. Il codice dovrà essere ben indentato e strutturato per agevolare future modifiche o estensioni del sistema.
Per questo motivo ad ogni team member sarà richiesto massimo rispetto delle linee guida discusse al prossimo paragrafo.
La richiesta di comprensibilità, ed il relativo controllo, comporta tuttavia un incremento del tempo per lo sviluppo e la realizzazione dell'intero sistema.

- **Interfaccia vs Easy-use:**

L'interfaccia dovrà presentarsi semplice ed intuitiva per garantire il successo del sistema. Infatti, l'effettivo utilizzo del sistema sarà determinato dall'utilità e le semplificazioni che saranno concretamente apportate agli utenti
Potrà essere richiesta una penalizzazione in termini di interfaccia per permettere il raggiungimento delle varie funzionalità in pochi passi.

- **Efficienza vs Memoria:**

Dal momento che il sistema coinvolge una grande quantità di dati, si è scelto di preferire la memoria a discapito della efficienza, eliminando ridondanze dalla base di dati.

- **Sicurezza vs Prestazioni:**

Nel nostro sistema ogni richiesta del client viene validata attraverso l'uso di sessioni ed un controllo del livello di utenza. Inoltre, si intende utilizzare una libreria per la cifratura della password, in modo tale da poterla inserire nell'URL inviato per e-mail, permettendo quindi l'inserimento dell'utente nel database solo dopo che siano stati validati i dati
Questo trade-off resta comunque piuttosto bilanciato: le caratteristiche descritte potrebbero far aumentare il tempo di risposta del sistema, tuttavia tale aumento è completamente trascurabile.

1.2 Linee guida per l'implementazione

Nell'implementazione del sistema, i programmatori dovranno attenersi alle linee guida di seguito definite.

1.2.1 Classi e interfacce Java

1. Si applicano le seguenti restrizioni e variazioni:

All'inizio di ogni file devono essere presenti alcune informazioni strutturate come segue:

```
/*
NomeClasse
Breve descrizione della classe
*/
```

- Tutte le variabili dovrebbero essere private
- Tutte le variabili che non vengono mai modificate dovrebbero essere dichiarate come costanti (final).
- Si devono usare le tabulazioni per gestire l'indentazione.

1.2.2. Basi di dati

Le tabelle della base di dati dovrebbero rispettare la terza forma normale di Codd (3NF). Ove ciò non si verifichi, tale fatto deve essere riportato e motivato nella documentazione della base di dati. Le scelte di gestione nel trattamento dell'integrità referenziale devono essere riportate e motivate nella documentazione della base di dati.

1.2.2 Java Server Page (JSP)

Le pagine JSP devono, quando eseguite, produrre, in ogni circostanza, un documento conforme allo standard HTML versione 5.
Le parti Java delle pagine devono aderire alle convenzioni per la codifica in Java, con le seguenti puntualizzazioni:

1. Il tag di apertura (<%) è seguito immediatamente dalla fine della riga;
2. Il tag di chiusura (%>) si trova all'inizio di una riga;
3. È possibile emendare alle due regole precedenti, se il corpo del codice Java consiste in una singola istruzione:

```
<!•• Accettabile ••>
<% for (String par : paragraphs) {%>
<p class='item'><% out.print(par); %></p>
<% } %>
```

```
<!•• Non accettabile ••>
<p class='item'><% List<String> paragraphs = getParagraphs();
out.print(paragraphs.get(i++));%></p>
```

1.2.3 Pagine HTML

Le pagine HTML, statiche e dinamiche, devono essere totalmente aderenti allo standard HTML versione 5.
Inoltre, il codice HTML statico deve utilizzare l'indentazione, per facilitare la lettura, secondo le seguenti regole:

1. Un'indentazione consiste in una tabulazione;
2. Ogni tag deve avere un'indentazione maggiore del tag che lo contiene
3. Ogni tag di chiusura deve avere lo stesso livello di indentazione del corrispondente tag di apertura;

```
<!•• Accettabile ••>
<div><span><nl>
<li>Uno</li>
<li>
Due
</li>
</nl></span></div>
```

```
<!•• Non accettabile ••>
<div><span>
<nl>
<li>Uno</li>
<li>
Due
</li>
</nl></span>
</div>
```

4. I tag di commento devono seguire le stesse regole che si applicano ai tag normali;

1.2.4 Script Javascript

Gli script che svolgono funzioni distinte dal mero rendering della pagina dovrebbero essere collocati in file dedicati.

