Progetto applicazione web "Agenda appuntamenti"

Gruppo: Salamanna Stefano, Chetta Flavio, Pisanò Luca

Nome Applicazione: Agenda;

Web Container: Apache Tomcat;

DBMS: MySql pacchetto XAMPP;

Url: "localhost:8080/Agenda";

Nome utente DB: root;

Password DB: stefano83.

Come avviare l'applicazione:

- importare in Eclipse il progetto Agenda.war;
- installare ed avviare il server Apache Tomcat;
- importare il database Mysql "agenda" in un web server (ex. Apache del pacchetto XAMPP) ed impostare le credenziali di accesso sopra-riportate, oppure modificarle dalla classe "Database.java" dell'applicazione impostando quelle proprie;
- la libreria di connessione al database "mysql-connector-java-5.1.27-bin.jar" è stata già inserita all'interno della cartella "lib" dell'applicazione, comunque facoltativamente si può anche copiare nella cartella "lib" di Tomcat;
- avviare l'applicazione in locale con l'url sopra indicato.

Per poter accedere all'applicazione tramite login ad ora è possibile utilizzare le seguenti credenziali utente di esempio (ovviamente modificabili in seguito con possibilità di nuovi inserimenti):

Username	**Password**
stefano83	corso
pietro76	meucci
giuseppe65	musica

Di seguito la descrizione delle varie Java Server Pages, Classi e Servlet Java che costituiscono l'applicazione, implementata attraverso il framework Struts con relativo

aggiornamento del file struts-config.xml tramite la definizione delle varie action tra le risorse del pattern MVC.
L'Action Standard del Mapping delle Servlet è definito dall'url con estensione ".do"

Database.java

E' la classe che si occupa di gestire il database con le relative connessioni, attraverso il driver JDBC mysql-connector che è stato scaricato e copiato all'interno della cartella lib dell'applicazione ed in quella di Tomcat.
Si è proceduto quindi all'importazione delle relative librerie di connessione e del driver, quindi alla costruzione della classe Database con gli attributi del driver e l'URI indicante i parametri di connessione (host, dbms, db, utente e password).
Quindi sono stati implementati i metodi di istanza e connessione che vengono invocati dalle altre classi che devono utilizzare il database; è stata inserita anche la gestione delle eccezioni in caso di problemi con relativi messaggi.
Infine l'ultimo metodo serve per chiudere la connessione.

Utenti.java

In questa classe definiamo tutti gli attributi e metodi costruttori degli oggetti utenti, quindi Nome, Cognome, Username, Password e Immagine, dati che come vedremo vengono richiesti in fase di registrazione.

UtentiDB.java

In questa classe definiamo determinate operazioni riguardanti l'entità Database tramite l'incapsulamento di una serie di operazioni indipendenti dal sistema di persistenza.
Qui sono contenute le istruzioni di accesso al database e le query di inserimento degli utenti al suo interno, con degli oggetti statement che provvedono man mano al riempimento dei campi tramite i parametri passati dal form di registrazione.
Ovviamente anche qui vengono gestite alcune eccezioni derivanti da problemi di connessione o inserimento dati.

benvenuto.jsp

E' la pagina iniziale di benvenuto dell'applicazione, contenente un'area login ed un link che reindirizza ad un form di registrazione.
Nel primo caso viene utilizzata una form action che si attiva alla pressione del pulsante di login, in modo da passare i parametri alla successiva servlet di controllo, mentre nel secondo caso è impostato un comando href che reindirizza alla pagina del form di registrazione.

ControlloLogin.java

E' la servlet java predisposta al controllo del login; si importano le librerie sql per la connessione ed interrogazione del database nonché ovviamente quelle necessarie all'implementazione delle request e response della servlet.
Dunque nelle prime righe si ottengono i parametri user e password inseriti dal form, e come primo controllo si verifica che i campi non siano vuoti; nel caso uno dei due lo fosse allora si viene reindirizzati ad una pagina di errore.
In caso positivo invece si apre una connessione al database in modo da effettuare la query dopo aver creato un oggetto della classe che implementa lo statement.
La select verifica quindi la presenza dell'username e password, quindi si procede al controllo da parte del metodo rs.next che scorre le varie righe dei rispettivi record.
Se l'username viene trovata, si procede al controllo della password, che ovviamente deve corrispondere a quel determinato record (per questo tale controllo è interno all'oggetto di resultset).
In caso positivo si procede quindi al prelevamento completo dei dati utente interrogando la relativa tabella che ha l'username come chiave primaria.
Si procede quindi alla creazione di un altro oggetto di "risultati" che restituisce i dati utente, i quali vengono impostati come attributi per la successiva classe GestioneLogin, sui cui viene impostato il dispatcher di indirizzamento delle request/response.
Nei casi in cui la username o la password siano errate, si viene reindirizzati ad una pagina di errore.

