Object Design Document
ODD_TechShop

Sommario

1. Introduction

1.1. Object design trade-offs

• **Functionality vs. Usability**

Il nostro software incorpora funzionalità che soddisfano le specifiche dei requisiti. Con MySQL come componente del database, è un DBMS. Invece per la logica di controllo e business utilizziamo le servlet basato sul linguaggio Java e JSP, Css e JavaScript come front-end. L'usabilità del sistema è pienamente raggiunta.

• **Portability vs. Readability**

Il nostro sistema sacrifica la leggibilità del codice, per far fronte ad un requisito non funzionale la portabilità del sistema attraverso vari dispositivi, che siano schermi dei pc, tablet, o smartphone. Usando opportuno linguaggio di formattazione delle pagine Web "Css", tramite le Media Queries permette di strutturare il contenuto di una pagina Html e JSP adattandola alla dimensione dello schermo. Inoltre usando Java Script siamo riusciti a realizzare un sito multilingua.

• **Rapid Development vs. Robustness**

Il Sistema sacrifica lo sviluppo di software robusto, per garantire un sistema funzionante nel più breve tempo richiesto, per poi garantire in futuro aggiornamenti per rendere il software robusto e per implementare altre funzionalità, o rendere più dettagliate quelle preesistenti.

• **Delivery Time vs Functionality**

Si preferisce dare priorità al delivery time rispetto al numero di funzionalità offerte dal sistema. Il sistema TechShop deve mettere a disposizione dell'utente le funzionalità definite. Tuttavia, il sistema deve essere consegnato entro la data prefissata. Questo porta inevitabilmente a un trade off.

Per tanto si preferisce il delivery time a discapito delle funzionalità, utilizzando un approccio in cui vengano realizzate prima le funzionalità ad alta priorità, lasciando le funzionalità con priorità media o bassa a future evoluzioni del sistema nel caso in cui il tempo a disposizione non fosse sufficiente a fornire tutte le implementazioni.

• **Memory space vs. Response time:**

Poiché il nostro sistema TechShop e un e-commerce dovrà gestire una grande mole di dati per cui abbiamo bisogno di un DBMS che memorizzi tutti i dati, dal carrello dell'utente, ai prodotti e categorie dei relativi negozi. Quindi questo implica scritture e

letture in un DB però avendo un tempo di risposta contenuto nei limiti dettati dai requisiti non funzionali nel RAD.

1.2. Interface documentation guidelines

• Le classi sono nominate con nomi singolari, relativamente brevi e significativi. Non usare verbi, aggiungere un suffisso relativo allo strato di appartenenza, usare suffisso Exception per le classi di eccezioni.

Per lo strato di integrazione con il database usare nome classe più DAO. Per lo strato control e view la scelta ricade su nomi significativi per lo scopo che sono state ralizzate.

• I metodi sono denominati con frasi verbali, campi e parametri con frasi di nomi, brevi ed intuitivi. Non ripetere nomi della classe nel nome del metodo.

•usare una convenzione chiara nella selezione dei nomi dei metodi (add, addAll, get, set, size, clear, remove)

Nome metodo	Utilizzo
add	Inserimento di un elemento in una lista.
addAll	Inserimento di una collezione in una lista.
clear	Rimuove tutti gli elementi di una lista.
get	Reperimento di un elemento specifico di una lista.
remove	Rimozione di un elemento.
size	Restituzione del numero degli elementi di una lista.

• selezionare nomi significativi per attributi e variabili e parametri, brevi.

• considerare l'utilizzo delle lettere i, j, k per le variabili di ciclo

• Lo stato dell'errore viene restituito tramite un'eccezione, non un valore di ritorno.

• utilizzare il livello di accesso più ristretto possibile

Livello di accesso	Stessa classe	Stesso package	Classe ereditante	Tutte le altre
Privato private	✓	×	×	×
Amichevole non specificato	✓	✓	×	×
Protetto protected	✓	✓	✓	×
Pubblico public	✓	✓	✓	✓

• massimizzare la coesione tra le classi e minimizzare accoppiamento.

• eseguire controllo del valore dei parametri passati ad un metodo appena possibile.

• L'indentazione deve essere effettuata con un TAB e qualunque sia il linguaggio usato per la produzione di codice, ogni istruzione deve essere opportunamente indentata.

Es.
```
for(int i=0; i<x; i++){
if(x>y){
x = 5;
return;
}
}
```
Deve essere sostituita da:
```
for(int i=0; i<x;i++){
        if(x>y){
                x = 5;
                return;
        }
}
```

• Mettere le dichiarazioni all'inizio dei blocchi. Non aspettare di dichiarare la variabile al loro primo uso: può confondere il programmatore inesperto e impedire la portabilità del codice dentro lo scope.
Le variabili di istanza verranno dichiarate nelle ultime righe di codice del file. L'unica eccezione a questa regola sono gli indici dei cicli for che in Java possono essere dichiarati nell' istruzione stessa.

• Inizializzare le variabili locali nel punto in cui sono state dichiarate a meno che il suo valore iniziale non dipenda da un calcolo che occorre eseguire prima.

1.3. Definizioni, acronimi e abbreviazioni

DBMS: Data Base Management System.
Off-The-Shelf: Servizi esterni di cui viene fatto utilizzo da terzi.
HTML: Linguaggio di mark-up per pagine web.
CSS: Linguaggio usato per definire la formattazione di pagine web.
JavaScript: Linguaggio di scripting orientato agli oggetti e agli eventi, comunemente utilizzato nella programmazione Web lato client per la creazione, in siti web e applicazioni web, di effetti dinamici interattivitramite funzioni di script invocate da eventi innescati a loro volta in vari modi dall'utente sulla pagina web in uso

1.4. References

- Problem Statement
- Requirements Analysis Document
- System Design Document
- http://java.sun.com

Organizzazione dei file

Ogni file deve appartenere ad uno dei seguenti package, in base al suo ruolo nella soluzione del problema:

-**Model:** contiene tutti qui file che si occupano della gestione della persistenza dei dati;

-**Eccezione:** contiene i file per nuovi tipi di eccezione creati per l'applicazione;

-managerAccaouting: contiene le classi per la gestione dei vari utenti del sistema, con le relative funzionalità di registrazione e login;

-**managerNegozio:** contiene le classi per la gestione del negozio, categorie e prodotti, con funzionalità di creazione, modifica e cancellazione;

-**managerOrdine:** contiene le classi per la gestione degli ordine;

-**Testing:** contiene testing blackbox e whiteblox delle varie componenti;

-**SystemTesting:** contiene i casi di test relativi al testing di sistema.

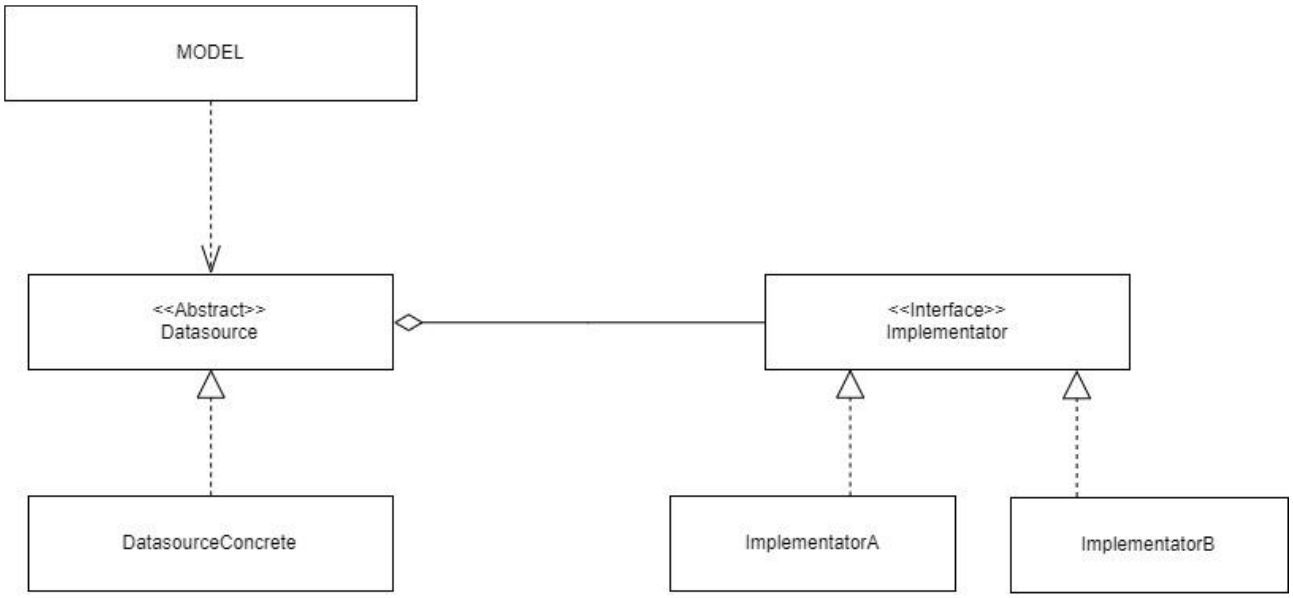
- Organizzare in una cartella "images" le varie risorse necessarie per la memorizzazione delle immagini del logo del negozio, delle categorie e dei prodotti. Queste saranno salvate secondo una specifica struttura di directory. Con convenzione di creare una cartella con nome negozio ed inserire tutte le relative immagini, con la specifica di usare nome prodotto uguale al proprio id, invece per la categoria il nome della categoria, per facilitare il salvataggio e il caricamento delle immagini, senza appesantire il server.

-**Tutte le view che si interfacciano con l'utente sono divise tra diverse cartelle** per permettere agli sviluppatori di facilitare la individuazione. La scelta è ricaduta su nomi specifici collegati a ciò che si andava a realizzare: seller (per le jsp di creazione del venditore e del negozio), venditore (per le varie jsp di visualizzazione e modifica dei vari elementi).

-Sono stati creati 4 Css uno generale, e uno per ogni colore che il venditore può scegliere per il proprio e-commerce.

-Tutta la documentazione è organizzata in una cartella "doc".

