Technische Dokumentation

ScanScore

Autoren: Isabell Schwertle, Timo Kehrer

Stand: 04.08.2006

Druck: 04.08.2006 16:28:00

Version: 0.9.1

Datei: ScanScore_Tech_Doku.doc

Historie:

Version	Datum	Bearbeiter	Inhalt
0.1.0	23.05.2005	Timo Kehrer	Initiale Version
0.2.0	29.05.2005	Isabell Schwertle	Architektur-Übersicht
0.3.0	01.07.2005	Timo Kehrer	Datenmodell
0.4.0	28.07.2006	Isabell Schwertle	Plugin-Konzept und Plugin-Module; Übersicht
0.5.0	29.07.2006	Timo Kehrer	Komponenten-Konzept, erste Version
0.6.0	01.08.2006	Timo Kehrer	Persistenz – Implementation, erste Version
0.7.0	01.08.2006	Timo Kehrer	Überarbeitung Plugin-Konzept.
0.8.0	02.08.2006	Timo Kehrer	Entwicklungsumgebung
0.9.0	03.08.2006	Timo Kehrer	Build-Prozess
0.9.1	04.08.2006	Timo Kehrer	Überarbeitung Build-Prozess

Inhalt

Abbildungsverzeichnis

1 Lesehinweise

Im Folgenden soll ein kurzer Überblick über den Inhalt und Aufbau dieses Dokuments gegeben werden.

TODO: Diesen Hinweis bei Fertigstellung wieder entfernen

Es ist zu beachten, dass dieses Dokument in der derzeitigen Version keinesfalls den Anspruch auf Vollständigkeit erhebt.

1.1 Potentielle Leser des Dokuments

Als potentielle Leser kommen in erster Linie Entwickler in Frage, welche die bestehende Funktionalität der Applikation erweitern, Performance-Optimierungen, insbesondere bei der Verarbeitung der Scandaten vornehmen, oder bestehende Fehler im System beheben wollen. Diese Leser seien auf Abschnitt REF verwiesen, um sich einen Überblick über den Aufbau des Dokuments zu verschaffen, um so schnell zu den für sie relevanten Informationen navigieren zu können.

1.2 Aufbau des Dokuments

Im Abschnitt REF wird ein skizzenhafter Überblick über die Architektur von ScanScore geliefert. Sollte ein tiefer Eingriff in das System erfolgen, sollte dieser Abschnitt keinesfalls überlesen werden.

Die Abschnitte REF geben eine detaillierte Einführung und Beschreibung in und über das der Anwendung zu Grunde liegende Datenmodell. Für das Verständnis der Anwendung und der dahinter stehenden Konzepte erscheint ein Einstieg über das Datenmodell und die daraus resultierende Klassen und Schnittstellen zur Implementierung der Entitäten am sinnvollsten zu sein, weshalb sich diesen Themengebieten schon zu einer frühen Phase des Dokuments gewidmet wird.

Entwicklern sei für zukünftige Weiterentwicklungen hinsichtlich der Anbindung der Entitäten an die Datenbank Abschnitt REF ans Herz gelegt.

Abschnitt REF erläutert das implementierte Plugin-Konzept und stellt die notwendigen Schritte dar, um neue Plugin-Module für die Anwendung zu entwickeln.

In den Abschnitten REF wird teilweise sehr ausführlich auf die technische Umsetzung bereits bestehender Plugin-Module eingegangen.

Sehr lesenswert im Hinblick auf neue Entwicklungen und Erweiterungen der Funktionalität sind die Abschnitte REF und REF. Über das Konzept der Localization wird auf komfortable Art und Weise die Mehrsprachigkeit der Anwendung unterstützt. In die querschnittlichen Komponenten wurde möglichst viel Funktionalität extrahiert und generalisiert, um so ein hohes Maß an Wiederverwendung von bereits bestehendem Code und Komponenten zu gewährleisten. Sollten während der Entwicklung weitere generische Ansatzpunkte entdeckt werden, so sollten diese, entsprechend dokumentiert, zu dieser Klassenbibliothek hinzugefügt werden.

2 Entwicklungsumgebung

2.1 Eclipse

- Als Entwicklungsumgebung wird Eclipse in der Version 3.0.1 verwendet.
- Um alle notwendigen Plugins installiert zu haben empfiehlt es sich, die für ScanScore bereit gestellte Eclipse Version zu installieren.
- Eclipse ist dabei mit folgenden Parametern zu starten:

  ```
  C:\Programme\Java\eclipse-
  3.0.1_for_ScanScore\eclipse.exe  -vm
  "C:\Programme\Java\j2sdk1.4.2_07\bin\javaw.exe" -
  showlocation -vmargs -Xmx512M
  ```

  Zu beachten sind dabei die in Abschnitt 2.2 aufgeführten Verzeichnis-Pfade für das J2SE-Environment.

Installierte Eclipse-Plugins:

Notwendig für Build-Prozess und in für ScanScore bereitgestellter Eclipse-Version bereits enthalten:

com.hdm.scanscore.tools.getclasspath_1.0.0

TODO: Wäre mal nicht schlecht hier ne Liste zu haben. Neu einträge sind immer herzlich willkommen

www.classycleplugin.graf-tec.ch/

2.2 J2SE-Environment

- Unterstützt werden derzeit die Java-Versionen 1.4.2_07 und 1.5.0_02. Die entsprechenden J2SE-Environments sind unter folgenden Verzeichnissen zu installieren.

- C:\Programme\Java\j2sdk1.4.2_07

- C:\Programme\Java\jdk1.5.0_02

3 Datenmodell

Die folgenden Abschnitte sollen einen Überblick über das der Anwendung zu Grunde liegende Datenmodell liefern. Eine Einarbeitung in das Datenmodell fördert das Verständnis für das übergeordnete Konzept des Anwendungsdesigns.

Gleichermaßen stellt es auch die Grundlage für die Erweiterung der Anwendung um neue Feautures[1] dar.

3.1 ER-Diagramme

Nachfolgende ER-Diagramme liefern einen ersten Überblick über das Datenmodell. Eine detailliertere Beschreibung der einzelnen Tabellen und ihrer Attribute folgt in weiteren Abschnitten. Das Modell lässt sich in drei Bereiche unterteilen:

- **Stammdaten:**

  Pflege von Stammdaten verschiedenster Natur (Dozenten, Vorlesungen, etc.)