Il codice Javascript deve seguire le stesse convenzioni per il layout, i nomi ed i commenti del codice Java.

Gli oggetti Javascript devono essere preceduti da un commento in stile Javadoc, che segue il seguente formato:

```
/**
* Descrizione breve
* Eventuale ulteriore descrizione
* Specifica degli argomenti del costruttore (@param)
*
* Metodo nomeMetodo1
* Descrizione breve
* Eventuale ulteriore descrizione
* Specifica degli argomenti (@param)
* Specifica dei risultati (@return)
*
* Metodo nomeMetodo2
* Descrizione breve
* Eventuale ulteriore descrizione
* Specifica degli argomenti (@param)
* Specifica dei risultati (@return)
*
* ...
*/
function ClasseX(a, b, c) {
```

1.2.5 Fogli di stile CSS

Tutti gli stili non in-line devono essere collocati in fogli di stile separati.
Ogni foglio di stile deve essere iniziato da un commento analogo a quello presente nei file Java.
Ogni regola CSS deve essere formattata come segue:

1. I selettori della regola si trovano a livello 0 di indentazione, uno per riga;
2. L'ultimo selettore della regola è seguito da parentesi graffa aperta ({);
3. Le proprietà che costituiscono la regola sono listate una per riga e sono indentate rispetto ai selettori;
4. La regola è terminata da una parentesi graffa chiusa (}), collocata da sola su una riga;

Le proprietà e le regole poco chiare dovrebbero essere precedute da un commento esplicativo.
Le regole possono essere divise in blocchi concettualmente legati, preceduti da commenti in stile Javadoc che ne spiegano lo scopo, e seguiti da 2 righe bianche.

1.2.6 SQL

Le tabelle del database dovrebbero essere in terza forma normale di Codd (3NF). Qualora non lo fossero, la cosa deve essere documentata e motivata nello script di creazione del database.
I nomi delle tabelle devono:

- essere costituiti da sole lettere maiuscole;
- essere sostantivi tratti dal dominio del problema ed esplicativo del contenuto.

I nomi dei campi devono seguire le seguenti regole:

- devono essere costituiti da sole lettere minuscole;
- se il nome è costituito da più parole, queste sono separate da un underscore (_);
- il nome deve essere un sostantivo singolare tratto dal dominio del problema ed esplicativo del contenuto.

1.3 Acronimi

CCJPL: Code Conventions For the Java Programming

RAD: Requirements Analysis Document

SDD: System Design Document

ODD: Object Design Document

SQL: Structured Query Language

JSP: Java Server Pages

HTML: HyperText Markup Language

CSS: Cascading Style Sheets

3NF: Terza forma normale

COTS: Component Off-The-Shelf

API: Application Programming Interface

1.4 Riferimenti

Il presente ODD fa riferimento alle ultime versioni dei precedenti documenti rilasciati, in particolare:

- DressShop_RAD_v_1.3
- DressShop_SDD_v_1.0

Si fa riferimento, inoltre, al materiale fornito dal prof. Andrea de Lucia ed al seguente libro di testo:

- B.Bruegge, A.H. Dutoit, Object Oriented Software Engineering – Using UML, Patterns and Java, Prentice Hall.