1.5. Scelta del Design Pattern



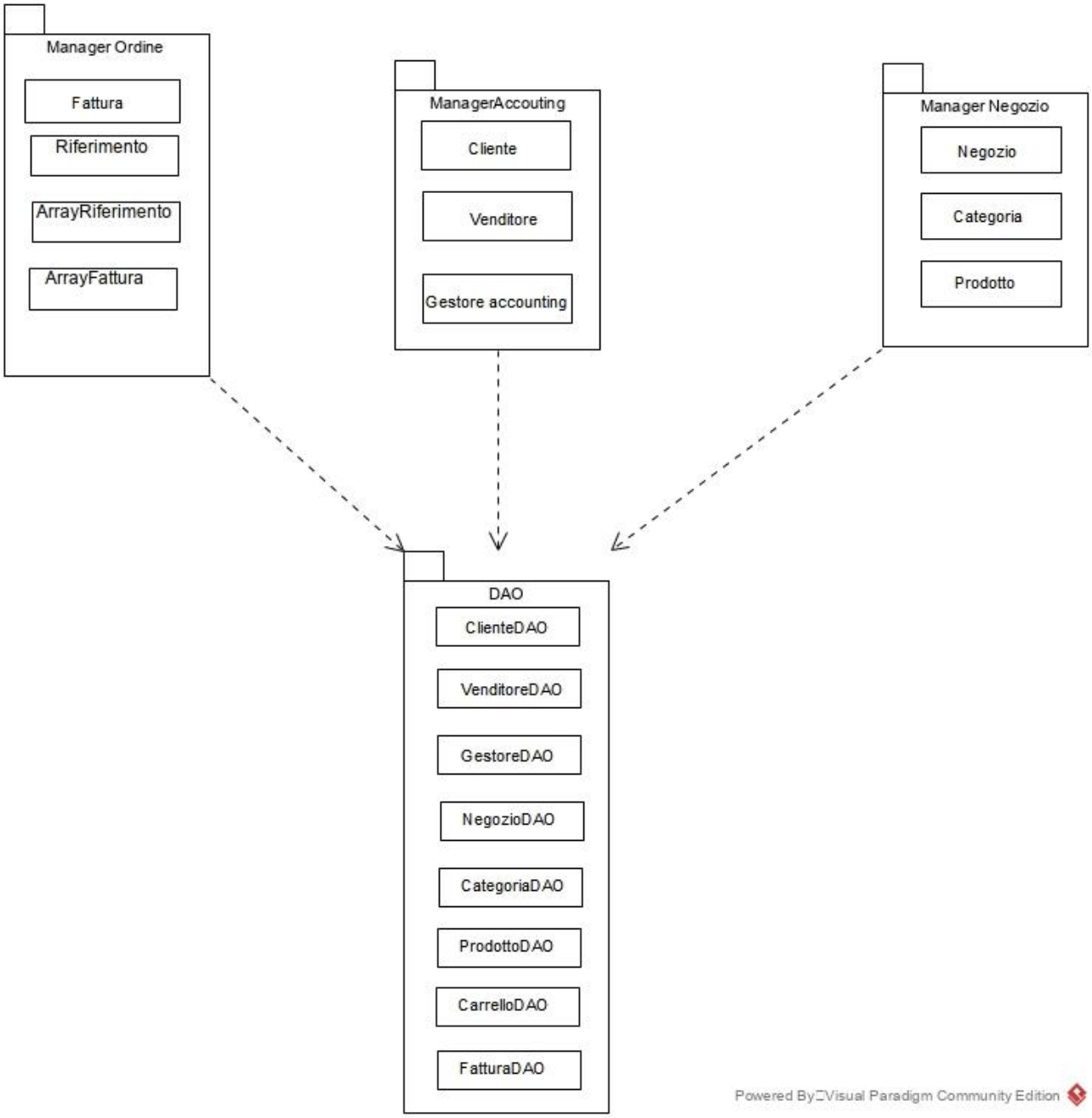
In successive versioni del sistema si potrebbe utilizzare il Bridge Pattern per la gestione della connessione del database. In questo modo si renderebbe indipendente l'implementazione per l'interfaccia al database a chi la utilizza.

Questo design pattern non è stato implementato, perché tra il trade off manutenibilità & tempo di consegna.

All'inizio del documento di System Design si era preso in considerazione di usare il design pattern Facade, che dava priorità alla manuntenibilità a discapito dell'efficienza. Facade permetteva di utilizzare ogni sottosistema in maniera semplice e indipendente dalla loro implementazione. Però durante la implementazione del sistema ci siamo accorti che portava all'aggiunta di altre classi, con un peggioramento della velocità del sistema.

Ciò ci ha portato ad utilizzare dipendenze dirette tra le varie classi permettendoci oltre ad una migliore facilità di implementazione, anche un'ottima scelta per non avere un alto accoppiamento tramite la stessa interfaccia tra le componenti del View (jsp e servlet) e del livello intermedio.

2. Packages


Manager Ordine
Fattura
Riferimento
ArrayRiferimento
ArrayFattura
ManagerAccouting
Cliente
Venditore
Gestore accounting
Manager Negozio
Negozio
Categoria
Prodotto
DAO
ClienteDAO
VenditoreDAO
GestoreDAO
NegozioDAO
CategoriaDAO
ProdottoDAO
CarrelloDAO
FatturaDAO
Powered By Visual Paradigm Community Edition

Packages	
Package	**Description**
control	
eccezione	
manageraccouting	
managernegozio	
managerordine	
model	
systemtesting	
testing	

- **Package control**

Class Summary	
Class	**Description**
Controlli	
InserisciCategoria	Servlet usata per reindirezzare utente all' aggiunta della categoria
InserisciProdotto	Servlet che reindirizza utente alla creazione di un nuovo prodotto
InsertCategoria	Servlet per inserire una nuova categoria
InsertProduct	Servlet che inserisce un nuovo prodotto
LoginCliente	Servlet effettua il login del cliente
LoginGestoreAccouting	Servlet effettua il login del gestore accouting
LoginVenditore	Servlet effettua il login al venditore
Logout	Servlet che fa il logout dell'utente

ModificaCategoria	Servlet che effetua la modifica della categoria
ModificaProdotto	Servlet effettua la modifica del prodotto
RegisterNegozio	Servlet Registrazione del negozio
RegisterSeller	Servlet registrazione del venditore
RemoveCat	Servlet cancella categoria
RemoveProd	Servlet rimuove prodotto
UpdateCategoria	Servlet update Categoria
UpdateProdotto	Servlet modifica prodotto
Upload	Servlet fa upload del logo del negozio
UploadCategoria	Servlet effettua upload del logo della categoria
UploadProdotto	Servlet effettua upload del logo del prodotto
ViewCategoria	Servlet ricarica la jsp delle categorie di un negozio
ViewProdotto	Servlet ricarica la jsp dei prodotti di una relativa categoria di un negozio
VisualizzaOrdineVenditore	Servlet che reindirizza il venditore alla visione degli ordini
VisualizzaProdotti	Servlet che fa carica i prodotti di una determinata categoria di un negozio

- **Package eccezione**

Exception Summary	
Exception	**Description**
CategoriaNonEsisteException	Eccezzione categoria non esiste
NegozioNonEsistenteException	Eccezione Negozio non esiste
ParametroNonCorrettoException	Eccezione parametro non corretto
UtenteNonTrovatoException	Eccezione Utente non trovato

- **Package manageraccouting**

Class Summary	
Class	**Description**
Utente	Classe Astratta Utente
Venditore	La classe rappresenta un Venditore, è una specializzazione di Utente Permette di trovare se un venditore è presente nel database,

- **Package managernegozio**

Class Summary	
Class	**Description**
Categoria	la classe Categoria gestisce le operazioni di modifica,cancellazione e rimozione di una categoria costruzine del path per inserimento delle imagini
Negozio	La classe negozio permette di aggiungere un negozio, creare la cartella del negozio sul server restituire oggetto negozio di un determinato venditore
Prodotto	la classe Prodotto gestisce le operazioni di modifica, cancellazione di un prodotto

- **Package managerordine**

Class Summary	
Class	**Description**
ArrayFattura	Permette di gestire una lista di fatture con operazioni aggiunta
ArrayRiferimento	Ha le funzionalità per gestire un array di oggetti riferimento
Fattura	
Riferimento	Contiene i dati permenenti di una fattura, ha i metodi getter e setter per modificare l'oggetto riferimento inoltre permette di gestire arrayriferimento.

- **Package model**

Class Summary	
Class	**Description**
CategoriaDAO	Posside le operazioni di gestione la table Categoria all'interno del database
DriverManagerConnectionPool	Connessione con il db tramite ConnectionPool
NegozioDAO	Realizza le operazioni da effettuare con li databese riferite alla table Negozio
ProdottoDAO	Permette di gestire la table Prodotto nel database con le operazioni piů comuni
RiferimentoDAO	Permette di operare sulla table riferimento per operazioni CRUD
VenditoreDAO	Permette di effettuare le operazioni CRUD per la table Venditore

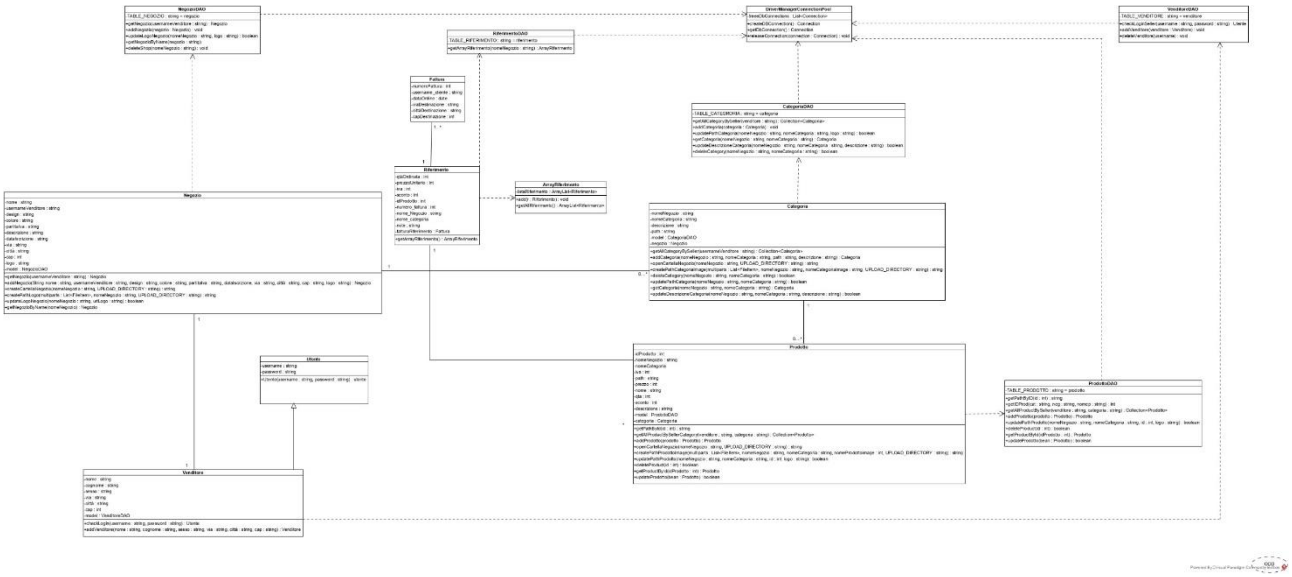
- **Package systemtesting**

Class Summary	
Class	**Description**
LoginVenditoreTest	Test di sistema Login Venditore
RegistrazioneVenditoreTest	Test di sistema Registrazione Venditore