GestioneLogin.java

Questa è la servlet di gestione del login che riceve gli attributi delle credenziali inserite e già controllate in precedenza, a loro volta impostate come attributi da passare all'area riservata successiva.

errorelog.jsp

E' la pagina di errore login, richiamata nei seguenti casi:

- username e/o password non esistenti;
- uno o entrambi i campi son vuoti.

Tramite il pulsante "Indietro" si torna alla pagina precedente di benvenuto per riprovare con l'inserimento dei dati.

registrazione.jsp

Se non si è ancora registrati all'applicazione, dalla pagina di benvenuto si deve cliccare il link "form di registrazione" che reindirizza alla pagina di inserimento dati.
Viene richiesto il Nome, Cognome, la Password (da confermare) e la scelta di un'immagine tra quelle proposte.
La registrazione avrà esito positivo se vengono rispettati i vincoli imposti dalle successive classi di controllo.
Il pulsante "Home" in alto a sinistra consente di tornare alla pagina principale di benvenuto.

ControlloReg1.java

E' la prima servlet di controllo registrazione, a cui vengono passati i parametri inseriti dall'utente mediante il metodo getParameter.
L'operatore condizionale effettua i seguenti controlli, impostati con un "or" poiché se anche uno solo di questi è vero si viene reindirizzati ad una pagina di errore.

I casi son i seguenti:

- presenza anche di un solo un campo vuoto;
- password di conferma diversa da quella impostata;
- immagine non selezionata.

Se si verifica uno o più di questi eventi si viene reindirizzati alla pagina di errore registrazione, altrimenti si può procedere passando il controllo alla successiva classe impostando gli attributi dei dati utente da richiamare in quest'ultima.

ControlloReg2.java

E' la seconda servlet di controllo registrazione, in cui vengono importate anche le librerie di gestione della connessione al database.
Come prima cosa otteniamo gli attributi impostati dalla classe precedente e a loro volta li impostiamo per la pagina jsp successiva, dopodichè si istanzia il database per ottenere una connessione e la creazione dello statement.
Quindi si definisce una stringa contenente la query di controllo dell'esistenza Username nel database e si procede alla lettura dei vari campi tramite l'oggetto ResultSet.
Se tale interrogazione restituisce "false" significa che l'username non esiste ancora e quindi il controllo può essere passato alla successiva classe di inserimento, altrimenti se esiste si viene reindirizzati alla pagina di errore poiché, come detto, le username devono essere univoche.

InsertUtenti.java

In questa servlet si procede con l'inserimento dell'utente nel database, tramite gli attributi ottenuti dalle classi precedenti.
Si creano quindi nuovi oggetti delle classi Utenti e UtentiDB con l'esecuzione dei rispettivi metodi di impostazione degli attributi ed inserimento nel database.
Subito dopo l'inserimento dell'utente si crea una nuova istanza del database e della connessione per creare la tabella appuntamenti personalizzata dell'utente, impostando la propria username come suffisso alla denominazione "appuntamenti", quindi viene creata una tabella per ciascun utente che si registra.

In questa tabella si ceano i seguenti campi:

- id appuntamento auto-incrementante, che costituisce anche la chiave primaria;
- nome categoria appuntamento, massimo 20 caratteri;
- giorno della settimana di inizio appuntamento, scritto per esteso (max 9 caratteri);
- data di inizio appuntamento espresso nel formato data sql "aaaa-mm-gg";
- ora di inizio appuntamento espresso nel formato orario sql "hh:mm:ss";
- giorno della settimana fine appuntamento;
- data fine appuntamento;
- ora fine appuntamento;
- descrizione appuntamento, max 50 caratteri;
- note relative all'evento, max 100 caratteri.

Alla fine si chiude quindi la connessione e si passa alla classe di gestione "Gestione Login" vista in precedenza.

errorereg.jsp

E' la pagina di errore registrazione, richiamata nei casi in cui non si può procedere con la registrazione per non aver rispettato i vincoli imposti dalle relative classi di controllo viste in precdenza.
Tramite il pulsante "Indietro" si torna alla pagina precedente del form di registrazione.

areariservata.jsp

Dalla pagina di gestione di login si passa finalmente alla pagina principale dell'area riservata dell'utente, con il layout composto dalle seguenti sezioni:

- sezione superiore con il titolo dell'applicazione;
- sezione laterale sinistra per la gestione dell'account, con Nome, Cognome, Immagine, pulsante di modifica password e logout;
- sezione centrale contenente un calendario gregoriano perpetuo di consultazione ed in basso le modalità di visualizzazione degli appuntamenti in base a delle opzioni di ricerca;

- sezione laterale destra per la gestione delle categorie, la gestione degli appuntamenti e la ricerca disponibilità.