- **Evaluierungsbogen:**

  Strukturierung von Evaluierungsbögen

- **Auswertungsdaten:**

  Anhand von ausgefüllten Evaluierungsbögen erfasste Daten

Die Bereiche Stammdaten und Evaluierungsbogen lassen sich dabei isoliert betrachten, und können getrennt von den restlichen Bereichen gepflegt werden. Die Auswertungsdaten stehen jedoch immer in einer Beziehung zu den bestehenden Stammdaten und einer Struktur eines Evaluierungsbogens.

3.1.1 Evaluierungsbogen

Das in Abbildung 1 dargestellte ER-Diagramm zeigt die Strukturierungsmöglichkeiten eines Evaluierungsbogens.

[1] Bspw. statistische Auswertungsmöglichkeiten

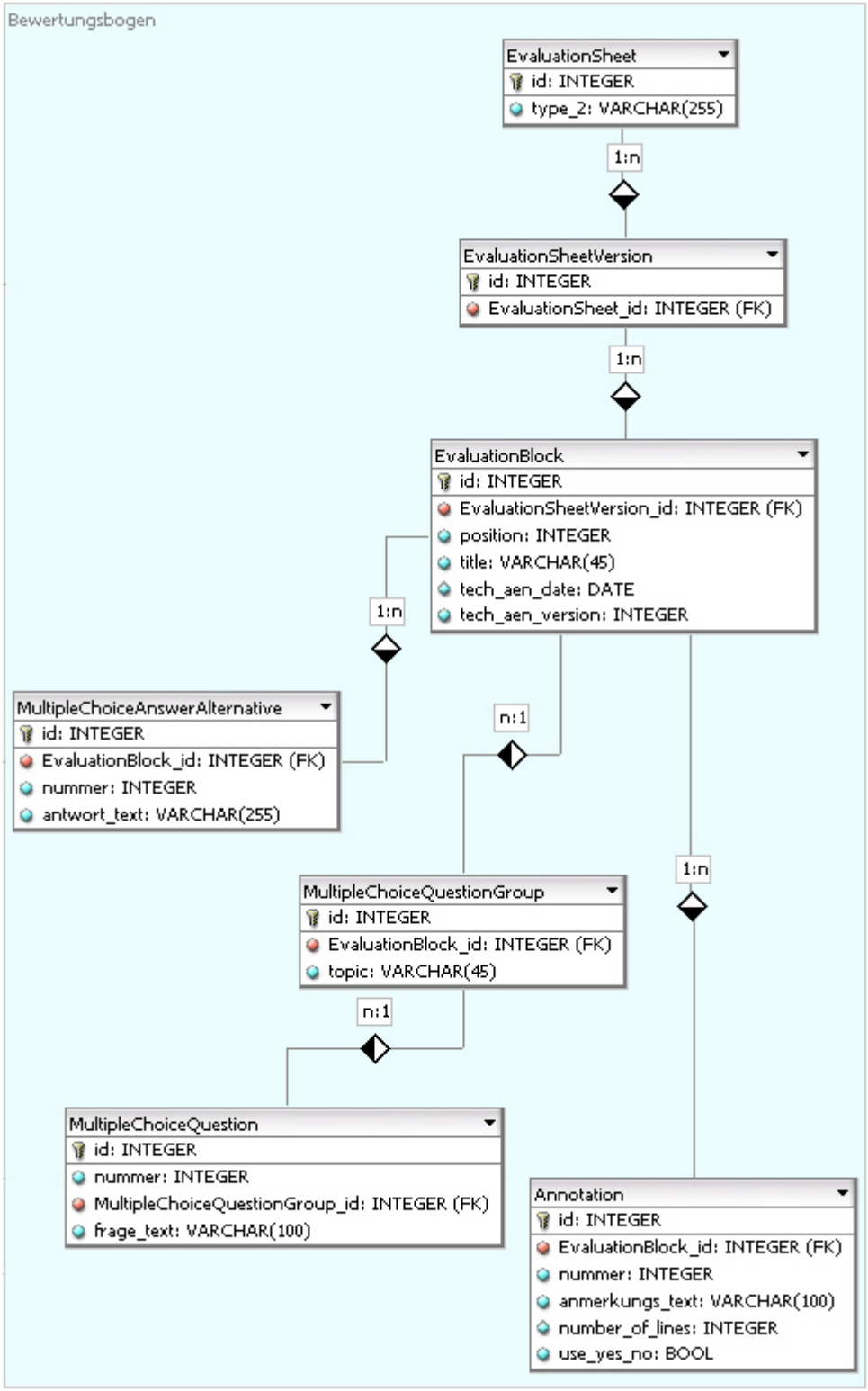


Abbildung 1 ER-Diagramm - Evaluierungsbogen

3.1.2 Stammdaten

Das in Abbildung 2 dargestellte ER-Diagramm zeigt die für die Stammdatenpflege relevanten Tabellen.

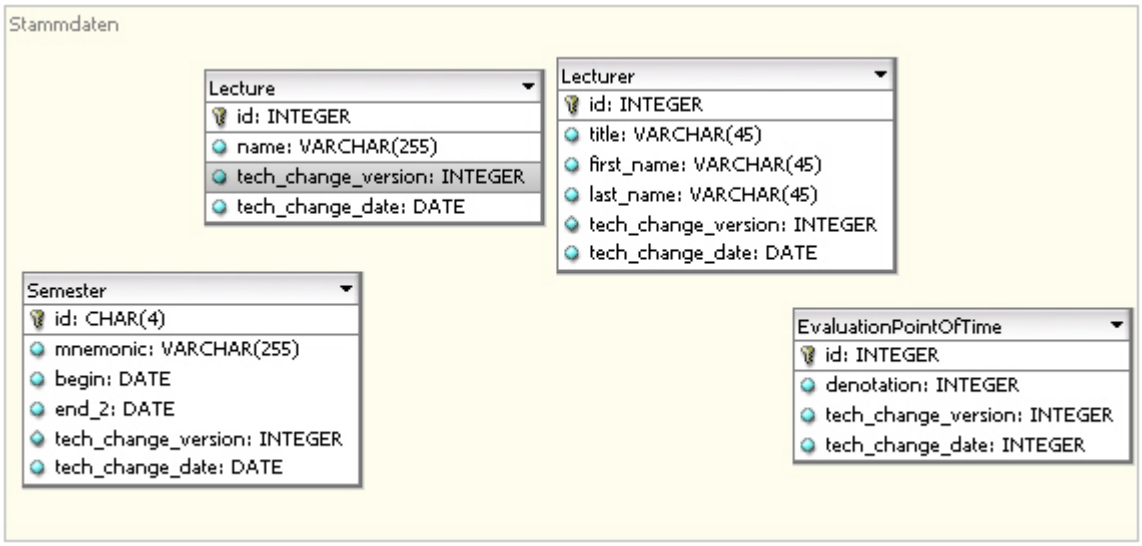


Abbildung 2 ER-Diagramm - Stammdaten

3.1.3 Auswertungsdaten

Abbildung 3 zeigt die für die Erfassung von Auswertungsdaten relevanten Tabellen und deren Integration und Beziehungen zu den Tabellen der Stammdatenhaltung und Evaluierungsbögen.