- **Package Testing**

Class Summary	
Class	**Description**
CategoriaDAOTest	Test di unità CategoriaDAO
CategoriaTest	Test di unità Categoria
NegozioDAOTest	Test di unità NegozioDAO
NegozioTest	Test di unità Negozio
ProdottoDAOTest	Test di unità ProdottoDAO
ProdottoTest	Test di unità Prodotto
Tc1_0_Login_Venditore	Test di sistema
Tc2_0_Registrazione_Venditore	Test di sistema
Tc3_0_Registrazione_Negozio	Test di sistema
Tc4_0_Inserisci_Categoria	Test di sistema
Tc5_0_Inserisci_Prodotto	Test di sistema
Tc6_0_Modifica_Categoria	Test di sistema
Tc7_0 Cancellazione_Categoria	Test di sistema
Tc8_0_Modifica_Prodotto	Test di sistema
Tc9_0_Cancellazione_Prodotto	Test di sistema
VenditoreDAOTest	Test di unità VenditoreDAO
VenditoreTest	Test di unità Venditore

3. Class interfaces



- **Package manageraccouting**

Class Summary	
Class	**Description**
Utente	Classe Astratta Utente
Venditore	La classe rappresenta un Venditore, è una specializzazione di Utente Permette di trovare se un venditore è presente nel database,

- **Class Utente**

java.lang.Object

manageraccouting.Utente

All Implemented Interfaces:

java.io.Serializable

Direct Known Subclasses:

Venditore

---

public abstract class **Utente**

extends java.lang.Object

implements java.io.Serializable

Classe Astratta Utente

Author:

cetra

See Also:

Serializable, Serialized Form

Constructor Summary

Constructors	
Constructor and Description	
Utente()	Costruttore vuoto
Utente(java.lang.String username, java.lang.String password)	Costruttore con i parametri

Method Summary

All Methods	Instance Methods	Concrete Methods
Modifier and Type	**Method and Description**	
java.lang.String	**getPassword**()	
java.lang.String	**getUsername**()	
void	**setPassword**(java.lang.String password)	
void	**setUsername**(java.lang.String username)	

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Utente

public Utente()

Costruttore vuoto

Utente

public Utente(java.lang.String username,
java.lang.String password)

Costruttore con i parametri

Parameters:

username -

password -

Method Detail

getUsername

public java.lang.String getUsername()

Returns:

the username

setUsername

public void setUsername(java.lang.String username)

Parameters:

username - the username to set

getPassword

public java.lang.String getPassword()

Returns:

the password

setPassword

public void setPassword(java.lang.String password)

Parameters:

password - the password to set

- **Class Venditore**

java.lang.Object

manageraccouting.Utente

manageraccouting.Venditore

All Implemented Interfaces:

java.io.Serializable

---

public class **Venditore**

extends Utente

La classe rappresenta un Venditore, č una specializzazione di Utente Permette di trovare se un venditore č presente nel database,

Author:

cetra

See Also:

Utente, Serialized Form

Constructor Summary

Constructors
Constructor and Description
Venditore() Costruttore vuoto
Venditore(java.lang.String username, java.lang.String password, java.lang.String nome, java.lang.String cognome, java.lang.String email, java.lang.String sesso, java.lang.String telefono, java.lang.String via, java.lang.String citta, java.lang.String cap) Costruttore che crea un istanza di venditore

Method Summary

All Methods	Instance Methods	Concrete Methods
Modifier and Type	**Method and Description**	
Venditore	**addVenditore**(java.lang.String username, java.lang.String password, java.lang.String nome, java.lang.String cognome, java.lang.String email, java.lang.String sesso, java.lang.String telefono, java.lang.String via, java.lang.String citta, java.lang.String cap) Crea un nuovo venditore, i parametri sono controllati da javascript e espressioni regolari direttamente all'inserimento.	
Utente	**checkLogin**(java.lang.String username, java.lang.String password) Controlla che nel database sia presente un istanza di venditore con username e password identiche come quelle passate come parametro	
java.lang.String	**getCap**()	
java.lang.String	**getCitta**()	
java.lang.String	**getCognome**()	
java.lang.String	**getEmail**()	
java.lang.String	**getNome**()	
java.lang.String	**getSesso**()	
java.lang.String	**getTelefono**()	
java.lang.String	**getVia**()	
void	**setCap**(java.lang.String cap)	
void	**setCitta**(java.lang.String citta)	
void	**setCognome**(java.lang.String cognome)	
void	**setEmail**(java.lang.String email)	
void	**setNome**(java.lang.String nome)	
void	**setSesso**(java.lang.String sesso)	

void	**setTelefono**(java.lang.String telefono)
void	**setVia**(java.lang.String via)

Methods inherited from class manageraccouting.Utente

getPassword, getUsername, setPassword, setUsername

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Venditore

public Venditore()

Costruttore vuoto

Venditore

public Venditore(java.lang.String username,
java.lang.String password,
java.lang.String nome,
java.lang.String cognome,
java.lang.String email,
java.lang.String sesso,
java.lang.String telefono,
java.lang.String via,
java.lang.String citta,
java.lang.String cap)

Costruttore che crea un istanza di venditore

Parameters:

username -

password -

nome -

cognome -

email -

sesso -

telefono -

via -

citta -

cap -

Method Detail

checkLogin

public Utente checkLogin(java.lang.String username,

java.lang.String password)

throws java.sql.SQLException,

UtenteNonTrovatoException,

ParametroNonCorrettoException

Controlla che nel database sia presente un istanza di venditore con username e password identiche come quelle passate come parametro

Parameters:

username, - username del venditore da cercare

password, - password del venditore da cercare

Returns:

ritorna l'istanza del utente cercato

Throws:

java.sql.SQLException

UtenteNonTrovatoException

ParametroNonCorrettoException

addVenditore

public Venditore addVenditore(java.lang.String username,

java.lang.String password,

java.lang.String nome,

java.lang.String cognome,

java.lang.String email,

java.lang.String sesso,

java.lang.String telefono,

java.lang.String via,

java.lang.String citta,

java.lang.String cap)

throws java.sql.SQLException

Crea un nuovo venditore, i parametri sono controllati da javascript e espressioni regolari direttamente all'inserimento.

Precondizione

username formato: lettere e numeri, min 3, max 16

password formato: almeno lettera minuscola e maiuscola, un numero, carattere speciale min 6, max 25

nome formato: solo lettere minimo 1 max 25

cognome formato: solo lettere minimo 1 max 25

email formato: una sola @, un punto con max 3 lettere alla fine.

sesso formato: uno dei tre specificati M,F,Other

telefono formato: 10 cifre.Senza spazi

via formato: lettere e cifre

citta formato: lettere e cifre

cap formato: 5 cifre

Parameters:

username, - del venditore da inserire

password, - del venditore da inserire

nome - , del venditore da inserire

cognome, - del venditore da inserire

email, - del venditore da inserire

sesso, - del venditore da inserire

telefono, - del venditore da inserire

via, - del venditore da inserire

citta, - del venditore da inserire

cap, - del venditore da inserire

Returns:

ritorna oggetto Venditore appena aggiunto.

Throws:

java.sql.SQLException

getNome

public java.lang.String getNome()

Returns:

the nome

setNome

public void setNome(java.lang.String nome)

Parameters:

nome - the nome to set

getCognome

public java.lang.String getCognome()

Returns:

the cognome

setCognome

public void setCognome(java.lang.String cognome)

Parameters:

cognome - the cognome to set

getEmail

public java.lang.String getEmail()

Returns:

the email

setEmail

public void setEmail(java.lang.String email)

Parameters:

email - the email to set

getSesso

public java.lang.String getSesso()

Returns:

the sesso

setSesso

public void setSesso(java.lang.String sesso)

Parameters:

sesso - the sesso to set

getTelefono

public java.lang.String getTelefono()

Returns:

the telefono

setTelefono

public void setTelefono(java.lang.String telefono)

Parameters:

telefono - the telefono to set

getVia

public java.lang.String getVia()

Returns:

the via

setVia

public void setVia(java.lang.String via)

Parameters:

via - the via to set

getCitta

public java.lang.String getCitta()

Returns:

the citta

setCitta

public void setCitta(java.lang.String citta)

Parameters:

citta - the citta to set

getCap

public java.lang.String getCap()

Returns:

the cap

setCap

public void setCap(java.lang.String cap)

Parameters:

cap - the cap to set

Package managernegozio

Class Summary	
Class	**Description**
Categoria	la classe Categoria gestisce le operazioni di modifica,cancellazione e rimozione di una categoria costruzine del path per inserimento delle imagini
Negozio	La classe negozio permette di aggiungere un negozio, creare la cartella del negozio sul server restituire oggetto negozio di un determinato venditore
Prodotto	la classe Prodotto gestisce le operazioni di modifica, cancellazione di un prodotto

Class Negozio

java.lang.Object

managernegozio.Negozio

All Implemented Interfaces:

java.io.Serializable

---

public class **Negozio**

extends java.lang.Object

implements java.io.Serializable

La classe negozio permette di aggiungere un negozio, creare la cartella del negozio sul server restituire oggetto negozio di un determinato venditore

Author:

cetra

See Also:

Serialized Form

Constructor Summary

Constructors	
Constructor and Description	
Negozio()	
costruttore vuoto	
Negozio(java.lang.String nomeNegozio, java.lang.String usernameVenditore, java.lang.String template, java.lang.String colore, java.lang.String partitaIva, java.lang.String dataIscrizione, java.lang.String descrizione, java.lang.String via, java.lang.String citta, java.lang.String cap, java.lang.String Logo)	
Costruttore che crea un istanza di negozio	

Method Summary

All MethodsInstance MethodsConcrete Methods	
Modifier and Type	**Method and Description**
Negozio	**addNegozio**(java.lang.String nomeNegozio, java.lang.String usernameVenditore, java.lang.Stri java.lang.String colore, java.lang.String partitaIva, java.lang.String dataIscrizione, java.lang.Str java.lang.String via, java.lang.String citta, java.lang.String cap, java.lang.String Logo) Aggiunge un nuovo negozio nel Database I parametri di input vengono controllati sia lato cliente server nelle servlet.
java.lang.String	**createCartellaNegozio**(java.lang.String nomeNegozio, java.lang.String UPLOAD_DIRECTORY) Crea la cartella negozio se non esiste altrimenti la apre e ritorna il path
java.lang.String	**createPathLogo**(java.util.List<org.apache.tomcat.util.http.fileupload.FileItem> multiparts, java.lang.String nomeNegozio, java.lang.String UPLOAD_DIRECTORY) Crea il la stringa del path per il logo del negozio
java.lang.String	**getCap**()
java.lang.String	**getCitta**()
java.lang.String	**getColore**()
java.lang.String	**getDataIscrizione**()
java.lang.String	**getDescrizione**()
java.lang.String	**getDesign**()
java.lang.String	**getLogo**() \