Nella sezione laterale sinistra contenente la gestione dell'account si procede prima di tutto ad aprire la nuova sessione utente mediante l'oggetto sessione della classe HttpSession, il cui id viene stampato a video all'interno di Eclipse (opzione che si può togliere ovviamente).
Per la visualizzazione del nome, cognome ed immagine sono state impostate omonimi variabili che prendono i valori passati come attributi dalle classi di controllo viste in precedenza (si fa un cast "String" perché il metodo getAttribute richiede come parametro un oggetto).
Riguardo l'immagine, dopo aver ottenuto la stinga corrispondente, per la visualizzazione dell'icona grafica si carica il corrispondente file .png della cartella "immagini" contenuta in "WebContent", mediante il metodo "getContextPath" della request.

Sempre nella medesima sezione, è presente l'opzione di modifica password che tramite l'apposito pulsante reindirizza alla pagina di modifica password; nel form html, oltre ad impostare il relativo input di tipo submit, sono stati impostati i parametri delle credenziali utente tramite il tipo "hidden", cioè nascosto, perché non devono essere visualizzati nel form di questa pagina.
Infine in questa sezione abbiamo il pulsante Logout che reindirizza all'apposita pagina in cui come vedremo viene terminata la sessione utente.

Nella sezione centrale è presente il calendario perpetuo il quale è stato realizzato mediante un listato javascript con una parte inserita all'interno del tag head del documento e l'altra all'interno del body.
Per utilizzarlo l'utente può cliccare i relativi pulsanti per selezionare una data specifica oppure scorrere i mesi sia in avanti che dietro, in modo da conoscere i giorni della settimana per qualunque data desiderata; si può ritornare alla data attuale tramite l'apposito pulsante.
Subito in basso al calendario è rappresentato il giorno ed ora attuali (aggiornabili con refresh della pagina), realizzati mediante l'oggetto Date di java ed il relativo metodo getTime.

Nella parte inferiore della sezione centrale sono presenti le modalità di visualizzazione appuntamenti che sono di 4 tipi: visualizzazione per giorno fissato, per mese fissato, per intervallo date, per categorie.
Per ciascuna di esse è stato creato quindi un form html contenente tutte le opzioni di selezione necessarie con un reindirizzamento alla jsp di interesse (che vedremo dopo) a cui vengono passati i parametri mediante il method "post" e l'action contenente il nome della jsp con estensione ".do".
Riguardo la visualizzazione per categorie si procede prima ad un'interrogazione del database per "sfogliare" i campi della relativa tabella in quanto come vedremo può essere modificata dall'utente che vuole aggiungere, modificare o cancellare determinate voci.
Per cui per ogni riga corrispondente ad un record della tabella, ottenuta con il metodo next dell'oggetto ResultSet, si "stampa" la relativa denominazione aggiungendola a ciascuna option del form html grazie all'espressione java contenuta al suo interno.

Nella sezione laterale destra sono contenute tutte le opzioni di gestione categorie, gestione appuntamenti e ricerca disponibilità.
Riguardo le categorie è possibile procedere all'inserimento, modifica e cancellazione accedendo alle rispettive pagine richiamate dai rispettivi pulsanti submit, in cui per ciascuno vengono passati tutti i parametri di tipo hidden contenenti i dati utente.
Discorso analogo per la gestione appuntamenti, con possibilità di inserimento, modifica e cancellazione degli stessi tramite gli appositi pulsanti a cui vengono passati i medesimi parametri; l'ultimo pulsante di ricerca disponibilità consente di entrare in una pagina in cui vengono mostrate le fasce orarie già occupate da appuntamenti inseriti in precedenza.

modpassword.jsp

Si procede prima di tutto ad impostare il medesimo layout della sezione laterale sinistra come per la pagina principale tramite i dati utente passati come parametri dalla stessa pagina, mentre il bottone "Modifica Password" diventa disabilitato in quanto siamo già in questa pagina; si aggiunge invece il pulsante indietro che consente di tornare alla pagina principale, se l'utente ad esempio non vuole più cambiare la password.

Nella sezione principale è contenuto il form che consente di cambiare la password: l'utente deve prima inserire la vecchia password e quindi digitare la nuova, confermandola a sua volta.
Quindi confermando i dati il controllo passa alla successiva servlet di controllo dati.

ModPassword.java

Come sempre per prima cosa si assegnano le variabili delle credenziali utente attraverso i parametri passati dalla classe precedente ed a loro volta si impostano come attributi per la jsp successiva.
Poi si controllano i dati provenienti dal form di modifica password, come sempre ottenuti tramite il metodo getParameter che ha come parametro il nome del riferimento a quel determinato input da tastiera.
Quindi tramite l'operatore condizionale si effettuano i seguenti controlli i quali, per poter passare alla fase successiva, devono avere tutti esito positivo, ovvero:

- che la vecchia password digitata corrisponda a quella già presente;
- che la password di conferma coincida con la nuova password impostata;
- che i campi della nuova password non siano vuoti.