1.5 Organizzazione del contenuto

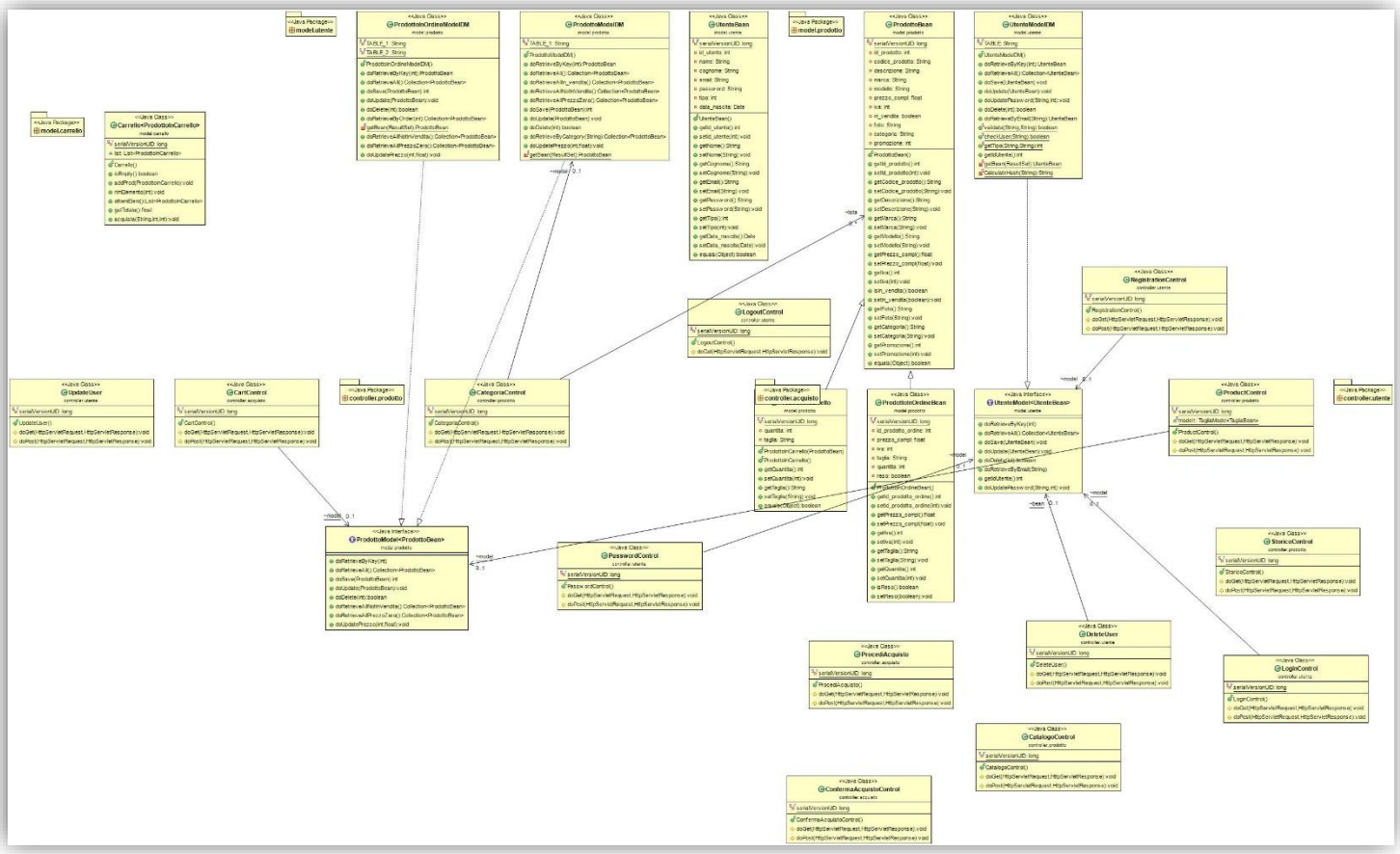
In questo primo capitolo si è fatta un'introduzione al documento, trattando i vari trade-off e le linee guida per l'implementazione. Il capitolo 2 mostrerà la suddivisione in classi e package, e nel capitolo 3 saranno presentate le tecniche utilizzate per garantire il riuso, come i COTS ed i Design Pattern.
Il documento si conclude con un glossario.

2. Packaging e class interfaces

Di seguito è presentato un diagramma di packaging e class interfaces, realizzato utilizzando l'applicativo ObjectAid per Eclipse.

Per semplicità di esposizione, alcune dipendenze sono state rimosse. Inoltre, ove è presente una dipendenza per tutte le classi di un package, esse sono rappresentate con una sola linea che collega il package stesso.

Per ulteriori dettagli sulle descrizioni e sulle dipendenze complete si faccia riferimento alla Javadoc, consegnata insieme a questo documento.



Per consultare in dettaglio il diagramma si rimanda al file specifico allegato alla documentazione.

3. Documentazione del riuso

3.1 Design Pattern

I design pattern a cui si è fatto affidamento sono i seguenti:

- **Model View Controller**

Il design pattern principale utilizzato nella realizzazione del sistema è il Model View Controller. Il comportamento dell'applicazione è catturato dai Model che, restando indipendente dall'interfaccia utente, gestisce direttamente i dati, mentre Le View sono utilizzate per la presentazione delle informazioni all'utente. Il Controller gestisce l'input e modifica di conseguenza lo stato delle altre due componenti.

- **Singleton**

Uno degli aspetti critici nel funzionamento del sistema è l'accesso ai dati persistenti.
Per questo motivo, si è scelto di delegare ad una singola classe la responsabilità di gestire le connessioni al database, applicando ad essa il design pattern Singleton, che permette di gestire le istanze delle connessioni dei vari utenti.