Negozio	**getNegozio**(java.lang.String usernameVenditore) Restituisce il negozio associato ad un venditore Il parametro usernameVenditore è controllato ne
Negozio	**getNegozioByName**(java.lang.String nomeNegozio) Restituire il negozio passato come parametro il nome del negozio
java.lang.String	**getNomeNegozio**()
java.lang.String	**getPartitaIva**()
java.lang.String	**getUsernameVenditore**()
java.lang.String	**getVia**()
void	**setCap**(java.lang.String cap)
void	**setCitta**(java.lang.String citta)
void	**setColore**(java.lang.String colore)
void	**setDataIscrizione**(java.lang.String dataIscrizione)
void	**setDescrizione**(java.lang.String descrizione)
void	**setDesign**(java.lang.String design)
void	**setLogo**(java.lang.String logo)
void	**setNomeNegozio**(java.lang.String nomeNegozio)
void	**setPartitaIva**(java.lang.String partitaIva)
void	**setUsernameVenditore**(java.lang.String usernameVenditore)
void	**setVia**(java.lang.String via)
boolean	**updateLogoNegozio**(java.lang.String nomeNegozio, java.lang.String urlLogo) Modifica url del path del logo negozio

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Negozio

public Negozio()

costruttore vuoto

Negozio

public Negozio(java.lang.String nomeNegozio,
java.lang.String usernameVenditore,
java.lang.String template,
java.lang.String colore,
java.lang.String partitaIva,
java.lang.String dataIscrizione,
java.lang.String descrizione,
java.lang.String via,
java.lang.String citta,
java.lang.String cap,
java.lang.String Logo)

Costruttore che crea un istanza di negozio

Parameters:

nomeNegozio -

usernameVenditore -

template -

colore -

partitaIva -

dataIscrizione -

descrizione -

via -

citta -

cap -

Logo -

Method Detail

getNegozio

public Negozio getNegozio(java.lang.String usernameVenditore)
throws java.sql.SQLException,
NegozioNonEsistenteException

Restituisce il negozio associato ad un venditore

Il parametro usernameVenditore è controllato nella servlet

Parameters:

usernameVenditore, - venditore su cui fare la ricerca

Returns:

Negozio, oggetto negozio del determinato venditore, altrimenti lancia un eccezione

Throws:

java.sql.SQLException

NegozioNonEsistenteException

addNegozio

public Negozio addNegozio(java.lang.String nomeNegozio,
java.lang.String usernameVenditore,
java.lang.String template,
java.lang.String colore,
java.lang.String partitaIva,
java.lang.String dataIscrizione,
java.lang.String descrizione,
java.lang.String via,
java.lang.String citta,
java.lang.String cap,
java.lang.String Logo)
throws java.sql.SQLException

Aggiunge un nuovo negozio nel Database

I parametri di input vengono controllati sia lato cliente nelle jsp che lato server nelle servlet.

per riferimento abbiamo un tabella dei formati nel RAD

Parameters:

nomeNegozio -

usernameVenditore -

template -

colore -

partitaIva -

dataIscrizione -

descrizione -

via -

citta -

cap -

Logo -

Returns:

Negozio, ritorna oggetto negozio appenna aggiunto, altrimenti si genera un eccezione in caso di errore.

Throws:

java.sql.SQLException

createCartellaNegozio

public java.lang.String createCartellaNegozio(java.lang.String nomeNegozio,

java.lang.String UPLOAD_DIRECTORY)

Crea la cartella negozio se non esiste altrimenti la apre e ritorna il path

Parameters:

nomeNegozio -

UPLOAD_DIRECTORY -

Returns:

String UPLOAD_DIRECTORY path della cartella

createPathLogo

public java.lang.String createPathLogo(java.util.List<org.apache.tomcat.util.http.fileupload.FileItem> mul
tiparts,

java.lang.String nomeNegozio,

java.lang.String UPLOAD_DIRECTORY)

throws java.lang.Exception

Crea il la stringa del path per il logo del negozio

Parameters:

multiparts -

nomeNegozio -

UPLOAD_DIRECTORY -

Returns:

stringa del path

Throws:

java.lang.Exception

updateLogoNegozio

public boolean updateLogoNegozio(java.lang.String nomeNegozio,

java.lang.String urlLogo)

throws java.sql.SQLException

Modifica url del path del logo negozio

Parameters:

nomeNegozio -

urlLogo -

Returns:

boolean, true se andato a buon fine la modifica altrimenti false

Throws:

java.sql.SQLException

getNegozioByName

public Negozio getNegozioByName(java.lang.String nomeNegozio)

throws java.sql.SQLException,

ParametroNonCorrettoException

Restituire il negozio passato come parametro il nome del negozio

Parameters:

nomeNegozio - il nome del negozio

Returns:

negozio

Throws:

java.sql.SQLException

ParametroNonCorrettoException

getNomeNegozio

public java.lang.String getNomeNegozio()

Returns:

the nomeNegozio

setNomeNegozio

public void setNomeNegozio(java.lang.String nomeNegozio)

Parameters:

nomeNegozio - the nomeNegozio to set

getUsernameVenditore

public java.lang.String getUsernameVenditore()

Returns:

the usernameVenditore

setUsernameVenditore

public void setUsernameVenditore(java.lang.String usernameVenditore)

Parameters:

usernameVenditore - the usernameVenditore to set

getDesign

public java.lang.String getDesign()

Returns:

the design

setDesign

public void setDesign(java.lang.String design)

Parameters:

design - the design to set

getColore

public java.lang.String getColore()

Returns:

the colore

setColore

public void setColore(java.lang.String colore)

Parameters:

colore - the colore to set

getPartitaIva

public java.lang.String getPartitaIva()

Returns:

the partitaIva

setPartitaIva

public void setPartitaIva(java.lang.String partitaIva)

Parameters:

partitaIva - the partitaIva to set

getDataIscrizione

public java.lang.String getDataIscrizione()

Returns:

the dataIscrizione

setDataIscrizione

public void setDataIscrizione(java.lang.String dataIscrizione)

Parameters:

dataIscrizione - the dataIscrizione to set

getDescrizione

public java.lang.String getDescrizione()

Returns:

the descrizione

setDescrizione

public void setDescrizione(java.lang.String descrizione)

Parameters:

descrizione - the descrizione to set

getVia

public java.lang.String getVia()

Returns:

the via

setVia

public void setVia(java.lang.String via)

Parameters:

via - the via to set

getCitta

public java.lang.String getCitta()

Returns:

the citta

setCitta

public void setCitta(java.lang.String citta)

Parameters:

citta - the citta to set

getCap

public java.lang.String getCap()

Returns:

the cap

setCap

public void setCap(java.lang.String cap)

Parameters:

cap - the cap to set

getLogo

public java.lang.String getLogo()

Returns:

the logo

setLogo

public void setLogo(java.lang.String logo)

Parameters:

logo - the logo to set

Class Categoria

java.lang.Object

managernegozio.Categoria

---

public class **Categoria**

extends java.lang.Object

la classe Categoria gestisce le operazioni di modifica,cancellazione e rimozione di una categoria costruzine del path per inserimento delle imagini

Author:

cetra

Constructor Summary

Constructors
Constructor and Description
Categoria() Costruttore vuoto
Categoria(java.lang.String nomeNegozio, java.lang.String nomeCategoria, java.lang.String path, java.lang.String descrizione) Costruttore crea istanza di Categoria

Method Summary

All Methods **Instance Methods** **Concrete Methods**	
Modifier and Type	**Method and Description**
Categoria	**addCategoria**(java.lang.String nomeNegozio, java.lang.String nomeCategor java.lang.String descrizione) Aggiunge una nuova categoria per un determinato negozio Precondizione non 16 nomeCategoria formato: lettere e numeri min 3, max 16 path percorso do automaticamente descrizione formato: lettere cifre min 3 max 500
java.lang.String	**createPathCategoriaImage**(java.util.List<org.apache.tomcat.util.http.fileu java.lang.String nomeNegozio, java.lang.String nomeCategoriaImage, java.la Crea il path per inserimento dell'image della categoria

boolean	**deleteCategory**(java.lang.String nomeNegozio, java.lang.String nomeCateg Cancella la categoria di un determinato negozio
java.util.Collection<**Categoria**>	**getAllCategoryBySeller**(java.lang.String usernameVenditore) Restituisce tutte le categorie di un determinato venditore, o lancia eccezione
Categoria	**getCategoria**(java.lang.String nomeNegozio, java.lang.String nomeCategori Restituisce oggetto categoria di un determinato negozio specificando il nome.
java.lang.String	**getDescrizione**()
java.lang.String	**getNomeCategoria**()
java.lang.String	**getNomeNegozio**()
java.lang.String	**getPath**()
java.lang.String	**openCartellaNegozio**(java.lang.String nomeNegozio, java.lang.String UPLO Crea la cartella nel server per un determinato negozio
void	**setDescrizione**(java.lang.String descrizione)
void	**setNomeCategoria**(java.lang.String nomeCategoria)
void	**setNomeNegozio**(java.lang.String nomeNegozio)
void	**setPath**(java.lang.String path)
boolean	**updateDescrizioneCategoria**(java.lang.String nomeNegozio, java.lang.Stri Modifica la descrizione di una categoria i parametri rispettano i formati e sono
boolean	**updatePathCategoria**(java.lang.String nomeNegozio, java.lang.String nome Modifica il path dell'imagine della categoria i parametri di input sono controlla

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Categoria

public Categoria()

Costruttore vuoto

Categoria

public Categoria(java.lang.String nomeNegozio,
java.lang.String nomeCategoria,

java.lang.String path,

java.lang.String descrizione)

Costruttore crea istanza di Categoria

Parameters:

nomeNegozio -

nomeCategoria -

path -

descrizione -

Method Detail

getAllCategoryBySeller

public java.util.Collection<Categoria> getAllCategoryBySeller(java.lang.String usernameVenditore)

throws java.sql.SQLException,

ParametroNonCorrettoException

Restituisce tutte le categorie di un determinato venditore, o lancia eccezione in caso venditore non esiste