In caso positivo si procede ad aprire una connessione al database per la modifica della password, quindi si effettuano le seguenti query (sempre istanziando un oggetto statement per ciascuna):

- una query di cancellazione del record corrispondente a quella determinata Username;
- una query di reinserimento del record con la nuova password lasciando inalterati gli altri dati.

Il controllo passa quindi alla successiva jsp che conferma il successo dell'operazione; nel caso invece che una delle condizioni specificate non sia vera si viene reindirizzati ad una pagina di errore.

passwsuccess.jsp

In questa pagina viene confermata l'avvenuta modifica della password; l'utente deve quindi uscire e logarsi nuovamente con la nuova password, tramite l'apposito pulsante.

errorepassw.jsp

In questa pagina invece è contenuto il messaggio di errore causato dall'esito negativo di uno o più controlli effettuati in precedenza, quindi tramite il pulsante indietro si può tornare alla pagina precedente di inserimento dati.

elencogiorno.jsp

Questa jsp visualizza l'elenco appuntamenti di un giorno prefissato.
Dopo aver importato i soliti parametri di gestione account utente posti sulla sezione sinistra, si importano anche quelli derivanti dall'input utente attraverso la scelta di giorno, mese ed anno effettuata dal form della pagina principale.

Quindi si apre una connessione al database e si esegue una query che, dalla tabella appuntamenti personale dell'utente, seleziona gli eventi la cui data di inizio equivale a quella impostata, ordinati per orario; la data è costruita con una stringa del tipo aaaa-mm-gg concatenando le stringhe di input provenienti dal form.
Tramite l'oggetto ResultSet si ottengono quindi tutti i campi degli eventi di quel giorno, inseriti all'interno di una tabella html contenente 2 righe e 5 colonne, con ciascuna cella che rappresenta un determinato campo (id, categoria, g.sett.inizio, data inizio, ora inizio, g. sett. fine, data fine, ora fine, descrizione, note).

Se non vi son tabelle significa ovviamente che non sono presenti appuntamenti per quel giorno.

elencomese.jsp

Questa pagina visualizza invece l'elenco appuntamenti di un mese prefissato.
Dopo aver importato i soliti parametri di gestione account utente posti sulla sezione sinistra, si importano anche quelli derivanti dall'input utente attraverso la scelta di mese ed anno effettuata dal form della pagina principale.

Quindi si apre una connessione al database e si esegue una query che, dalla tabella appuntamenti personale dell'utente, seleziona gli eventi il cui mese ed anno di inizio equivalgono a quella impostati, ordinati per data ed orario.
Tramite l'oggetto ResultSet si ottengono quindi tutti i campi degli eventi di quel mese, anche in questo caso inseriti all'interno di una tabella html contenente i campi come descritto in precedenza.

elencoint.jsp

In questa pagina si visualizza l'elenco appuntamenti contenuti in un intervallo di date.
Dopo la solita impostazione dei parametri di gestione account utente sulla sezione sinistra, si importano i parametri del form di input ottenuti tramite la scelta di giorno, mese ed anno di inizio intervallo nonché giorno, mese ed anno di fine intervallo.
Si creano quindi le stringhe formato sql "aaaa-mm-gg" tramite la concatenazione dei valori ottenuti, quindi si apre una connessione al database per l'esecuzione della relativa query; per la selezione dell'intervallo date si utilizza l'opzione "between".

Come prima si istanzia quindi l'oggetto ResultSet da cui si ottengono i campi degli eventi presenti in quell'intervallo date, inseriti all'interno della tabella html già descritta precedentemente.

elencocat.jsp

Questa jsp visualizza invece l'elenco appuntamenti appartenenti ad una determinata categoria.
Dopo aver importato i parametri di gestione account utente posti sulla sezione sinistra, si importa il parametro derivante dall'input utente attraverso la scelta della categoria effettuata dall'apposito form.
Quindi si apre la connessione al database e si esegue la query che seleziona gli eventi di tale categoria, ordinati per data ed orario di inizio; tramite l'oggetto ResultSet si ottengono quindi i campi degli eventi ordinati nella medesima tabella.

inscategoria.jsp

In questa pagina l'utente può inserire una nuova categoria.
Vengono visualizzate le categorie già presenti attualmente attraverso l'interrogazione della tabella "categorie" presente nel database, quindi si visualizzano i risultati attraverso l'oggetto ResultSet che "sfoglia" tutti i record inseriti.
Quindi si chiede all'utente se vuole inserire un'altra categoria, attraverso l'apposito form di input di inserimento testo, quindi la conferma tramite il relativo pulsante.
Il controllo passa quindi alla servlet InsCategoria.

