Hochschule für
Technik und Wirtschaft
Dresden
University of Applied Sciences

Entwicklerdokumentation

Case-Gruppe 04

Modul: Software Engineering II

Studiengang Informatik

Sommersemester 2014

Contents

1 Einleitung

Die vorliegende Dokumentation dient zukünftigen Entwicklern dazu sich in das bestehende System einarbeiten zu können und die Arbeit fortsetzen zu können.

2 Enwurf

2.1 Grobentwurf

2.1.1 Architektur

2.1.2 Paketdiagramm

3 Implementierungsentwurf

Unser Programm ist in insgesamt 4 Pakete gegliedert. Die beiden Pakete DozentBelegverwaltungUI und StudentBelegverwaltungUI enthalten sämtliche Klassen für die Benutzeroberflächen (Login, Bearbeitung, ...) des Dozenten- und des Studentenprogrammes. Sie verwenden die Klassen aus dem Typen-paket und greifen gemeinsam auf die Datenbank des Paketes DB_Services zu.
Im folgenden werden für jedes Paket die Klassen aus dem Grobentwurf mit ihren Methoden definiert und ihre Aufgaben beschrieben.

3.1 Paket Typen

3.1.1 Die Klasse Beleg

Die Klasse Beleg enthält einen Konstruktor zur Initialisierung der Attribute. Die für einen Beleg relevanten Eigenschaften sind eine Kennung, ein Passwort zur Erstanmeldung, das automatisch generierte Semester, einen Dozenten, ein Start- und Enddatum, sowie die minimale und maximale Gruppenanzahl.
Außerdem beinhaltet die Klasse noch eine Funktion **AddGruppe** mit welcher Studenten-Gruppen vom Typ Gruppe zu einem Beleg hinzugefügt werden können.

3.1.2 Die Klasse Thema

Ein Thema besitzt nur eine Nummer und einen Namen. Diese beiden Attribute werden durch den Konstruktor gesetzt.

3.1.3 Die Klasse Gruppe

Die Attribute der Klasse Gruppe sind eine Kennung, eine Liste mit den Studenten die in der Gruppe arbeiten, ein Passwort, ein von der Gruppe gewähltes Thema (Nummer),

sowie eine Belegkennung. Sie werden vom Konstruktor initialisiert, wobei einer der beiden Konstruktoren die Belegkennung setzt und der andere nicht.
Desweiteren enthält diese Klasse eine Funktion **AddStudent** mit welcher ein Student vom Typ Student hinzugefügt werden kann.

3.1.4 Die Klasse Rolle

Die Klasse Rolle enthält nur den Rollennamen als Attribut, welcher vom Konstruktor initialisiert wird.

3.1.5 Die Klasse Student

Die wichtigsten Informationen über einen Studenten sind der Nach- und Vorname, die s-Nummer, sowie die Email-Adresse und die Rolle. Diese Attribute sind in der Klasse Student enthalten und werden vom Konstruktor gesetzt.

3.2 Paket DozentBelegverwaltungUI

3.2.1 Die Klasse MainForm

Die MainForm-Klasse im Programm Dozent ist dafür da, um einen schnellen Überblick über alle laufenden Belege, dessen Gruppen und der darin enthaltenden Studenten zu gewährleisten. Dabei wird sehr eng mit der Datenbank gearbeitet, um immer aktuelle und richtige Daten anzuzeigen.

- **public MainForm()**
  Dieser Konstruktor initialisiert die Klasse. Es wird der Titel des Fenster gesetzt und die Startposition auf die Mitte des Bildschirms verlagert, so dass der Nutzer einen direkten Überblick über die Funktionen bekommt. Außerdem wird die Funktion UpdateBelege() aufgerufen, wodurch die aktuell laufenden Belege aus der Datenbank geladen und angezeigt werden. Desweiteren werden die Eventhandler für die Doppelklick-Funktionen der Listboxen für die Belege und Gruppen hinzugefügt, damit man diese intuitiv bearbeiten kann.