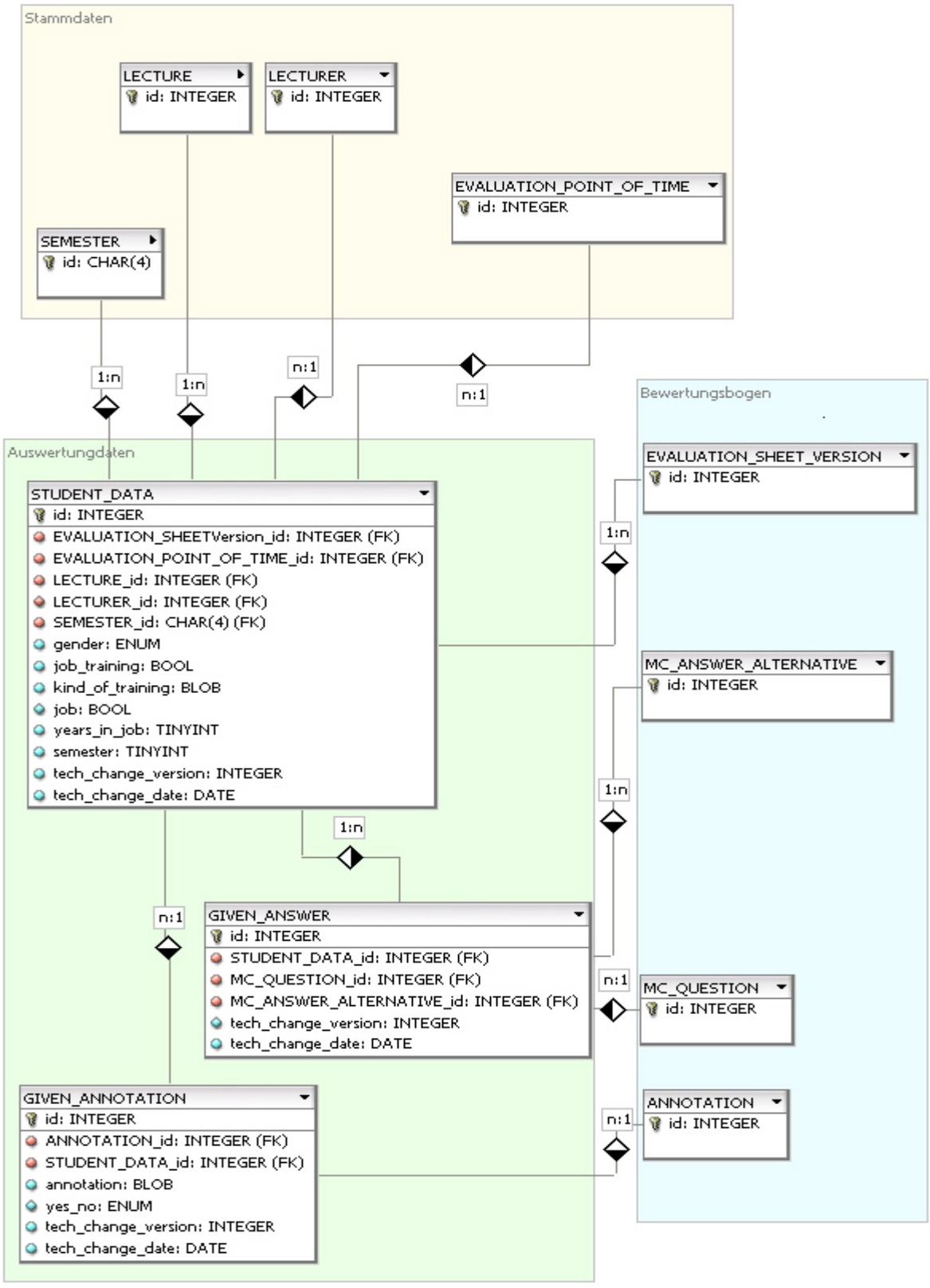


Abbildung 3 ER-Diagramm - Auswertungdaten

3.1.4 Tabelle EVALUATION_SHEET

Entität	EvaluationSheet	
Beschreibung	Haupt-Entität eines Evaluierungsbogens	
Attribute		
Name	**Typ**	**Beschreibung**
id	INTEGER	Technischer Primärschlüssel
type	TINYINT	Bogentyp {Lehrveranstaltung, Studiengang, Studiengangskonzept}
tech_change_version	INTEGER	Versionsnummer der letzten Änderung
tech_change_date	DATE	Datum der letzen Änderung

3.1.5 Tabelle EVALUATION_SHEET_VERSION

Entität	EvaluationSheetVersion	
Beschreibung	Finale Version eines Evaluierungsbogens	
Attribute		
Name	**Typ**	**Beschreibung**
id	INTEGER	Technischer Primärschlüssel
evaluation_sheet_id	INTEGER	Fremdschlüssel
tech_change_version	INTEGER	Versionsnummer der letzten Änderung

tech_change_date	DATE	Datum der letzen Änderung

3.1.6 Tabelle EVALUATION_BLOCK

Entität	EvaluationBlock	
Beschreibung	Strukturierungselement in Evaluierungsbögen	
Attribute		
Name	**Typ**	**Beschreibung**
id	INTEGER	Technischer Primärschlüssel
evaluation_sheet_version_id	INTEGER	Fremdschlüssel
position	INTEGER	Positionierung innerhalb von Evaluierungsblöcken der gleichen Fremdschlüssel-Beziehung
title	VARCHAR(45)	Titel des Evaluierungsblocks
tech_change_version	INTEGER	Versionsnummer der letzten Änderung
tech_change_date	DATE	Datum der letzen Änderung