InsCategoria.java

Dopo l'impostazione dei soliti parametri ed attributi delle credenziali utente, si ottiene il parametro della nuova categoria inserita per effettuare il controllo che il campo non sia vuoto.
Se è vuoto si viene reindirizzati ad una pagina di errore, altrimenti si procede con la connessione al database, quindi si esegue una query per verificare l'eventuale presenza di un record con la medesima denominazione.
In caso positivo, si viene reindirizzati alla pagina di errore, altrimenti si può procedere all'inserimento della nuova categoria, con successivo redirect alla pagina "catsuccess.do".

modcategoria.jsp

Questa pagina consente invece di modificare una categoria già esistente.
Si procede quindi ad un'interrogazione del database per "sfogliare" i campi della relativa tabella e, per ogni riga corrispondente ad determinato record, ottenuta con il metodo next dell'oggetto ResultSet, si "stampa" la relativa denominazione aggiungendola a ciascuna option del form html grazie all'espressione java contenuta al suo interno.
Poi viene creato il campo testo per l'input della nuova denominazione per la categoria selezionata, con successiva conferma mediante l'apposito pulsante.

ModCategoria.java

Questa è la servlet di controllo di modifica categoria.
Dopo l'impostazione dei parametri/attributi delle credenziali utente, si ottengono i parametri di selezione categoria ed inserimento della nuova denominazione.
Se il campo denominazione è vuoto, si viene reindirizzati ad una pagina di errore, altrimenti si procede con la connessione al database.
Quindi si effettua la query che ricerca l'eventuale esistenza di un campo con la stessa denominazione, in caso affermativo si viene reindirizzati alla pagina di errore, altrimenti si procede con l'inserimento.
Per far ciò si creano due oggetti statement che consentono rispettivamente di eliminare il record in esame e crearne un'altro con la nuova denominazione.
Infine si viene reindirizzati alla successiva pagina di conferma avvenuta operazione.

errorecat.jsp

In questa pagina è contenuto il messaggio di errore a cui si viene indirizzati nei casi illustrati nella classe di controllo inserimento (campo categoria vuoto oppure già presente nel database), nonché nei casi illustrati nella classe di modifica in cui il campo della nuova denominazione categoria è vuoto oppure equivale ad un campo già esistente nella rispettiva tabella del database.

cancategoria.jsp

Questa pagina consente di cancellare una determinata categoria.
Come prima si procede quindi ad un'interrogazione del database per "sfogliare" i campi della relativa tabella e, per ogni riga corrispondente ad determinato record, ottenuta con il metodo next dell'oggetto ResultSet, si "stampa" la relativa denominazione aggiungendola a ciascuna option del form html grazie all'espressione java contenuta al suo interno.
Il successivo pulsante di conferma rimanda quindi alla servlet che si occupa della cancellazione.

CanCategoria.java

In questa servlet si procede alla cancellazione della categoria selezionata.
Dopo l'impostazione dei parametri/attributi delle credenziali utente, si ottiene il parametro di selezione categoria, quindi si procede con la connessione al database e si esegue la query di cancellazione di quel determinato record.
Quindi si viene reindirizzati alla successiva pagina di conferma avvenuta operazione.

catsuccess.jsp

In questa pagina viene confermata la corretta modifica della sezione categorie (inserimento/modifica/cancellazione).

insappuntamento.jsp

In questa pagina l'utente può procedere all'inserimento di un nuovo appuntamento, a scelta tra due opzioni: inserimento di un singolo appuntamento oppure di uno ricorrente.
Sono stati creati quindi due form action che reindirizzano alle rispettive pagine di inserimento.

insappsingolo.jsp

In questa pagina l'utente può procedere con l'inserimento di un singolo appuntamento.
E' stato creato quindi un form per la selezione del giorno, mese, anno, ora di inizio appuntamento nonché giorno, mese, anno, ora di fine appuntamento; l'orario è impostabile alla mezz'ora.
Quindi si procede alla selezione della categoria tra quelle esistenti, elencate tramite interrogazione del database per la tabella "categorie", come visto in precedenza; poi è stato creato un campo testo per l'inserimento della descrizione ed infine una "textarea" più capiente (5 righe da 50 caratteri ciascuna) per l'inserimento del campo note.
Il pulsante di conferma rimanda quindi alla successiva servlet di controllo.

ContrAppSing.java

Dopo le solite impostazioni di parametri ed attributi utente, si procede all'inizializzazione di tutte le variabili attraverso l'input di ciascun parametro immesso dall'utente nella sezione di inserimento, quindi:

- giorno, mese ed anno di inizio;
- ora di inizio;
- giorno, mese ed anno di fine;
- ora di fine;
- categoria;
- descrizione;
- note.

Vengono quindi create variabili data per il corretto inserimento nel database tramite il formato aaaa-mm-gg, e variabili ora tramite il formato hh:mm:ss, mediante la concatenazione di stringhe delle variabili data e ora passate come parametri (i secondi vengono impostati a "00").