- **void mitgliederDataGridView_UserDeletingRow**
  (object sender, DataGridViewRowCancelEventArgs e)
  Diese Funktion wird aufgerufen, wenn der Nutzer eine Zeile (einen Studenten) aus einer Gruppe löschen möchte. Dies wird gemacht, um bei dem Nutzer nachzufragen, ob er sich über die Konsequenzen im Klaren ist und außerdem, um diese Änderung direkt in der Datenbank einzutragen. Ein Student mit der S-Nummer "na" als Platzhalter kann nicht gelöscht werden, damit die Mindest- und Maximalanzahl der Studenten in der Gruppe intakt bleibt.

- **private void UpdateBelege(object sender)**
  Die Funktion UpdateBelege zieht die aktuell laufenden Belege aus der Datenbank und zeigt diese direkt in der belegListBox an. Somit bekommt der Nutzer eine direkte Übersicht auf die Daten in der Datenbank.

- **private void belegListBox_SelectedIndexChanged (object sender, EventArgs e)**
  Dies ist ein Eventhandler, der aufgerufen wird, sobald ein neuer Beleg ausgewählt wurde. Er wird dafür genutzt, um die zu dem Beleg gehörigen Gruppen anzuzeigen. Es wird zuerst der ausgewählte Beleg gesucht. Da die Funktion allgemein aufgerufen werden soll, um die Listbox der Gruppen zu aktualisieren, wird zuerst geschaut, ob überhaupt ein Beleg existiert. Wenn keiner exisitiert, wird der Inhalt der gruppenListBox geleert, da es dann keine zugehörigen Gruppen gibt. Wenn ein Beleg exisitiert, der ausgewählt ist, dann wird eine Anfrage an die Datenbank geschickt, die die zugehörigen Gruppen zurückgibt. Diese werden direkt in die gruppenListBox eingetragen.

- **private void UpdateRollen(Beleg beleg)**
  Diese Funktion zieht die verfügbaren Rollen zu dem ausgewählten Beleg aus der Datenbank. Sie ist dafür notwendig, damit man den Studenten einer Gruppe eine neue Rolle zuweisen kann. Der Beleg als Parameter wird benötigt, um die Belegkennung in der Datenbankabfrage zu verwenden.

- **private void belegListBox_DoubleClicked(object sender, EventArgs e)**
  Dies ist der oben angesprochene Eventhandler für das Doppel-Klick-Event in der belegListBox. Es wird eine neue Instanz der Klasse BelegBearbeiten erstellt, wobei direkt der Delegate-Handler festgelegt wird. Dieser ist notwendig, um Änderungen nach beenden dieses Dialoges in den ListBoxen anzuzeigen. Der zweite Parameter in dem Konstruktor ist ein Boolean-Wert, der angibt, ob es sich um einen neu angelegten Beleg handelt. Da hier das Doppel-Klick-Event abgefangen wird, handelt es sich nicht um einen neuen Beleg, sondern um einen existierenden, der bearbeitet werden soll. Danach wird der Dialog mit der Methode show() dem Nutzer angezeigt.

- **private void gruppenListBox_DoubleClicked(object sender, EventArgs e)**
  Dieser Eventhandler behandelt das Doppel-Klick-Event für die gruppenListBox. Wie bei den Belegen auch gibt es hier einen Boolean-Wert, der der Instanz sagt, ob es sich um eine neue oder eine existierende Gruppe handelt. Auch hier handelt es sich um eine existierende Gruppe, weshalb der Wert false übergeben wird.

- **private void gruppenListBox_SelectedIndexChanged (object sender, EventArgs e)**
  Dieser Eventhandler wird aufgerufen, sobald der Nutzer eine neue Gruppe in der

gruppenListBox auswählt. An dieser Stelle müssen jetzt die Studenten dieser Gruppe angezeigt werden. Dafür wird zuerst die ausgewählte Gruppen rausgesucht und anhand dieser und der Gruppenkennung (Case-Kennung) eine Datenbankabfrage nach den Studenten gestartet. Mit dem Ergebnis der Abfrage wird das mitglieder-DataGridView gefüllt, sodass der Nutzer alle Studenten dieser Gruppe einsehen kann. Bei dem Füllen werden außerdem Platzhalter für Studenten angezeigt, die repräsentieren, wieviele Studenten noch bis zur festgelegten Minimalanzahl bzw bis zur festgelegten Maximalanzahl fehlen. Noch auszufüllende Studenten sind dabei Gelb markiert.