3.1.7 Tabelle MC_ANSWER_ALTERNATIVE

Entität	MultipleChoiceAnswerAlternative	
Beschreibung	Strukturierungselement in Evaluierungsbögen	
Attribute		
Name	**Typ**	**Beschreibung**

id	INTEGER	Technischer Primärschlüssel
evaluation_block_id	INTEGER	Fremdschlüssel
position	INTEGER	Positionierung innerhalb von Evaluierungsblöcken der gleichen Fremdschlüssel-Beziehung
answer_text	VARCHAR(45)	Mögliche Antwortmöglichkeit
tech_change_version	INTEGER	Versionsnummer der letzten Änderung
tech_change_date	DATE	Datum der letzen Änderung

3.1.8 Tabelle MC_QUESTION_GROUP

Entität	MultipleChoiceQuestionGroup	
Beschreibung	Strukturierungselement in Evaluierungsbögen	
Attribute		
Name	**Typ**	**Beschreibung**
id	INTEGER	Technischer Primärschlüssel
evaluation_block_id	INTEGER	Fremdschlüssel
position	INTEGER	Positionierung innerhalb von Evaluierungsblöcken der gleichen Fremdschlüssel-Beziehung
topic	VARCHAR(45)	Themengebiet der Fragegruppe
tech_change_version	INTEGER	Versionsnummer der letzten Änderung
tech_change_date	DATE	Datum der letzen Änderung

3.1.9 Tabelle MC_QUESTION

Entität	MultipleChoiceQuestion	
Beschreibung	Strukturierungselement in Evaluierungsbögen	
Attribute		
Name	**Typ**	**Beschreibung**
id	INTEGER	Technischer Primärschlüssel
mc_question_group_id	INTEGER	Fremdschlüssel
position	INTEGER	Positionierung innerhalb von Evaluierungsblöcken der gleichen Fremdschlüssel-Beziehung
question_text	VARCHAR(45)	Formulierte Frage
tech_change_version	INTEGER	Versionsnummer der letzten Änderung
tech_change_date	DATE	Datum der letzen Änderung

3.1.10 Tabelle ANNOTATION

Entität	Annotation	
Beschreibung	Strukturierungselement in Evaluierungsbögen	
Attribute		
Name	**Typ**	**Beschreibung**
id	INTEGER	Technischer Primärschlüssel
evaluation_block_id	INTEGER	Fremdschlüssel

position	INTEGER	Positionierung innerhalb von Anmerkungen der gleichen Fremdschlüssel-Beziehung
annotation_text	VARCHAR(100)	Anmerkungs-Text
number_of_lines	INTEGER	Anzahl der Linien, welche zum Ausfüllen der Anmerkung zur Verfügung stehen
use_yes_no	BOOL	Ja-Nein-Anmerkung?
tech_change_version	INTEGER	Versionsnummer der letzten Änderung
tech_change_date	DATE	Datum der letzen Änderung

3.1.11 Tabelle LECTURE

Entität	Lecture	
Beschreibung	Angebotene Vorlesungen	
Attribute		
Name	**Typ**	**Beschreibung**
id	INTEGER	Technischer Primärschlüssel
name	VARCHAR(255)	Vorlesungs-Name
tech_change_version	INTEGER	Versionsnummer der letzten Änderung
tech_change_date	DATE	Datum der letzen Änderung

3.1.12 Tabelle SEMESTER

Entität	Semester	
Beschreibung	Angelegte Semester	
Attribute		
Name	**Typ**	**Beschreibung**
id	INTEGER	Technischer Primärschlüssel
mnemonic	VARCHAR(255)	Semester-Kürzel
begin	DATE	Semester-Beginn
end	DATE	Semester-Ende
tech_change_version	INTEGER	Versionsnummer der letzten Änderung
tech_change_date	DATE	Datum der letzen Änderung

3.1.13 Tabelle LECTURER

Entität	Lecturer	
Beschreibung	Angelegte Dozenten	
Attribute		
Name	**Typ**	**Beschreibung**
id	INTEGER	Technischer Primärschlüssel
title	VARCHAR(45)	Titel (z.B. „Prof. Dr.“)
first_name	VARCHAR(45)	Vorname

last_name	VARCHAR(45)	Nachname
tech_change_version	INTEGER	Versionsnummer der letzten Änderung
tech_change_date	DATE	Datum der letzen Änderung

3.1.14 Tabelle EVALUATION_POINT_OF_TIME

Entität	EvaluationPointOfTime
Beschreibung	Mögliche Zeitpunkte für die Beurteilung des Studiengangs oder des Studiengangskonzeptes

Attribute

Name	**Typ**	**Beschreibung**
id	INTEGER	Technischer Primärschlüssel
denotation	VARCHAR(255)	Beschreibung des Beurteilungszeitpunktes (z.B. „Nach dem Praxissemester“)
tech_change_version	INTEGER	Versionsnummer der letzten Änderung
tech_change_date	DATE	Datum der letzen Änderung

4 Architektur

4.1 Überblick

Abbildung TODO zeigt einen ersten Überblick über die Architektur der Anwendung.