Prima di procedere oltre, mediante la classe SimpleDateFormat si ottengono i giorni della settimana corrispondenti alle date di inizio e fine impostate in precedenza, con i seguenti passaggi:

- si crea la stringa nel formato gg/mm/aaaa;
- si istanziano due oggetti della medesima classe rispettivamente per il formato data locale italiano e per l'ottenimento del giorno della settimana;
- si istanzia un oggetto della classe "Date" con un parsing della stringa data precedente;
- si ottiene il nome del giorno della settimana mediante il metodo format che ha come parametro l'oggetto data precedente.

Successivamente si impostano tutti gli attributi date/orari da passare alla successiva servlet, quindi le stringhe formato sql di date ed orari, stringhe di giorni della settimana, categoria, descrizione, note.

InsAppSing.java

Dopo le consuete operazioni di ottenimento parametri/impostazione attributi dei dati relativi all'account, si ottengono tutti gli attributi di inserimento passati dalla servlet precedente, quindi si procede con la connessione al database.
Si inserisce il record all'interno della tabella appuntamenti con tutti i campi necessari, quindi si viene indirizzati alla successiva pagina di conferma avvenuta operazione.

insappricorrente.jsp

Da questa pagina è possibile procedere con l'inserimento di un appuntamento ricorrente, scegliendo tra le due opzioni proposte: con cadenza settimanale oppure per giorno del mese.
I due pulsanti rimandano alle rispettive jsp di inserimento.

insappricsett.jsp

In questa pagina l'utente inserisce un appuntamento che ha cadenza settimanale, a partire da una data di inizio.
Viene creato quindi un form per:

- l'inserimento del giorno, mese ed anno di inizio ricorrenza (per il giorno della settimana si può sempre consultare il calendario perpetuo);
- l'orario di inizio e fine appuntamento;
- numero di settimane consecutive per le quali si deve fissare l'appuntamento;
- categoria;
- descrizione;
- note.

Il pulsante di conferma rimanda alla successiva servlet di controllo.

ContrAppGsett.java

Dopo aver ottenuto i soliti parametri di account, si ottengono tutti i parametri relativi alla data/orario di inizio appuntamento, ora fine, numero di settimane consecutive, categoria, descrizione, note.
Anche in questo caso si ottengono le stringhe data/orario ottimizzate per il formato mysql ed il giorno della settimana ottenuto con il procedimento illustrato in precedenza.
Si impostano quindi tutti i parametri da passare alla successiva servlet di inserimento nel database.

InsAppGsett.java

Dopo aver ottenuto tutti gli attributi passati dalla classe precedente, si crea una nuova variabile di tipo intero che ha come valore il parsing della variabile stringa che rappresenta il numero di settimane.
Ciò permette infatti di creare un ciclo for che esegue le operazioni di inserimento per il numero di settimane impostato.
Prima di eseguire il ciclo si istanzia un oggetto della classe Calendar a cui va impostata la data selezionata mediante il parametro passato dalla classe precedente, in modo da utilizzare il metodo "add" per effettuare una sottrazione di 7 giorni (vedremo fra poco perché).

Nel ciclo for si procede prima di tutto all'incremento di 7 giorni per la classe Calendar, ed è per questo motivo che viene effettuata la sottrazione precedente, in modo che il giorno di inizio ciclo coincida con quello selezionato da input utente; a partire dal 2° ciclo e per la durata impostata, questo incremento serve ovviamente ad aggiungere una settimana all'oggetto calendar che rappresenta la data di interesse.

All'interno del ciclo si procede quindi con i seguenti passi:

- ottenimento della stringa data mediante metodo toString, nel formato esteso (giorno sett. 3 caratteri – mese 3 caratteri – giorno – anno – ora:minuti:secondi – GMT+0100);
- ottenimento stringa giorno mediante una sottostringa di quella precedente, corrispondente ai caratteri compresi tra il n. 8 e 10;

- ottenimento stringa mese mediante una sottostringa di quella precedente, corrispondente ai caratteri compresi tra il n. 4 e 7;
- la variabile stringa mese viene trasformata in stringa numerica mediante operatori condizionali if per ciascun valore;
- ottenimento stringa anno mediante una sottostringa di quella precedente, corrispondente ai caratteri compresi tra il n. 24 e 28;
- costruzione stringa formato sql tramite concatenazione delle precedenti.

Si procede quindi alla connessione al database, quindi alla creazione dell'oggetto PreparedStatement per l'inserimento del record con tutti i dati necessari.
Alla fine del ciclo e dopo tutti gli inserimenti si viene quindi reindirizzati alla pagina di conferma avvenuta operazione.

insappricgiorno.jsp

In questa pagina l'utente inserisce un appuntamento che ha cadenza mensile per giorno prefissato, a partire da una data di inizio.
Viene creato quindi un form per:

- l'inserimento del giorno, mese ed anno di inizio ricorrenza;
- l'orario di inizio e fine appuntamento;
- numero di mesi consecutivi per cui si deve fissare l'appuntamento;
- categoria;
- descrizione;
- note.

Il pulsante di conferma rimanda alla successiva servlet di controllo.