- **Data Access Object**

L'implementazione del sistema prevede l'utilizzo di un database MySQL, attraverso l'utilizzo di API JDBC, per garantire la persistenza dei dati. Per rendere il sistema flessibile ad eventuali futuri cambi di tecnologie è previsto l'utilizzo del design pattern Data Access Object.
Il sistema prevede l'utilizzo di Manager per ogni entità, accessibili dai controller attraverso delle interfacce. In questo modo, un eventuale cambio di tecnologia non modificherebbe il cambiamento dei controller, che continueranno ad utilizzare le interfacce.

- **Abstract Factory**

Nonostante l'utilizzo del Data Access Object, il disaccoppiamento non si può dire completo. Infatti, nonostante i manager accedano alle interfacce, la creazione delle istanze concrete è ancora in mano alle classi controller. La soluzione a cui si è giunti è quindi l'utilizzo del pattern Abstract Factory, che si occuperà di creare e distribuire istanze delle classi manager.

- **Front Controller**

Il design pattern Front Controller è utile a fornire un punto di ingresso centralizzato per la gestione delle richieste. Questo gestore può eseguire l'autenticazione, controllare permessi o in generale tracciare la richiesta.

3.2 Component Off-The-Shelf (COTS)

L'E-Commerce DressShop farà uso di componenti off-the-shelf, ossia componenti software disponibili sul mercato per facilitare la creazione del progetto, allo scopo di contenere i costi di sviluppo e manutenzione. Verranno utilizzate applicazioni complete che offrono un API per permettere ad altri componenti software di accedere alle proprie funzionalità.

Il sistema si servirà di un DBMS per gestire i dati persistenti: in particolare, sarà utilizzato MySQL Server.
Andremo ad utilizzare, inoltre, Bootstrap, framework open source che contiene una raccolta di strumenti per la creazione di siti e applicazioni per il Web. Esso offre modelli di progettazione basati su HTML e CSS, sia per la tipografia sia per le varie componenti dell'interfaccia, quali moduli, pulsanti e strumenti per la navigazione, piuttosto che alcune estensioni opzionali di JavaScript.
La scelta di usufruire di tale framework deriva prevalentemente dalla necessità di garantire il responsive web design, per la visualizzazione della piattaforma su una più vasta tipologia di dispositivi.

Infine, per la gestione della messaggistica tra l'amministratore del sistema e gli utenti, sarà usato un provider esterno, predisposto all'invio di e-mail. Tale componente si attiverà in più occasioni durante l'utilizzo dell'E-Commerce DressShop:

- per facilitare il recupero della password da parte dell'utente;
- per verificare l'avvenuta registrazione con l'invio di un codice di conferma;

4. Glossario

DBMS: Database Management System, software progettato per la creazione e la manipolazione di database.

3NF: Una base dati è in 3NF (*terza forma normale*) se è in 2NF e tutti gli attributi non-chiave dipendono dalla chiave soltanto, ossia non esistono attributi che dipendono da altri attributi non-chiave. Tale normalizzazione elimina la dipendenza transitiva degli attributi dalla chiave.

Bootstrap: Bootstrap è una raccolta di strumenti liberi per la creazione di siti ed applicazioni per il Web. Essa contiene modelli di progettazione basati su HTML e CSS, sia per la tipografia, che per le varie componenti dell'interfaccia, come moduli, pulsanti e navigazione, così come alcune estensioni opzionali di JavaScript.

COTS: L'espressione componente COTS o componente OTS, in inglese (Commercial) Off-the-Shelf component, si riferisce a componenti hardware e software disponibili sul mercato per l'acquisto da parte di aziende di sviluppo interessate a utilizzarli nei loro progetti.

Design Pattern: Un design pattern è un concetto che può essere definito "una soluzione progettuale generale ad un problema ricorrente". Si tratta di una descrizione o modello logico da applicare per la risoluzione di un problema che può presentarsi in diverse situazioni durante le fasi di progettazione e sviluppo del software.

HTML: L'HyperText Markup Language è il linguaggio di markup solitamente usato per la formattazione e impaginazione di documenti ipertestuali disponibili nel World Wide Web sotto forma di pagine web.

Javadoc: E' un applicativo incluso nel Java Development Kit della Sun Microsystems, utilizzato per la generazione automatica della documentazione del codice sorgente scritto in linguaggio Java.

Trade-off: In economia un trade-off (o trade off) è una situazione che implica una scelta tra due o più possibilità, in cui la perdita di valore di una costituisce un aumento di valore in un'altra.