- **protected override void OnFormClosing(FormClosingEventArgs e)**
  Wie das Stichwort override verrät, wird an dieser Stelle die Closing-Methode der Windows-Form überschrieben. Dies ist notwendig, weil es sein kann, dass diese Form nur von einer anderen präsentiert wird. Wenn das Fenster geschlossen wird, soll auf jeden Fall auch der Prozess beendet werden.

- **private void belegAnlegenButton_Click(object sender, EventArgs e)**
  Mit dieser Funktion kann man einen neuen Beleg anlegen. Sie wird geworfen, wenn der entsprechende Button im Interface angeklickt wird. Dafür wird ebenfalls die Klasse BelegBearbeiten verwendet, allerdings wird diesmal im Konstruktor mit dem Boolean "true" mitgeteilt, dass es sich um eine neue Klasse handelt. Auch hier wird ein Delegate-Handler festgelegt, um die Listboxen nach dem Anlegen zu aktualisieren.

- **private void gruppeAnlegenButton_Click(object sender, EventArgs e)**
  Dies ist der äquivalente Button zu der BelegAnlegen-Funktion. Es wird eine neue Gruppe angelegt, indem eine Instanz der Klasse GruppeBearbeiten angelegt wird. Auch hier wird ein Delegate gesetzt, um nach dem Anlegen die gruppenListBox zu aktualisieren.

- **List<string>getFreieCases(Gruppe gruppe)**
  Diese Funktion repräsentiert eine Datenbankabfrage, mit der freie Case-Kennungen zu einem Beleg aus der Datenbank abgerufen werden können. Diese Funktion wird in der gruppeAnlegenButton_Click-Funktion verwende, um zu schauen, ob es überhaupt möglich ist, eine weitere Gruppe zu diesem Beleg anzulegen, oder ob nicht alle Case-Kennungen schon vergeben sind. Es wird eine Gruppe und kein Beleg übergeben, weil diese Funktion wie gesagt bei dem Anlegen einer Gruppe aufgerufen wird. Dabei wird vorher eine Platzhalter-Gruppe angelegt, die dem Konstruktor von GruppeBearbeiten übergeben wird. Diese Gruppe kann auch direkt als Anhaltspunkt für diese Funktion gewählt werden.

- **private void dataGridViewFreigebenButton_Click (object sender, EventArgs e)**
  Mit dieser Funktion wird das DataGridView, das die Studenten einer Gruppe repräsentiert für Bearbeitungen freigegeben. Standardmäßig ist dieses gesperrt, damit nicht versehentlich Änderungen gemacht werden.

- **private void saveDataGridViewButton_Click(object sender, EventArgs e)**
  Diese Funktion ruft die Funktion SaveMitglieder() auf und sperrt danach das DataGridView wieder, um den Nutzer vor ungewollten Änderungen zu schützen.

- **private bool SaveMitglieder()**
  Diese Funktion speichert alle Änderungen von Studenten der ausgewählten Gruppe. Dabei wird für jeden Studenten unterschieden, ob es sich um einen neuen oder einen geänderten Studenten handelt. Dies wird davon abhängig gemacht, ob die Zelle, die die S-Nummer beinhaltet, gesperrt ist oder nicht. Wenn sie gesperrt ist, handelt es sich um einen schon existierenden Studenten, der in der Datenbank mit den neuen Werten aktualisiert wird. Ansonsten wird der Student eingefügt. Teil dieses Prozesses ist auch die Überprüfung auf eine korrekte und neue S-Nummer sowie auf einen korrekte E-Mail-Adresse.