ContrAppGmese.java

Dopo aver ottenuto i consueti parametri di account, si ottengono tutti i parametri relativi alla data/orario di inizio appuntamento, ora fine, numero di mesi consecutivi, categoria, descrizione, note.

Si ottengono quindi le stringhe data/orario ottimizzate per il formato mysql ed il giorno della settimana di inizio appuntamento ottenuto con l'analogo procedimento illustrato in precedenza.
Si impostano quindi tutti i parametri da passare alla successiva servlet di inserimento nel database.

InsAppGmese.java

Una volta ottenuti tutti gli attributi passati dalla classe precedente, anche qui si crea una nuova variabile di tipo intero che ha come valore il parsing della variabile stringa che rappresenta stavolta il numero di mesi.
Analogamente al caso della ricorrenza settimanale, ciò permette di creare un ciclo for che esegue le operazioni di inserimento per il numero di mesi impostato.
Anche qui prima di eseguire il ciclo si istanzia un oggetto della classe Calendar a cui va impostata la data selezionata mediante il parametro passato dalla classe precedente, in modo da utilizzare il metodo "add" per effettuare una sottrazione di 1 mese, per lo stesso motivo per il quale il successivo incremento all'interno del ciclo for faccia coincidere la data di inizio ciclo con quella impostata dall'utente.

Nel ciclo for si procede infatti inizialmente all'incremento di 1 mese per la classe Calendar; a partire dal 2° ciclo e per la durata impostata, questo incremento serve ovviamente ad aggiungere un mese all'oggetto calendar che rappresenta la data di interesse.
All'interno del ciclo si procede quindi con passi analoghi al caso precedente, ovvero:

- ottenimento della stringa data mediante metodo toString, nel formato esteso (giorno sett. 3 caratteri – mese 3 caratteri – giorno – anno – ora:minuti:secondi – GMT+0100);
- ottenimento stringa giorno mediante una sottostringa di quella precedente, corrispondente ai caratteri compresi tra il n. 8 e 10;
- ottenimento stringa mese mediante una sottostringa di quella precedente, corrispondente ai caratteri compresi tra il n. 4 e 7;
- la variabile stringa mese viene trasformata in stringa numerica mediante operatori condizionali if per ciascun valore;

- ottenimento stringa anno mediante una sottostringa di quella precedente, corrispondente ai caratteri compresi tra il n. 24 e 28;
- costruzione stringa formato sql tramite concatenazione delle precedenti;
- ottenimento del giorno della settimana mediante la classe SimpleDateFormat ed il parsing descritto in precedenza.

Si procede quindi alla connessione al database, quindi alla creazione dell'oggetto PreparedStatement per l'inserimento del record con tutti i dati necessari.
Alla fine del ciclo e dopo tutti gli inserimenti si viene quindi reindirizzati alla pagina di conferma avvenuta operazione.

modappuntamento.jsp

In questa pagina l'utente deve selezionare un determinato giorno per controllare gli appuntamenti già esistenti per quella data.
Viene quindi creato un form per l'inserimento di giorno, mese ed anno con parametri che poi vengono inviati alla successiva jsp dopo aver cliccato il pulsante di conferma.

modappuntamento2.jsp

Qui viene visualizzato l'elenco degli appuntamenti per il giorno selezionato.
Si ottengono quindi i parametri data passati dalla precedente pagina di input e si costruisce la consueta stringa sql per l'interrogazione del database, con la selezione di tutti i record appuntamenti relativi a quella determinata data di inizio, ordinati per orario.
Quindi si scorrono i risultati tramite l'oggetto ResultSet e si estrapolano i singoli campi, in modo da inserirli in una tabella html per elencare gli appuntamenti presenti allo stesso modo della visualizzazione per giorno vista in precedenza.
In blu, nella prima cella, è rappresentato l'id che costituisce la chiave primaria della tabella appuntamenti, pertanto ognuno di questi è identificato univocamente da tale numero.

Quindi viene creato un form di input contenente l'elenco di id appuntamenti relativo a quel giorno, in modo che l'utente possa selezionare quello da modificare; per costruire tale elenco si interroga un'altra volta il database estrapolando solo questo campo.
Nel form di scelta vengono quindi inserite le opzioni che sono output del campo id dei record restituiti dall'oggetto ResultSet.

Quindi dopo questa scelta e cliccando poi il pulsante di conferma si passa alla successiva pagina di re-inserimento.

modappuntamento3.jsp

In questa pagina si richiede di modificare l'appuntamento tramite reinserimento di tutti i campi previsti.
Analogamente alla pagina di inserimento è stato quindi creato un form per la selezione del giorno, mese, anno, ora di inizio appuntamento nonché giorno, mese, anno, ora di fine appuntamento, con orari impostabili alla mezz'ora; quindi poi si procede con il re-inserimento della categoria e del campo note.
Il pulsante di conferma rimanda quindi alla successiva servlet di controllo.