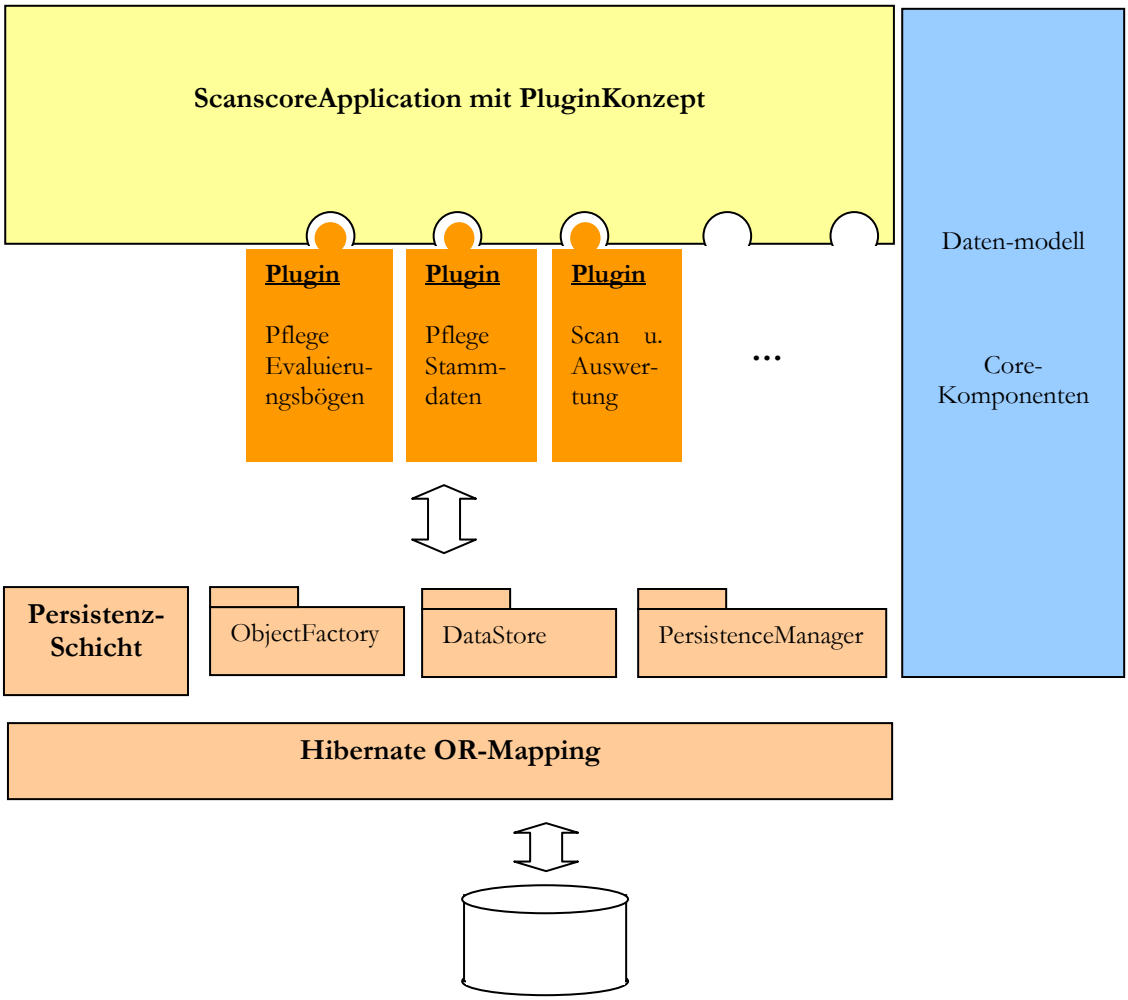


Abbildung 4 Architektur-Übersicht

Eine Kommunikation mit der Datenbank findet ausschließlich über eine Persistenz-Schicht mit definierter Persistenz-Schnittstelle (s. Kapitel TODO) statt. Die Persistenz-Schicht übernimmt dabei das objektrelationale Mapping der jeweiligen Entitäten und kapselt die Anwendung von Spezifika verschiedener Datenbank-Implementierungen und -Hersteller ab. Derzeit wird dabei das Mapping-Tool Hibernate (www.hibernate.org) verwendet.

Grundsätzlich kann aber unter bei korrekter Implementation der nach außen definierten Persistenz-Schnittstelle jede beliebige Implementierung und die Verwendung verschiedenster Bibliotheken eingesetzt werden.

Zusätzlich werden der Anwendung einige zentralen Core-Komponenten wir etwa Utility-Klassen, ein Logging-Interface oder die Unterstützung von Mehrsprachigkeit zur Verfügung gestellt.

Für die grafische Oberfläche von Scanscore wurde ein Plugin-Konzept realisiert, mit dem der Applikation jederzeit einfach neue GUI-Module hinzugefügt werden können. Das Plugin-Konzept sowie bereits realisierte Module werden in den Kapiteln TODO und TODO beschrieben.

4.2 Komponentenmodell

Im folgenden wird die im vorherigen Abschnitt anhand eines Überblicks vorgestellte Architektur der Anwendung näher erläutert. Wie bereits angedeutet, setzt sich die Anwendung aus mehreren Komponenten zusammen. Komponenten definieren ihre Schnittstellen, die sie anderen Komponenten zur Verfügung stellen, die letztendliche Realisierung soll nach außen hin verborgen bleiben. Da die verwendete Programmiersprache Java einen solchen Mechanismus nur unzureichend unterstützt, werden nach außen freigegebene Klassen und Schnittstellen einer Komponente per Konvention in speziellen Packages bereitgestellt, welche mit `export` zu benennen sind. Die Konvention nur nach außen freigegebene Schnittstellen einer Komponente zu verwenden, wird also derzeit nur durch die beschriebene Namenskonvention realisiert.

TODO: Diesen Hinweis bei Fertigstellung wieder entfernen

Zieht sich noch nicht durch die ganze Anwendung hindurch. Nochmals Abhängigkeiten zwischen den verschiedenen Paketen prüfen. Evtl. sollte vielleicht jedes Projekt in zwei Projekte geteilt werden. Export und Implementierung. Referenziert werden sollten dann nur die Export-Projekte. So kann die korrekte Verwendung direkt von der Entwicklungsumgebung überprüft werden.

In der Entwicklungsumgebung Eclipse werden Komponenten durch separate Java-Projekte dargestellt. So können zumindest die Abhängigkeiten der einzelnen Komponenten untereinander per Definition der Classpath-Einstellungen sichergestellt werden.

Im Zuge des Build-Prozesses (s. hierzu Kapitel TODO) werden alle Komponenten in separate Jar-Archive übersetzt, gepackt und versioniert. In der näheren Erläuterung der einzelnen Komponenten in den folgenden Abschnitten wird hier nochmals ausführlicher darauf eingegangen.

Abbildung TODO zeigt die zur Zeit in die Anwendung integrierten Komponenten und deren Abhängigkeiten. Als Spezialfälle sind hier ScanScoreCore und ScanScoreDatamodel

zu nennen. Es handelt sich bei diesen beiden Artefakten eher um Klassenbibliotheken als um Komponenten im oben definierten Sinne. Sie werden daher auch nicht über nach außen definierte Schnittstellen in die Anwendung eingebunden, sondern als echte Klassenbibliotheken in allen anderen Komponenten referenziert, was durch die Depndency `<<use>>` visualisiert wird. Der Übersichtlichkeit ist die `<<use>>`-Dependency nur für die beiden Komponenten ScanScorePersistence und ScanScoreEvaluationsheetPdf dargestellt.