- **private bool checkMail(string mail)**
  Dies ist eine einfache und kurze Funktion, die anhand von Regular-Expressions eine eingegebene E-Mail-Adresse auf korrekte Validierung untersucht.

- **private bool checkSNummer(string sNummer)**
  Diese Funktion überprüft, ob es sich bei der eingegebenen S-Nummer wirklich um eine korrekte S-Nummer handelt.

- **private void updateStudent(Student student)**
  Die Funktion bekommt einen Studenten als Parameter, um dessen Werte in der Datenbank zu aktualisieren. Da nicht unterschieden werden kann, ob bei dem Studenten überhaupt etwas geändert wurde, wird diese Funktion beim Speichern für jeden existierenden Studenten durchgeführt.

- **private void insertStudent(Student student, Gruppe gruppe)**
  Diese Funktion trägt einen neuen Studenten in die Datenbank ein. Da an dieser Stelle nicht nur die Tabelle Student wichtig ist, sondern auch die Zuordnungs-Tabelle zwischen Student und Gruppe, wird hier auch die Gruppe mit übergeben.

- **private void cancelDataGridViewButton_Click (object sender, EventArgs e)**
  Mit dieser Funktion kann das Bearbeiten der Studenten einer Gruppe wieder rückgängig gemacht werden. Dabei werden die Elemente zum Bearbeiten wieder gesperrt und

außerdem werden die Studenten der aktuell ausgewählten Gruppe einfach wieder neu aus der Datenbank geladen.

- **private void themenVerwaltenButton_Click(object sender, EventArgs e)**
  Mit dieser Funktion, durch den Button "Themen verwalten" ausgelöst, wird eine neue Instanz der Klasse ThemenVerwalten erstellt und diese angezeigt.

- **private void rolleTextBox_Click(object sender, EventArgs e)**
  Mit dieser Funktion wird eine Instanz der Klasse RolleVerwalten geladen und angezeigt, damit man den Pool an Rollen neu aufstellen kann.

- **private void buttonArchivieren_Click(object sender, EventArgs e)**
  Diese Funktion erlaubt das Archivieren von allen aktuellen Belegen. Dabei wird eine Instanz der Klasse PdfArchivierung erstellt. Die Klasse hat ebenfalls einen Delegate-Handler, damit die gelöschten Datensätze nach der Archivierung nicht mehr in dem Programm sichtbar sind.

- **private void button1_Click(object sender, EventArgs e)**
  Diese etwas unglücklich benannte Funktion erstellt eine neue Instanz der Klasse kontaktForm, wodurch der Nutzer bestimmte Gruppen oder Studenten der einzelnen Belege per Mail kontaktieren kann.

- **private void belegLoeschenButton_Click(object sender, EventArgs e)**
  Mit dieser Funktion kann ein ausgewählter Beleg gelöscht werden. Dabei wird zuerst nachgefragt, ob der Nutzer sich der Konsequenzen bewusst ist und ob er sich sicher ist. Danach wird geschaut, ob der zu löschende Beleg noch Gruppen enthält. Wenn dem so ist, wird der Nutzer darüber informiert und ein zweites Mal gefrag, ob er sich dessen sicher ist. Wenn die Funktion bis hier noch nicht beendet wurde, wird der Beleg inklusive der existierenden Gruppen und dessen Studenten aus der Datenbank gelöscht. Danach werden automatisch die Listboxen und das Datagridview aktualisiert, sodass der Nutzer den gelöschten Beleg nicht mehr im Programm sieht.

- **private void gruppeLoeschenButton_Click(object sender, EventArgs e)**
  Die Funktion löscht die ausgewählte Gruppe. Äquivalent zum Beleg wird auch hier Nachgefragt und im Anschluss geschaut, ob die Gruppe Studenten beinhaltet. Wenn dem so ist, wird auch hier darüber informiert und ein zweites Mal nachgefragt. Erst danach werden die Studenten und die Gruppe selbst aus allen betreffenden Tabellen in der Datenbank gelöscht. Danach werden die Listboxen wieder aktualisiert, um keine alten, falschen Daten anzuzeigen.