ModAppuntamento.java

Si ottengono i parametri account utente e si impostano come attributi per la pagina successiva, quindi si ottengono tutti gli attributi di inserimento passati dalla servlet precedente e si costruiscono le stringhe di date ed orari nel formato mysql tramite la concatenazione dei rispettivi parametri ottenuti in precedenza, quindi si ricavano i giorni della settimana di inizio e fine appuntamento.
Si procede poi alla creazione di una variabile intera "idsql" ottenuta come parsing della stringa id, quindi si effettua la connessione al database per eliminare il record appuntamento che ha quel determinato identificativo.
Successivamente si procede al re-inserimento del record con i dati di tutti i campi necessari e passati da input utente, quindi si viene indirizzati alla successiva pagina di conferma avvenuta operazione.

canappuntamento.jsp

Come in precedenza, in questa pagina l'utente deve selezionare un determinato giorno per controllare gli appuntamenti già esistenti per quella data.
Viene quindi creato un form per l'inserimento di giorno, mese ed anno con parametri che poi vengono inviati alla successiva jsp dopo aver cliccato il pulsante di conferma.

canappuntamento2.jsp

Anche qui viene visualizzato l'elenco degli appuntamenti per il giorno selezionato.
Si ottengono quindi i parametri data passati dalla precedente pagina di input e si costruisce la consueta stringa sql per l'interrogazione del database, con la selezione di tutti i record appuntamenti relativi a quella determinata data di inizio, ordinati per orario.
Quindi si scorrono i risultati tramite l'oggetto ResultSet e si estrapolano i singoli campi, in modo da inserirli in una tabella html per elencare gli appuntamenti presenti allo stesso modo della visualizzazione per giorno vista in precedenza.
In blu, nella prima cella, è rappresentato l'id che costituisce la chiave primaria della tabella appuntamenti, pertanto ognuno di questi è identificato univocamente da tale numero.

Quindi viene creato un form contenente l'elenco di id appuntamenti relativo a quel giorno, in modo che l'utente possa selezionare quello da cancellare; per costruire tale elenco si interroga un'altra volta il database estrapolando solo il campo id.
Nel form di scelta vengono quindi inserite le opzioni che sono output del campo id dei record restituiti dall'oggetto ResultSet.

Pertanto una volta effettuata la scelta e cliccando poi il pulsante di conferma si passa alla successiva servlet di cancellazione.

CanAppuntamento.java

Come sempre si ottengono i parametri account utente e si impostano come attributi per la pagina successiva.
Si procede quindi all'ottenimento del parametro id scelto dall'utente, poi si fa il parsing con un'altra variabile di tipo intera da sottoporre all'interrogazione del database.
Tramite la query di cancellazione si procede quindi ad eliminare il record appuntamento che ha quel determinato identificativo, prima di passare alla successiva pagina di conferma avvenuta operazione.

appsuccess.jsp

In questa pagina viene confermata la corretta modifica della sezione appuntamenti (inserimento/modifica/cancellazione).

erroreapp.jsp

In questa pagina è contenuto il messaggio di errore a cui si viene indirizzati nei casi di errori riguardanti la sezione appuntamenti.

ricercadisp.jsp

In questa pagina occorre selezionare un giorno per visualizzare le fasce orario già occupate da appuntamenti inseriti in precedenza.
Come di consueto viene quindi creato un form per l'inserimento di giorno, mese ed anno con parametri che poi vengono inviati alla successiva jsp dopo aver cliccato il pulsante di conferma.

orarioccupati.jsp

Si procede con l'ottenimento dei parametri giorno/mese/anno inseriti nel precedente form, quindi si costruisce la stringa sql mediante la concatenazione stringhe dei dati ottenuti.
Successivamente si apre una connessione al database, si istanzia l'oggetto della classe Statement per l'esecuzione della query che seleziona gli orari di inizio e fine di ciascun appuntamento presente in quel determinato giorno, ordinati per ora d'inizio.
Tramite il consueto oggetto della classe ResultSet si scorrono i record per ottenere le stringhe orari, quindi si estrapolano i singoli valori orario e minuti mediante nuove variabili che sono sotto-stringa delle precedenti (ora ottenibile con una sottostringa dei caratteri dal n. 0 al 2, minuti ottenibili con una sottostringa dei caratteri dal n. 3 al 5).
Si crea quindi una tabella html in cui vengono inserite le fasce orario già occupate.

Tramite il pulsante indietro si torna alla pagina principale.

Logout.java

Dopo aver cliccato il relativo pulsante di logout dalla pagina principale, si passa a questa servlet in cui si procede alla chiusura della sessione attuale.
Infine si viene reindirizzati alla relativa pagina di conferma logout.

logout.jsp

In questa pagina viene confermata la corretta chiusura della sessione, quindi si torna all'home page di benvenuto mediante l'apposito pulsante.