Parameters:

usernameVenditore, - del venditore per restituire tutte le categorie associate

Returns:

ritorna la collezione delle categorie di un determinato venditore

Throws:

java.sql.SQLException

ParametroNonCorrettoException

addCategoria

public Categoria addCategoria(java.lang.String nomeNegozio,

java.lang.String nomeCategoria,

java.lang.String path,

java.lang.String descrizione)

throws java.sql.SQLException

Aggiunge una nuova categoria per un determinato negozio

Precondizione

nomeNegozio formato: lettere e numeri min 3, max 16

nomeCategoria formato: lettere e numeri min 3, max 16

path percorso dove si trova image sul server inserita automaticamente

descrizione formato: lettere cifre min 3 max 500

Parameters:

nomeNegozio -

nomeCategoria -

path -

descrizione -

Returns:

categoria aggiunta

Throws:

java.sql.SQLException

openCartellaNegozio

public java.lang.String openCartellaNegozio(java.lang.String nomeNegozio,

java.lang.String UPLOAD_DIRECTORY)

Crea la cartella nel server per un determinato negozio

Parameters:

nomeNegozio, - nome che viene dato alal cartella

UPLOAD_DIRECTORY, - stringa path base del server

Returns:

String del path della cartella appena creato

createPathCategoriaImage

public java.lang.String createPathCategoriaImage(java.util.List<org.apache.tomcat.util.http.fileupload.FileItem> multiparts,

java.lang.String nomeNegozio,

java.lang.String nomeCategoriaImage,

java.lang.String UPLOAD_DIRECTORY)

throws java.lang.Exception

Crea il path per inserimento dell'image della categoria

Parameters:

multiparts, - lista

nomeNegozio, - serve per la composizione del path

nomeCategoriaImage, - serve per la composizione del path

UPLOAD_DIRECTORY, - stringa base del path per la costruzione

Returns:

String del path dove per image della categoria

Throws:

java.lang.Exception

deleteCategory

public boolean deleteCategory(java.lang.String nomeNegozio,

java.lang.String nomeCategoria)

throws ParametroNonCorrettoException,

java.sql.SQLException

Cancella la categoria di un determinato negozio

Parameters:

nomeNegozio, - nome del negozio di cui cancellare la categoria

nomeCategoria, - nome della categoria da cancellare

Returns:

boolean true se la cancellazione avviene con successo altrimenti lancia un eccezione

Throws:

ParametroNonCorrettoException

java.sql.SQLException

updatePathCategoria

public boolean updatePathCategoria(java.lang.String nomeNegozio,

java.lang.String nomeCategoria,

java.lang.String urlLogo)

throws java.sql.SQLException

Modifica il path dell'imagine della categoria

i parametri di input sono controllati dalla servlet che chiama il metodo

Parameters:

nomeNegozio -

nomeCategoria -

urlLogo -

Returns:

boolean True se andato a buon fine

Throws:

java.sql.SQLException

getCategoria

public Categoria getCategoria(java.lang.String nomeNegozio,

java.lang.String nomeCategoria)

throws java.sql.SQLException

Restituisce oggetto categoria di un determinato negozio specificando il nome.

I paramtri di input sono controllati dalla servlet, prelevati dalla sessione

Parameters:

nomeNegozio -

nomeCategoria -

Returns:

Categoria, dato il nome negozio e il nome categoria

Throws:

java.sql.SQLException

updateDescrizioneCategoria

public boolean updateDescrizioneCategoria(java.lang.String nomeNegozio,
java.lang.String nomeCategoria,
java.lang.String descrizione)
throws java.sql.SQLException

Modifica la descrizione di una categoria

i parametri rispettano i formati e sono controllati sia da javascript che sul server

Parameters:

nomeNegozio -

nomeCategoria -

descrizione -

Returns:

boolean True in caso di successo della modifica altrimenti false

Throws:

java.sql.SQLException

getNomeNegozio

public java.lang.String getNomeNegozio()

Returns:

the nomeNegozio

setNomeNegozio

public void setNomeNegozio(java.lang.String nomeNegozio)

Parameters:

nomeNegozio - the nomeNegozio to set

getNomeCategoria

public java.lang.String getNomeCategoria()

Returns:

the nomeCategoria

setNomeCategoria

public void setNomeCategoria(java.lang.String nomeCategoria)

Parameters:

nomeCategoria - the nomeCategoria to set

getDescrizione

public java.lang.String getDescrizione()

Returns:

the descrizione

setDescrizione

public void setDescrizione(java.lang.String descrizione)

Parameters:

descrizione - the descrizione to set

getPath

public java.lang.String getPath()

Returns:

the path

setPath

public void setPath(java.lang.String path)

Parameters:

path - the path to set

Class Prodotto

java.lang.Object

managernegozio.Prodotto

---

public class **Prodotto**

extends java.lang.Object

la classe Prodotto gestisce le operazioni di modifica, cancellazione di un prodotto

Author:

cetra

Constructor Summary

Constructors	
Constructor and Description	
Prodotto()	
Costruttore vuoto	
Prodotto(int idProdotto, java.lang.String nomeNegozio, java.lang.String nomeCategoria, java.lang.String nome, int iva, java.lang.String path, float prezzo, int qta, int sconto, java.lang.String descrizione)	
Costruttore crea un oggetto Prodotto	

Method Summary

All MethodsInstance MethodsConcrete Methods	
Modifier and Type	**Method and Description**
Prodotto	**addProdotto**(**Prodotto** prodotto)
	Aggiunge un prodotto nel database il prodotto passato in input non deve esser
java.lang.String	**createPathProdottoImage**(java.util.List<org.apache.tomcat.util.http.fileupl java.lang.String nomeNegozio, java.lang.String nomeCategoria, int nomeProd java.lang.String UPLOAD_DIRECTORY)
	Crea il path del image prodotto
boolean	**deleteProduct**(int id)
	Cancella il prodotto tramite il suo id dal database
java.util.Collection<**Prodotto**>	**getAllProductBySellerCategory**(java.lang.String venditore, java.lang.String
	restituisce tutti i prodotti relativa a quella categoria di quel venditore.
java.lang.String	**getDescrizione**()

int	**getIdProdotto**()
int	**getIva**()
java.lang.String	**getNome**()
java.lang.String	**getNomeCategoria**()
java.lang.String	**getNomeNegozio**()
java.lang.String	**getPath**()
java.lang.String	**getPathByID**(int id) Crea il path per inserire image del prodotto
float	**getPrezzo**()
Prodotto	**getProductById**(int idProdotto) restituisce il prodotto tramite il suo id
int	**getQuantita**()
int	**getSconto**()
java.lang.String	**openCartellaNegozio**(java.lang.String nomeNegozio, java.lang.String UPLOA crea la cartella se non esiste se no preleva il path per restituirlo
void	**setDescrizione**(java.lang.String descrizione) questo metodo setta la descrizione di un prodotto
void	**setIdProdotto**(int idProdotto) questo metodo setta l'id di un prodotto
void	**setIva**(int iva) questo metodo setta l'iva di un prodotto
void	**setNome**(java.lang.String nome)
void	**setNomeCategoria**(java.lang.String nomeCategoria)
void	**setNomeNegozio**(java.lang.String nomeNegozio)
void	**setPath**(java.lang.String path) questo metodo setta il path dove è contenuta l'immagine del prodotto
void	**setPrezzo**(float prezzo)

	questo metodo setta il prezzo di un prodotto
void	**setQuantita**(int qta) questo metodo setta la quantità di un prodotto
void	**setSconto**(int sconto) questo metodo setta lo sconto di un prodotto
boolean	**updatePathProdotto**(java.lang.String nomeNegozio, java.lang.String nomeC Modifica il path dell'image del prodotto
boolean	**updateProdotto**(**Prodotto** bean) Modifica il prodotto

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Prodotto

public Prodotto()

Costruttore vuoto

Prodotto

public Prodotto(int idProdotto,
java.lang.String nomeNegozio,
java.lang.String nomeCategoria,
java.lang.String nome,
int iva,
java.lang.String path,
float prezzo,
int qta,
int sconto,
java.lang.String descrizione)

Costruttore crea un oggetto Prodotto

Parameters:

idProdotto -

nomeNegozio -

nomeCategoria -

nome -

iva -

path -

prezzo -

qta -

sconto -

descrizione -

Method Detail

getPathByID

public java.lang.String getPathByID(int id)

throws java.sql.SQLException

Crea il path per inserire image del prodotto

Parameters:

id -

Returns:

string, path dell'image del prodotto

Throws:

java.sql.SQLException

getAllProductBySellerCategory

public java.util.Collection<Prodotto> getAllProductBySellerCategory(java.lang.String venditore,

java.lang.String categoria)

throws java.sql.SQLException,

ParametroNonCorrettoException

restituisce tutti i prodotti relativa a quella categoria di quel venditore.

Precondizione

venditore, categoria devono non essere null

Parameters:

venditore -

categoria -

Returns:

Collection di prodotti relativi a quella categoria di quel venditore

Throws:

java.sql.SQLException

ParametroNonCorrettoException

addProdotto

public Prodotto addProdotto(Prodotto prodotto)

throws java.sql.SQLException,

ParametroNonCorrettoException

Aggiunge un prodotto nel database

il prodotto passato in input non deve essere null

Parameters:

prodotto -

Returns:

Prodotto, restituisce il prodotto appena aggiunto o un eccezione

Throws:

java.sql.SQLException

ParametroNonCorrettoException

openCartellaNegozio

public java.lang.String openCartellaNegozio(java.lang.String nomeNegozio,

java.lang.String UPLOAD_DIRECTORY)

crea la cartella se non esiste se no preleva il path per restituirlo

Parameters:

nomeNegozio -

UPLOAD_DIRECTORY -

Returns:

String, url path dove inserisce image del prodotto

createPathProdottoImage

public java.lang.String createPathProdottoImage(java.util.List<org.apache.tomcat.util.http.fileupload.FileItem> multiparts,

java.lang.String nomeNegozio,

java.lang.String nomeCategoria,

int nomeProdottoImage,

java.lang.String UPLOAD_DIRECTORY)

throws java.lang.Exception

Crea il path del image prodotto

Parameters:

multiparts -

nomeNegozio -

nomeCategoria -

nomeProdottoImage - id del prodotto

UPLOAD_DIRECTORY -

Returns:

String path del logo

Throws:

java.lang.Exception

updatePathProdotto

public boolean updatePathProdotto(java.lang.String nomeNegozio,
java.lang.String nomeCategoria,
int id,
java.lang.String logo)
throws java.sql.SQLException

Modifica il path dell'image del prodotto

Parameters:

nomeNegozio -

nomeCategoria -

id -

logo -

Returns:

boolean true in caso di successo

Throws:

java.sql.SQLException

deleteProduct

public boolean deleteProduct(int id)
throws java.sql.SQLException

Cancella il prodotto tramite il suo id dal database

Parameters:

id -

Returns:

boolean true se la cancellazione ha avuto successo

Throws:

java.sql.SQLException

getProductById

public Prodotto getProductById(int idProdotto)
throws java.sql.SQLException

restituisce il prodotto tramite il suo id

Parameters:

idProdotto -

Returns:

Prodotto

Throws:

java.sql.SQLException

updateProdotto

public boolean updateProdotto(Prodotto bean)

throws java.sql.SQLException

Modifica il prodotto

Parameters:

bean -

Returns:

boolean true se avvenuta la modifica

Throws:

java.sql.SQLException

getIdProdotto

public int getIdProdotto()

Returns:

int

setIdProdotto

public void setIdProdotto(int idProdotto)

questo metodo setta l'id di un prodotto

Parameters:

id - è l'id da assegnare

getIva

public int getIva()

Returns:

int

setIva

public void setIva(int iva)

questo metodo setta l'iva di un prodotto

Parameters:

iva - è l'iva da assegnare

getPath

public java.lang.String getPath()

Returns:

string

setPath

public void setPath(java.lang.String path)

questo metodo setta il path dove è contenuta l'immagine del prodotto

Parameters:

path - è il path da assegnare

getPrezzo

public float getPrezzo()

Returns:

float

setPrezzo

public void setPrezzo(float prezzo)

questo metodo setta il prezzo di un prodotto

Parameters:

prezzo - è il prezzo da assegnare

getQuantita

public int getQuantita()

Returns:

int

setQuantita

public void setQuantita(int qta)

questo metodo setta la quantità di un prodotto

Parameters:

qtà - è la quantità da assegnare

getSconto

public int getSconto()

Returns:

int

setSconto

public void setSconto(int sconto)

questo metodo setta lo sconto di un prodotto

Parameters:

sconto - è la sconto da assegnare

getDescrizione

public java.lang.String getDescrizione()

Returns:

string

setDescrizione

public void setDescrizione(java.lang.String descrizione)

questo metodo setta la descrizione di un prodotto

Parameters:

descrizione - è la descrizione da assegnare

getNomeNegozio

public java.lang.String getNomeNegozio()

Returns:

the nomeNegozio

setNomeNegozio

public void setNomeNegozio(java.lang.String nomeNegozio)

Parameters:

nomeNegozio - the nomeNegozio to set

getNomeCategoria

public java.lang.String getNomeCategoria()

Returns:

the nomeCategoria

setNomeCategoria

public void setNomeCategoria(java.lang.String nomeCategoria)

Parameters:

nomeCategoria - the nomeCategoria to set

getNome

public java.lang.String getNome()

Returns:

the nome

setNome

public void setNome(java.lang.String nome)

Parameters:

nome - the nome to set

Class ArrayFattura

java.lang.Object

managerordine.ArrayFattura

---

public class **ArrayFattura**

extends java.lang.Object

Permette di gestire una lista di fatture con operazioni aggiunta

Author:

cetra

Constructor Summary

Constructors	
Constructor and Description	
ArrayFattura() costruttore che crea una arrylist di fatture vuoto	

Method Summary

All Methods	Instance Methods	Concrete Methods
Modifier and Type	**Method and Description**	
void	**add**(**Fattura** f) Aggiunge una fattura alla lista	
java.util.ArrayList<**Fattura**>	**getAllFatture**() restituisce la lista delle fatture	

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ArrayFattura

public ArrayFattura()

costruttore che crea una arrylist di fatture vuoto

Method Detail

add

public void add(Fattura f)

Aggiunge una fattura alla lista

Parameters:

f - Fattura

getAllFatture

public java.util.ArrayList<Fattura> getAllFatture()

restituisce la lista delle fatture

Returns: Restituisce tutte le fatture

Class ArrayRiferimento

java.lang.Object

managerordine.ArrayRiferimento

---

public class **ArrayRiferimento**

extends java.lang.Object

Ha le funzionalità per gestire un array di oggetti riferimento

Author:

cetra

Constructor Summary

Constructors
Constructor and Description
ArrayRiferimento() Costruttore crea un lista riferimento vuota

Method Summary

All Methods	Instance Methods	Concrete Methods

Modifier and Type	Method and Description
void	**add**(**Riferimento** f) Aggiunge un riferimento alla lista
java.util.ArrayList<**Riferimento**>	**getAllRiferimento**() restituisce la lista riferimento

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ArrayRiferimento

public ArrayRiferimento()

Costruttore crea un lista riferimento vuota

Method Detail

add

public void add(Riferimento f)

Aggiunge un riferimento alla lista

Parameters:

f -

getAllRiferimento

public java.util.ArrayList<Riferimento> getAllRiferimento()

restituisce la lista riferimento

Returns:

ArrayList

Class Fattura

java.lang.Object

managerordine.Fattura

---

public class **Fattura**

extends java.lang.Object

Author:

manlio la classe Fattura gestisce le operazioni di un ordine

Constructor Summary

Constructors
Constructor and Description
Fattura() Costruttore vuoto
Fattura(int numFattura, java.lang.String username_Cliente, java.lang.String nomeNegozio, java.lang.String dataOrdine, java.lang.String descrizione, java.lang.String viaPartenza, java.lang.String cittaPartenza, java.lang.String capPartenza, java.lang.String viaDestinazione, java.lang.String cittaDestinazione, java.lang.String capDestinazione) Costruttore che crea una fattura

Method Summary

All Methods **Instance Methods** **Concrete Methods**	
Modifier and Type	**Method and Description**
java.lang.String	**getCapDestinazione**()
java.lang.String	**getCittaDestinazione**()
java.lang.String	**getDataOrdine**()

Riferimento	**getListaRiferimento**()
java.lang.String	**getNomeNegozio**()
int	**getNumFattura**()
java.lang.String	**getUsername_Cliente**()
java.lang.String	**getViaDestinazione**()
void	**setCapDestinazione**(java.lang.String capDestinazione)
void	**setCittaDestinazione**(java.lang.String cittaDestinazione)
void	**setDataOrdine**(java.lang.String dataOrdine)
void	**setListaRiferimento**(**Riferimento** listaRiferimento)
void	**setNomeNegozio**(java.lang.String nomeNegozio)
void	**setNumFattura**(int numFattura)
void	**setUsername_Cliente**(java.lang.String username_Cliente)
void	**setViaDestinazione**(java.lang.String viaDestinazione)
java.lang.String	**toString**()

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

Fattura

public Fattura()

Costruttore vuoto

Fattura

public Fattura(int numFattura,

java.lang.String username_Cliente,

java.lang.String nomeNegozio,

java.lang.String dataOrdine,

java.lang.String descrizione,

java.lang.String viaPartenza,

java.lang.String cittaPartenza,

java.lang.String capPartenza,

java.lang.String viaDestinazione,

java.lang.String cittaDestinazione,

java.lang.String capDestinazione)

Costruttore che crea una fattura

Parameters:

numFattura -

username_Cliente -

nomeNegozio -

dataOrdine -

descrizione -

viaPartenza -

cittaPartenza -

capPartenza -

viaDestinazione -

cittaDestinazione -

capDestinazione -

Method Detail

getUsername_Cliente

public java.lang.String getUsername_Cliente()

Returns:

the username_Cliente

setUsername_Cliente

public void setUsername_Cliente(java.lang.String username_Cliente)

Parameters:

username_Cliente - the username_Cliente to set

getNomeNegozio

public java.lang.String getNomeNegozio()

Returns:

the nomeNegozio

setNomeNegozio

public void setNomeNegozio(java.lang.String nomeNegozio)

Parameters:

nomeNegozio - the nomeNegozio to set

getListaRiferimento

public Riferimento getListaRiferimento()

Returns:

the listaRiferimento

setListaRiferimento

public void setListaRiferimento(Riferimento listaRiferimento)

Parameters:

listaRiferimento - the listaRiferimento to set

getDataOrdine

public java.lang.String getDataOrdine()

Returns:

the dataOrdine

setDataOrdine

public void setDataOrdine(java.lang.String dataOrdine)

Parameters:

dataOrdine - the dataOrdine to set

getViaDestinazione

public java.lang.String getViaDestinazione()

Returns:

the viaDestinazione

setViaDestinazione

public void setViaDestinazione(java.lang.String viaDestinazione)

Parameters:

viaDestinazione - the viaDestinazione to set

getCittaDestinazione

public java.lang.String getCittaDestinazione()

Returns:

the cittaDestinazione

setCittaDestinazione

public void setCittaDestinazione(java.lang.String cittaDestinazione)

Parameters:

cittaDestinazione - the cittaDestinazione to set

getCapDestinazione

public java.lang.String getCapDestinazione()

Returns:

the capDestinazione

setCapDestinazione

public void setCapDestinazione(java.lang.String capDestinazione)

Parameters:

capDestinazione - the capDestinazione to set

getNumFattura

public int getNumFattura()

Returns:

the numFattura

setNumFattura

public void setNumFattura(int numFattura)

Parameters:

numFattura - the numFattura to set

toString

public java.lang.String toString()

Overrides:

toString in class java.lang.Object

Class Riferimento

java.lang.Object

managerordine.Riferimento

---

public class **Riferimento**

extends java.lang.Object

Contiene i dati permenenti di una fattura, ha i metodi getter e setter per modificare l'oggetto riferimento inoltre permette di gestire arrayriferimento.

Author:

cetra

Constructor Summary

Constructors
Constructor and Description
Riferimento() costruttore vuoto
Riferimento(int id_prodotto, int numero_Fattura, java.lang.String nome_Negozio, java.lang.String nome_Categoria, int qtaOrdinata, int sconto, float prezzoUnitario, int iva) Costruttore che crea un oggetto Riferimento
Riferimento(int id_prodotto, int numero_Fattura, java.lang.String nome_Negozio, java.lang.String nome_Categoria, int qtaOrdinata, int sconto, float prezzoUnitario, int iva, java.lang.String note) Costruttore che crea un oggetto riferimento con nota

Method Summary

All MethodsInstance MethodsConcrete Methods	
Modifier and Type	**Method and Description**
ArrayRiferimento	**getArrayRiferimento**(java.lang.String nomeNegozio) Restituisce una lista di oggetti riferimenti associati ad un negozio nomeNegozio diverso da null
Fattura	**getFatturaRiferimento**() restituisce la fattura referita
int	**getId_prodotto**()
int	**getIva**()
java.lang.String	**getNome_Categoria**()
java.lang.String	**getNome_Negozio**()

java.lang.String	**getNote**()
int	**getNumero_Fattura**()
float	**getPrezzoUnitario**()
int	**getQtaOrdinata**()
int	**getSconto**()
void	**setFatturaRiferimento**(**Fattura** fatturaRiferimento) setta la fattura riferimento
void	**setId_prodotto**(int id_prodotto)
void	**setIva**(int iva)
void	**setNome_Categoria**(java.lang.String nome_Categoria)
void	**setNome_Negozio**(java.lang.String nome_Negozio)
void	**setNote**(java.lang.String note)
void	**setNumero_Fattura**(int numero_Fattura)
void	**setPrezzoUnitario**(float prezzoUnitario)
void	**setQtaOrdinata**(int qtaOrdinata)
void	**setSconto**(int sconto)
java.lang.String	**toString**()

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

Riferimento

public Riferimento()

costruttore vuoto

Riferimento

public Riferimento(int id_prodotto,
int numero_Fattura,
java.lang.String nome_Negozio,
java.lang.String nome_Categoria,

int qtaOrdinata,

int sconto,

float prezzoUnitario,

int iva)

Costruttore che crea un oggetto Riferimento

Parameters:

id_prodotto -

numero_Fattura -

nome_Negozio -

nome_Categoria -

qtaOrdinata -

sconto -

prezzoUnitario -

iva -

Riferimento

public Riferimento(int id_prodotto,

int numero_Fattura,

java.lang.String nome_Negozio,

java.lang.String nome_Categoria,

int qtaOrdinata,

int sconto,

float prezzoUnitario,

int iva,

java.lang.String note)

Costruttore che crea un oggetto riferimento con nota

Parameters:

id_prodotto -

numero_Fattura -

nome_Negozio -

nome_Categoria -

qtaOrdinata -

sconto -

prezzoUnitario -

iva -

note -

Method Detail

getFatturaRiferimento

public Fattura getFatturaRiferimento()

restituisce la fattura referita

Returns:

Fattura

setFatturaRiferimento

public void setFatturaRiferimento(Fattura fatturaRiferimento)

setta la fattura riferimento

Parameters:

fatturaRiferimento -

toString

public java.lang.String toString()

Overrides:

toString in class java.lang.Object

getArrayRiferimento

public ArrayRiferimento getArrayRiferimento(java.lang.String nomeNegozio)
throws ParametroNonCorrettoException,
java.sql.SQLException

Restituisce una lista di oggetti riferimenti associati ad un negozio

nomeNegozio diverso da null

Parameters:

nomeNegozio -

Returns:

Throws:

ParametroNonCorrettoException

java.sql.SQLException

getId_prodotto

public int getId_prodotto()

Returns:

the id_prodotto

setId_prodotto

public void setId_prodotto(int id_prodotto)

Parameters:

id_prodotto - the id_prodotto to set

getNumero_Fattura

public int getNumero_Fattura()

Returns:

the numero_Fattura

setNumero_Fattura

public void setNumero_Fattura(int numero_Fattura)

Parameters:

numero_Fattura - the numero_Fattura to set

getNome_Negozio

public java.lang.String getNome_Negozio()

Returns:

the nome_Negozio

setNome_Negozio

public void setNome_Negozio(java.lang.String nome_Negozio)

Parameters:

nome_Negozio - the nome_Negozio to set

getNome_Categoria

public java.lang.String getNome_Categoria()

Returns:

the nome_Categoria

setNome_Categoria

public void setNome_Categoria(java.lang.String nome_Categoria)

Parameters:

nome_Categoria - the nome_Categoria to set

getQtaOrdinata

public int getQtaOrdinata()

Returns:

the qtaOrdinata

setQtaOrdinata

public void setQtaOrdinata(int qtaOrdinata)

Parameters:

qtaOrdinata - the qtaOrdinata to set

getSconto

public int getSconto()

Returns:

the sconto

setSconto

public void setSconto(int sconto)

Parameters:

sconto - the sconto to set

getPrezzoUnitario

public float getPrezzoUnitario()

Returns:

the prezzoUnitario

setPrezzoUnitario

public void setPrezzoUnitario(float prezzoUnitario)

Parameters:

prezzoUnitario - the prezzoUnitario to set

getIva

public int getIva()

Returns:

the iva

setIva

public void setIva(int iva)

Parameters:

iva - the iva to set

getNote

public java.lang.String getNote()

Returns:

the note

setNote

public void setNote(java.lang.String note)

Parameters:

note - the note to set

Package model

Class Summary	
Class	**Description**
CategoriaDAO	Posside le operazioni di gestione la table Categoria all'interno del database
DriverManagerConnectionPool	Connessione con il db tramite ConnectionPool

NegozioDAO	Realizza le operazioni da effettuare con li databese riferite alla table Negozio
ProdottoDAO	Permette di gestire la table Prodotto nel database con le operazioni più comuni
RiferimentoDAO	Permette di operare sulla table riferimento per operazioni CRUD
VenditoreDAO	Permette di effettuare le operazioni CRUD per la table Venditore

Class CategoriaDAO

java.lang.Object

model.CategoriaDAO

All Implemented Interfaces:

java.io.Serializable

---

public class **CategoriaDAO**

extends java.lang.Object

implements java.io.Serializable

Posside le operazioni di gestione la table Categoria all'interno del database

Author:

cetra

See Also:

Serialized Form

Constructor Summary

Constructors
Constructor and Description
CategoriaDAO()

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	**Method and Description**

void	**addCategoria**(**Categoria** categoria) Aggiunge una nuova categoria Il nome del negozio, nome venditore deve essere già presente nel database Il nome della categoria non può essere inserito in caso sia già presente con lo stesso nome, nome del venditore, negozio
boolean	**deleteCategory**(java.lang.String nomeNegozio, java.lang.String nomeCategoria) Cancella la categoria di un determinato negozio il nome negozio e la categoria associata, da cancellare devono essere presenti nel db
java.util.Collection<**Categoria**>	**getAllCategoryBySeller**(java.lang.String venditore) Restituisce una lista di categorie di un determinato venditore
Categoria	**getCategoria**(java.lang.String nomeNegozio, java.lang.String nomeCategoria) Restituisce la categoria di un determinato negozio nel database deve essere presente il negozio, e la categoria associata a quel negozio
boolean	**updateDescrizioneCategoria**(java.lang.String nomeNegozio, java.lang.String nomeCategoria, java.lang.String descrizione) Modifica la descrizione di una categoria nel database devono essere presente nome negozio e il nome categoria associato al negozio
boolean	**updatePathCategoria**(java.lang.String nomeNegozio, java.lang.String nomeCategoria, java.lang.String logo) Modifica il path della categoria nel caso il path non sia già aggiunto lo inserisce il parametro nomeNegozio deve essere presente nel database il parametro nomeCategoria deve essere riferito al nomeNegozio è deve essere presente nel db

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

CategoriaDAO

public CategoriaDAO()

Method Detail

getAllCategoryBySeller

public java.util.Collection<Categoria> getAllCategoryBySeller(java.lang.String venditore)

throws java.sql.SQLException

Restituisce una lista di categorie di un determinato venditore

Parameters:

venditore -

Returns:

Collection, lista di categoria

Throws:

java.sql.SQLException

addCategoria

public void addCategoria(Categoria categoria)

throws java.sql.SQLException

Aggiunge una nuova categoria

Il nome del negozio, nome venditore deve essere già presente nel database

Il nome della categoria non può essere inserito in caso sia già presente con lo stesso nome, nome del venditore, negozio

Parameters:

categoria -

Throws:

java.sql.SQLException

updatePathCategoria

public boolean updatePathCategoria(java.lang.String nomeNegozio,

java.lang.String nomeCategoria,

java.lang.String logo)

throws java.sql.SQLException

Modifica il path della categoria

nel caso il path non sia già aggiunto lo inserisce

il parametro nomeNegozio deve essere presente nel database

il parametro nomeCategoria deve essere riferito al nomeNegozio è deve essere presente nel db

Parameters:

nomeNegozio -

nomeCategoria -

logo -

Returns:

boolean true in caso di successo della modifica

Throws:

java.sql.SQLException

getCategoria

public Categoria getCategoria(java.lang.String nomeNegozio,

java.lang.String nomeCategoria)

throws java.sql.SQLException

Restituisce la categoria di un determinato negozio

nel database deve essere presente il negozio, e la categoria associata a quel negozio

Parameters:

nomeNegozio -

nomeCategoria -

Returns:

restituisce la categoria

Throws:

java.sql.SQLException

updateDescrizioneCategoria

public boolean updateDescrizioneCategoria(java.lang.String nomeNegozio,
java.lang.String nomeCategoria,
java.lang.String descrizione)
throws java.sql.SQLException

Modifica la descrizione di una categoria

nel database devono essere presente nome negozio e il nome categoria associato al negozio

Parameters:

nomeNegozio -

nomeCategoria -

descrizione -

Returns:

flag boolean true se la modifica avviene altrimenti false

Throws:

java.sql.SQLException

deleteCategory

public boolean deleteCategory(java.lang.String nomeNegozio,
java.lang.String nomeCategoria)
throws java.sql.SQLException

Cancella la categoria di un determinato negozio

il nome negozio e la categoria associata, da cancellare devono essere presenti nel db

Parameters:

nomeNegozio -

nomeCategoria -

Returns:

boolean true se la cancellazione avviene con successo

Throws:

java.sql.SQLException

Class DriverManagerConnectionPool

java.lang.Object

model.DriverManagerConnectionPool

---

public class **DriverManagerConnectionPool**

extends java.lang.Object

Connessione con il db tramite ConnectionPool

Author:

cetra

Constructor Summary

Constructors
Constructor and Description
DriverManagerConnectionPool()

Method Summary

All Methods **Static Methods** **Concrete Methods**	
Modifier and Type	**Method and Description**
static java.sql.Connection	**createDBConnection**()
static java.sql.Connection	**getDbConnection**()
static void	**releaseConnection**(java.sql.Connection connection)

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

DriverManagerConnectionPool

public DriverManagerConnectionPool()

Method Detail

createDBConnection

public static java.sql.Connection createDBConnection()
throws java.sql.SQLException

Throws:

java.sql.SQLException

getDbConnection

public static java.sql.Connection getDbConnection()
throws java.sql.SQLException

Throws:

java.sql.SQLException

releaseConnection

public static void releaseConnection(java.sql.Connection connection)

Class NegozioDAO

java.lang.Object

model.NegozioDAO

All Implemented Interfaces:

java.io.Serializable

---

public class **NegozioDAO**

extends java.lang.Object

implements java.io.Serializable

Realizza le operazioni da effettuare con li databese riferite alla table Negozio

Author:

cetra

See Also:

Serialized Form

Constructor Summary

Constructors
Constructor and Description
NegozioDAO()

Method Summary

All MethodsInstance MethodsConcrete Methods	
Modifier and Type	**Method and Description**
void	**addNegozio**(**Negozio** negozio) Aggiunge un negozio nel db il negozio deve rispettare il formato per essere inserito
void	**deleteShop**(java.lang.String nomeNegozio) Cancella un negozio
Negozio	**getNegozio**(java.lang.String usernameVenditore) Restituisce il negozio di un venditore Il venditore deve essere presente nel db e deve aver creato un negozio
Negozio	**getNegozioByName**(java.lang.String negozio) Restituisce il nome del negozio Il negozio deve essere già presente con nome uguale al parametro negozio
boolean	**updateLogoNegozio**(java.lang.String nomeNegozio, java.lang.String logo) Modifica il path del logo del negozio Il negozio deve essere già presente

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

NegozioDAO

public NegozioDAO()

Method Detail

getNegozio

public Negozio getNegozio(java.lang.String usernameVenditore)

throws java.sql.SQLException

Restituisce il negozio di un venditore

Il venditore deve essere presente nel db e deve aver creato un negozio

Parameters:

usernameVenditore -

Returns:

Negozio, riferito ad un venditore

Throws:

java.sql.SQLException

addNegozio

public void addNegozio(Negozio negozio)
throws java.sql.SQLException

Aggiunge un negozio nel db

il negozio deve rispettare il formato per essere inserito

Parameters:

negozio -

Throws:

java.sql.SQLException

updateLogoNegozio

public boolean updateLogoNegozio(java.lang.String nomeNegozio,
java.lang.String logo)
throws java.sql.SQLException

Modifica il path del logo del negozio

Il negozio deve essere già presente

Parameters:

nomeNegozio -

logo -

Returns:

boolean, true se avvenuto la modifica

Throws:

java.sql.SQLException

getNegozioByName

public Negozio getNegozioByName(java.lang.String negozio)

throws java.sql.SQLException

Restituisce il nome del negozio

Il negozio deve essere già presente con nome uguale al parametro negozio

Parameters:

negozio -

Returns:

Negozio

Throws:

java.sql.SQLException

deleteShop

public void deleteShop(java.lang.String nomeNegozio)

throws java.sql.SQLException

Cancella un negozio

Parameters:

nomeNegozio -

Throws:

java.sql.SQLException

Class ProdottoDAO

java.lang.Object

model.ProdottoDAO

All Implemented Interfaces:

java.io.Serializable

---

public class **ProdottoDAO**

extends java.lang.Object

implements java.io.Serializable

Permette di gestire la table Prodotto nel database con le operazioni più comuni

Author:

cetra

See Also:

Serialized Form

Constructor Summary

Constructors
Constructor and Description
ProdottoDAO()

Method Summary

All Methods	Instance Methods	Concrete Methods

Modifier and Type	Method and Description
Prodotto	**addProdotto**(**Prodotto** prodotto) Aggiunge un prodotto alla categoria specificata La categoria, negozio devono essere presenti nel database.
boolean	**deleteProduct**(int id) Cancella il prodotto tramite id del prodotto id del prodotto deve essere presente nel database
java.util.Collection<**Prodotto**>	**getAllProductBySellerCategory**(java.lang.String venditore, java.lang.String categoria) Restituisce la collezione di prodotti in base al venditore e categoria i parametri venditori e categoria devono essere presenti nel database.
int	**getIDProd**(java.lang.String cat, java.lang.String neg, java.lang.String nomep) Restituisce id del negozio tramite la categoria negozio e prodotto Negozio deve essere presente avere la categoria indicata nel parametro cat.
java.lang.String	**getPathByID**(int id) Restituisc il path (url dove sono salvate le immagini del negozio) del prodotto tramite id del negozio deve essere presente nel database
Prodotto	**getProductById**(int idProdotto) Restituisce il prodotto, tramite il suo id deve essere presente nel database
boolean	**updatePathProdotto**(java.lang.String nomeNegozio, java.lang.String nomeCategoria, int id, java.lang.String logo) Ritorna true o false a seconda se il path è stato modificato Nome negozio deve essere presente nel db , la categoria deve essere associata al negozio e deve essere presente nel db table categoria, id del prodotto deve essere riferito allo stesso negozio e categoria.

boolean	**updateProdotto**(**Prodotto** bean) Modifica il prodotto Id del prodotto deve essere presente nel database, nome negozio e nome categoria devono essere associati nel database allo stesso id prodotto

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ProdottoDAO

public ProdottoDAO()

Method Detail

getPathByID

public java.lang.String getPathByID(int id)

throws java.sql.SQLException

Restituisc il path (url dove sono salvate le immagini del negozio) del prodotto tramite id

ID del negozio deve essere presente nel database

Parameters:

id -

Returns:

String, path del negozio

Throws:

java.sql.SQLException

getIDProd

public int getIDProd(java.lang.String cat,

java.lang.String neg,

java.lang.String nomep)

throws java.sql.SQLException

Restituisce id del negozio tramite la categoria negozio e prodotto

Negozio deve essere presente avere la categoria indicata nel parametro cat.

Nella categoria deve essere presente il prodotto riferito con nome nomep.

Parameters:

cat -

neg -

nomep -

Returns:

int, id del negozio

Throws:

java.sql.SQLException

getAllProductBySellerCategory

public java.util.Collection<Prodotto> getAllProductBySellerCategory(java.lang.String venditore,
java.lang.String categoria)
throws java.sql.SQLException

Restituisce la collezione di prodotti in base al venditore e categoria

i parametri venditori e categoria devono essere presenti nel database.

la categoria deve essere associata al negozio del venditore passato per parametro.

Parameters:

venditore -

categoria -

Returns:

Collection, lista di prodotti

Throws:

java.sql.SQLException

addProdotto

public Prodotto addProdotto(Prodotto prodotto)
throws java.sql.SQLException

Aggiunge un prodotto alla categoria specificata

La categoria, negozio devono essere presenti nel database.

Il negozio deve avere la categoria, specificata per insirire il prodotto.

Parameters:

prodotto -

Returns:

Prodotto aggiunto

Throws:

java.sql.SQLException

updatePathProdotto

public boolean updatePathProdotto(java.lang.String nomeNegozio,
java.lang.String nomeCategoria,
int id,
java.lang.String logo)
throws java.sql.SQLException

Ritorna true o false a seconda se il path è stato modificato

Nome negozio deve essere presente nel db ,

la categoria deve essere associata al negozio e deve essere presente nel db table categoria,

id del prodotto deve essere riferito allo stesso negozio e categoria.

Parameters:

nomeNegozio -

nomeCategoria -

id -

logo -

Returns:

boolean, true se la modifica è avvenuta

Throws:

java.sql.SQLException

deleteProduct

public boolean deleteProduct(int id)

throws java.sql.SQLException

Cancella il prodotto tramite id del prodotto

id del prodotto deve essere presente nel database

Parameters:

id -

Returns:

Throws:

java.sql.SQLException

getProductById

public Prodotto getProductById(int idProdotto)

throws java.sql.SQLException

Restituisce il prodotto, tramite il suo id

id deve essere presente nel database

Parameters:

idProdotto -

Returns:

Prodotto

Throws:

java.sql.SQLException

updateProdotto

public boolean updateProdotto(Prodotto bean)

throws java.sql.SQLException

Modifica il prodotto

Id del prodotto deve essere presente nel database,

nome negozio e nome categoria devono essere associati nel database allo stesso id prodotto

Parameters:

bean -

Returns:

booleam, true se il prodotto viene modificato

Throws:

java.sql.SQLException

Class RiferimentoDAO

java.lang.Object

model.RiferimentoDAO

All Implemented Interfaces:

java.io.Serializable

---

public class **RiferimentoDAO**

extends java.lang.Object

implements java.io.Serializable

Permette di operare sulla table riferimento per operazioni CRUD

Author:

cetra

See Also:

Serialized Form

Constructor Summary

Constructors	
Constructor and Description	
RiferimentoDAO()	

Method Summary

All Methods Instance Methods Concrete Methods	
Modifier and Type	**Method and Description**
ArrayRiferimento	**getArrayRiferimento**(java.lang.String nomeNegozio) Restituisce la lista di riferimento di un determinato negozio Il negozio deve essere presente nel db

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

RiferimentoDAO

public RiferimentoDAO()

Method Detail

getArrayRiferimento

public ArrayRiferimento getArrayRiferimento(java.lang.String nomeNegozio)
throws java.sql.SQLException

Restituisce la lista di riferimento di un determinato negozio

Il negozio deve essere presente nel db

Parameters:

nomeNegozio -

Returns:

ArrayRiferimento lista che contiene tutte le fatture del determinato negozio passato come input

Throws:

java.sql.SQLException

Class VenditoreDAO

java.lang.Object

model.VenditoreDAO

All Implemented Interfaces:

java.io.Serializable

---

public class **VenditoreDAO**

extends java.lang.Object

implements java.io.Serializable

Permette di effettuare le operazioni CRUD per la table Venditore

Author:

cetra

See Also:

Serialized Form

Constructor Summary

Constructors
Constructor and Description
VenditoreDAO()

Method Summary

All Methods Instance Methods Concrete Methods	
Modifier and Type	**Method and Description**
void	**addVenditore**(**Venditore**) Aggiunge un venditore nel database Username e password non devono essere giù presenti nel db
Utente	**checkLoginSeller**(java.lang.String username, java.lang.String password) Verifica la presenza di un venditore nel database Controlla se quello restituito dal database è null
void	**deleteVenditore**(java.lang.String username) Cancella il venditore dal database tramite username deve essere presente nel database

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

VenditoreDAO

public VenditoreDAO()

Method Detail

checkLoginSeller

public Utente checkLoginSeller(java.lang.String username,
java.lang.String password)
throws java.sql.SQLException

Verifica la presenza di un venditore nel database

Controlla se quello restituito dal database è null

Parameters:

username -

password -

Returns:

Utente

Throws:

java.sql.SQLException

addVenditore

public void addVenditore(Venditore)

throws java.sql.SQLException

Aggiunge un venditore nel database

Username e password non devono essere giù presenti nel db

Parameters:

venditore -

Throws:

java.sql.SQLException

deleteVenditore

public void deleteVenditore(java.lang.String username)

throws java.sql.SQLException

Cancella il venditore dal database tramite username

Username deve essere presente nel database

Parameters:

username -

Throws:

java.sql.SQLException

Package systemtesting

Class Summary

Class	Description
LoginVenditoreTest	
RegistrazioneVenditoreTest	