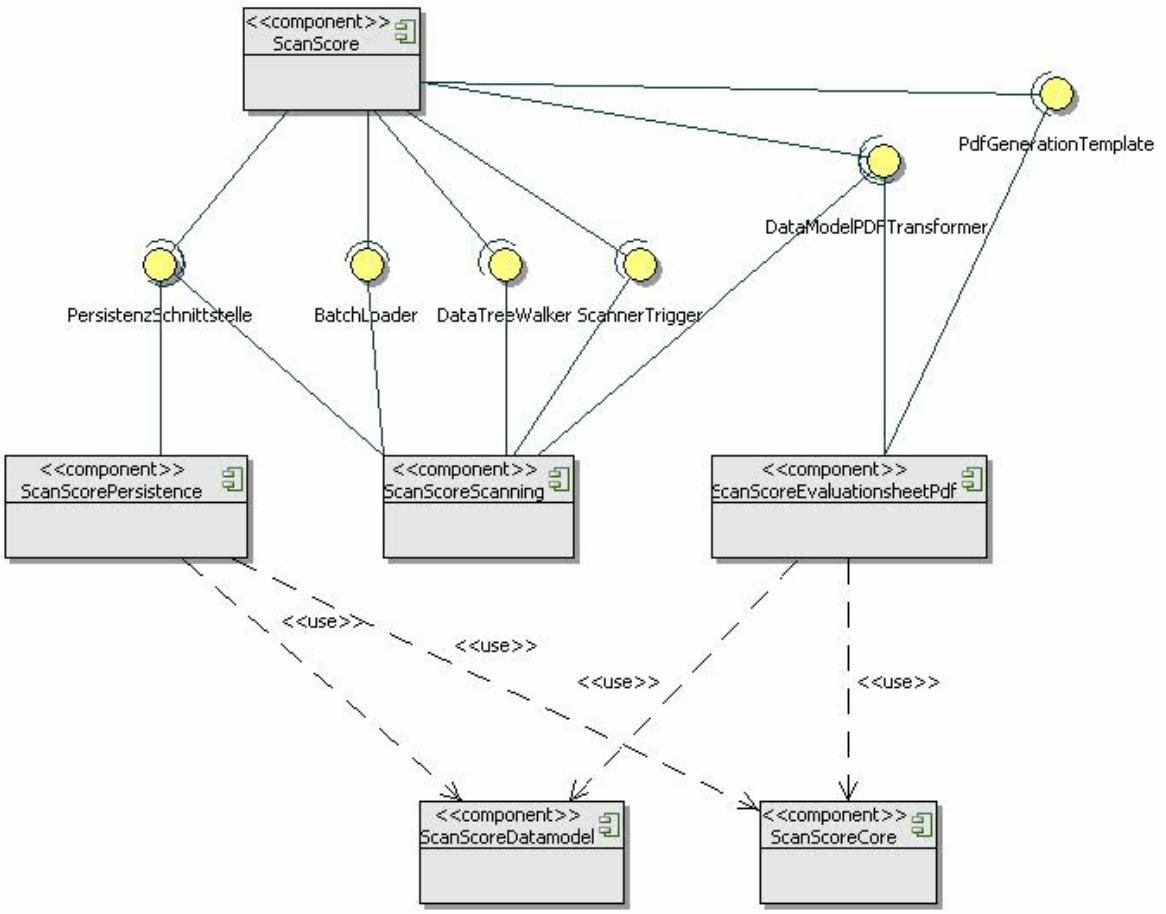
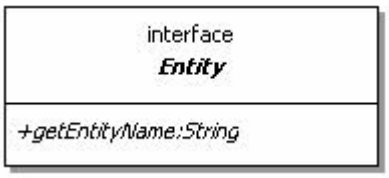


Abbildung 5 Übersicht Komponenten

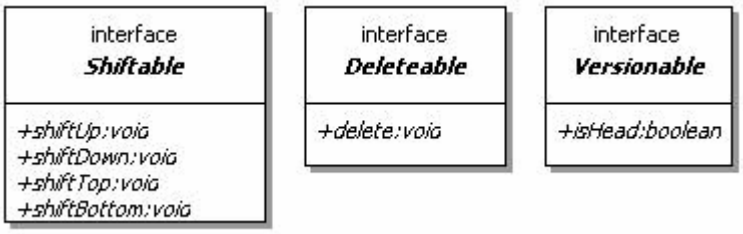
4.2.1 ScanScoreDatamodel

Die Bibliothek ScanScoreDatamodel beinhaltet die Modellierung der im Datenmodell (s. Kapitel TODO) entworfenen Entitäten als Java-Interfaces. Implementiert werden diese durch spezielle Persistenz-Klassen der Komponente ScanScorePersistence.

Jede Entität hat das Interface `com.hdm.scanscore.datamodel.Entity` zu implementieren:



Zusätzlich zu den im Datenmodell entworfenen Entitäten wurden noch einige weitere nützliche Interfaces definiert, die von einigen der Entitäten erweitert werden.



4.2.2 ScanScoreCore

Die Core-Komponente stellt allen anderen Komponenten nützliche Utilities in Form einer Klassenbibliothek zur Verfügung. Jegliche Funktionalität mit solchem Basis-Charakter sollte in die Klassenbibliothek aufgenommen werden, um so einen möglichst hohen Grad an Code-Wiederverwendung zu erreichen.

Als besonders zu nennen sind in diesem Zusammenhang die im Folgenden beschriebenen Funktionalitäten.

4.2.2.1 Mehrsprachigkeit und Localization

TODO

4.2.2.2 Logging

TODO

4.2.2.3 Nebenläufigkeiten durch Tasks

TODO

4.2.2.4 States, StateWatcher und StateChanger

TODO

4.2.3 ScanScorePersistence

Die in Abbildung TODO als Persistenz-Schnittstelle deklarierte Schnittstelle wird von der Komponente ScanScorePersistence für alle Komponenten, welche lesenden oder schreibenden Zugriff auf die Datenbank benötigen, bereit gestellt. Sie lässt sich dabei weiter in die in Abbildung TODO dargestellten Interfaces aufgliedern:

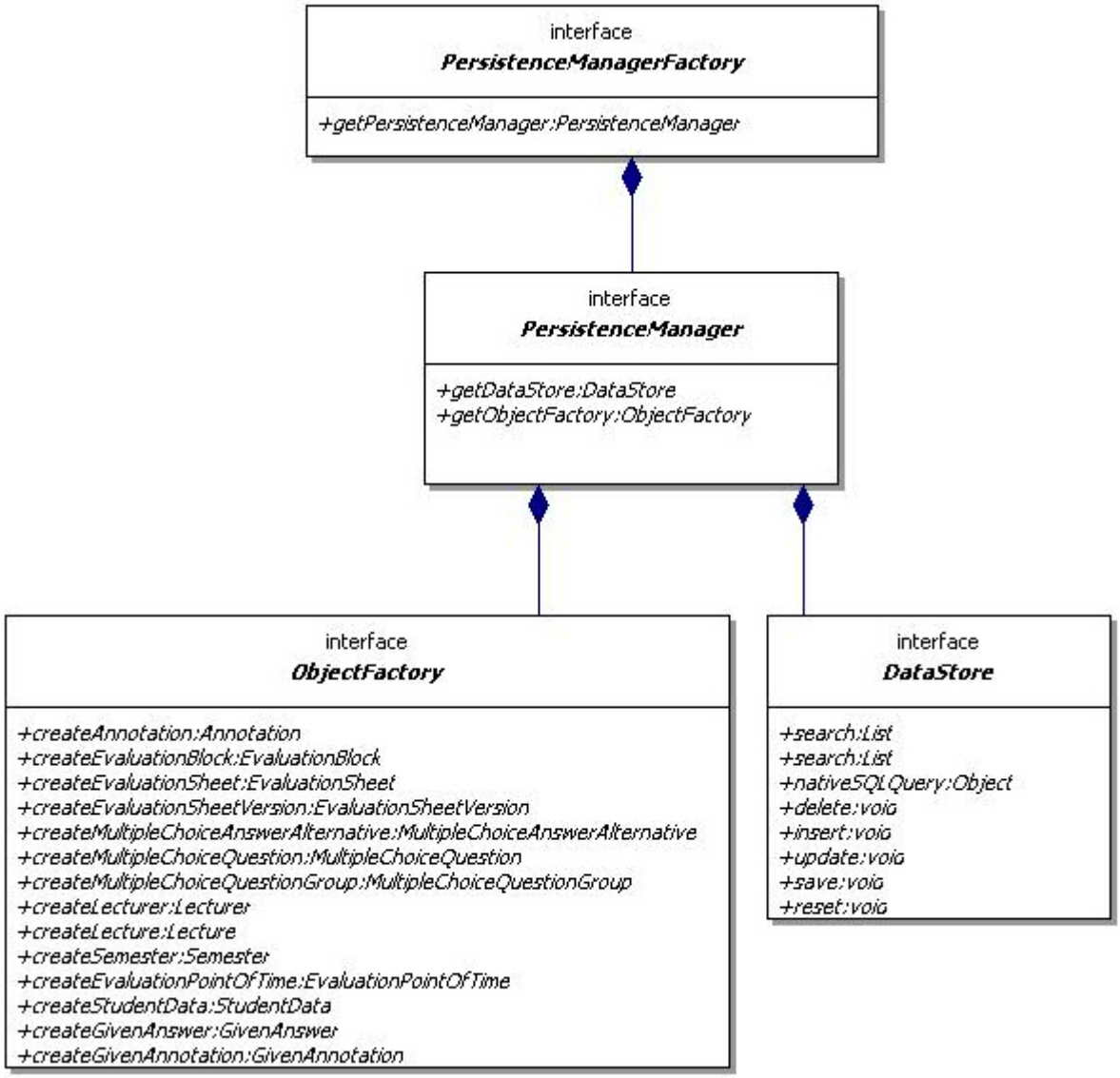


Abbildung 6 Zentrale Persistenz-Interfaces

Von zentraler Bedeutung sind die beiden Interfaces

- `com.hdm.scanscore.persistence.export.DataStore`

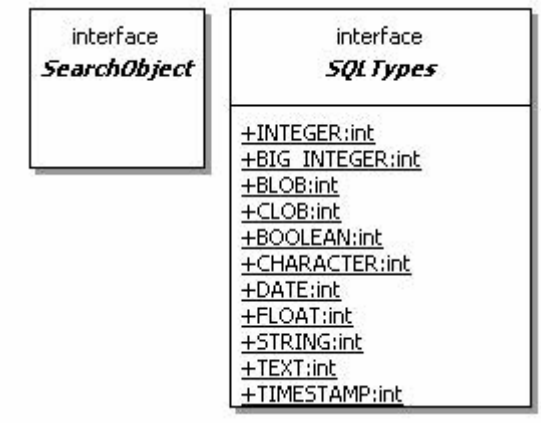
- und com.hdm.scanscore.persistence.export.ObjectFactory.

Die ObjectFactory stellt Methoden für die initiale Erzeugung neuer Entitäten bereit. Der DataStore stellt verschiedene Methoden zur Abfrage von Entitäten bereit. Sollen Entitäten geändert oder gelöscht werden, so sind diese über die Methoden `update(Entity entity)` bzw. `delete(Entity entity)` in den DataStore einzustellen. Zu beachten ist hierbei, dass die Änderungen erst bei einem Aufruf von `save()` tatsächlich an die Datenbank weitergereicht und wirksam werden.

Konkrete Implementierungen einer ObjectFactory und eines DataStore erhalten Clients über den `PersistenceManager`, welcher wiederum über eine `PersistenceManagerFactory` erzeugt wird. Die konkrete Implemetierung der `PersistenceManagerFactory` ist der Anwendung über die Angabe der implementierenden Klasse in der Konfigurationsdatei `startup.xml` (s. Kapitel TODO) bekannt zu machen. Durch den Mechanismus des AbstractFactory-Patterns und der lediglich aus Interfaces bestehenden Persistenz-Schnittstelle werden die restlichen Komponenten der Anwendung von der konkreten Implementierung der Persistenz-Schicht abgekoppelt, und diese jederzeit durch eine andere Implementation ersetzbar.

Zusätzlich zu den bereits beschriebenen Interfaces enthält die Persistenz-Schnittstelle noch die beiden Interfaces

- `com.hdm.scanscore.persistence.export.SearchObject`
- und `com.hdm.scanscore.persistence.export.SQLTypes`:



TODO: Beschreiben

4.2.4 ScanScoreScanning

TODO

4.2.5 ScanScoreEvaluationsheetPdf

TODO

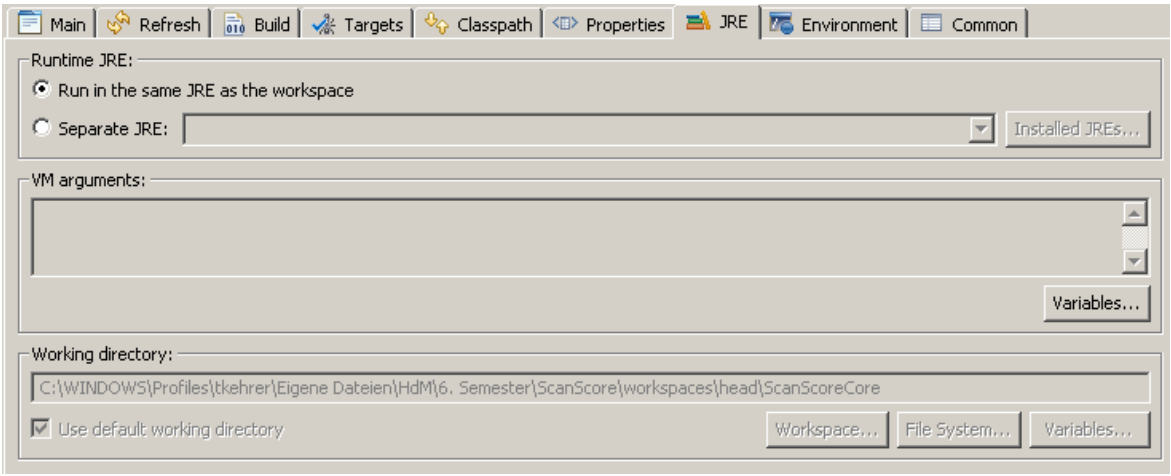
4.2.6 ScanScore

TODO

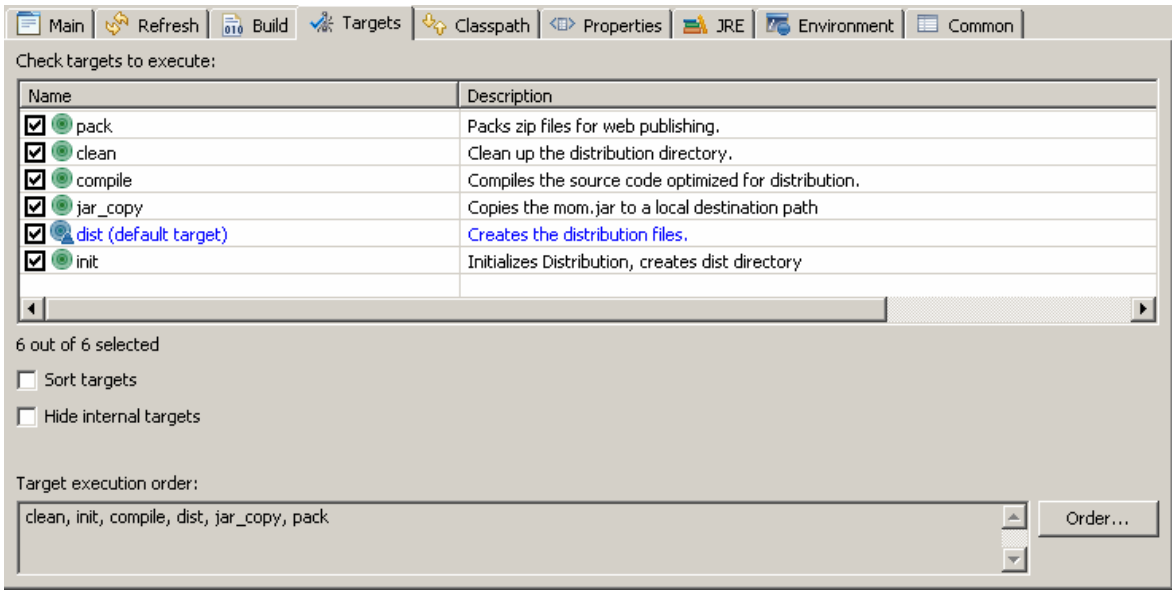
4.3 Build-Prozess

Wie bereits beschrieben, wird im Zuge des Build-Prozesses jedes Komponente, und somit jedes zugehörige Eclipse-Projekt, in ein separates Jar-Archiv übersetzt. Als Build-Tool wird Apache-Ant verwendet. Jedes Projekt enthält somit eigene make-Datei (build.xml), welche direkt aus der Entwicklungsumgebung heraus gestartet werden können.

Wichtig ist, dass der Ant-Classpath zu allen benötigten Bibliotheken richtig gesetzt ist. Am Besten wird Ant in derselben Java-Runtime wie auch die Eclipse-Workbench ausgeführt:



Die Reihenfolge der auszuführenden Ant-Targets sollte folgendermaßen gesetzt werden:



Zusätzlich sollte folgende Parameter gesetzt werden:

- **debug:** Inputparameter für den Java-Compiler. (Default: false)
- **destination:** Verzeichnis für die übersetzten Jar-Archive und Bibliotheken
- **version:** Die Version der zu übersetzenden Komponente (s. Abschnitt TODO); wird in die Manifest-Datei generiert.

4.4 Versionierung der Komponenten

Die Versionen der Komponenten werden in der Datei release_notes.txt der jeweiligen Projekte verwaltet. Vorgesehen ist hier ein dreistelliges Versionierungskonzept. Jede Version ist mit Änderungsdatum, Autor und einer kurzen Beschreibung zu versehen, bspw.:

```
2.0.7      (2006-08-03) tkehrer

           Anpassung von XY / Änderungen wegen …
```

Die zu erhöhende Ziffer richtet sich nach der Größe der Änderung und liegt im Ermessen des Entwicklers. Je nach Größe der Anpassung und Anzahl der Änderungen sollten Versionen getagged werden. Vor einem Build-Prozess für die Produktivumgebung ist immer zu taggen und die aktuelle Version als Ant-Parameter anzugeben, um auf die entsprechenden Versionen in der Entwicklungsumgebung zugreifen zu können, und evtl. auftretende Fehler nachstellen zu können.

5 Persistenzschicht – Implementation

5.1 Implementation unter Verwendung von Hibernate

Nachdem in Kapitel TODO bereits die Verwendung der Komponente ScanScorePersistence durch andere Komponenten beschrieben wurde, und dabei insbesondere die Persistenz-Schnittstelle, d.h. die Außensicht der Komponente beschrieben wurde, soll in diesem Kapitel die konkrete Implementation unter Verwendung der Hibernate-Library beschrieben werden. Wie bereits beschrieben lässt sich diese Implementation jeder Zeit durch eine andere ersetzen. Da Hibernate jedoch relativ gut dokumentierte OpenSource Software ist, wird hier im Moment nur wenig für eine Änderung der Implementation sprechen.

5.1.1 Zwischenschicht / Adapter-Schnittstelle

TODO

5.1.2 Konkrete Implementation der Entities

TODO

5.1.3 Hibernate OR-Mapping

TODO

6 Plugin-Konzept für GUI-Module

Für die grafische Oberfläche von Scanscore wurde ein Plugin Konzept realisiert, mit dem der Applikation jederzeit einfach neue Dialoge hinzugefügt werden können. Sämtliche Module sind und müssen in Java Swing realisiert werden, um ein einheitliches Look &Feel zu gewährleisten.

Um ein neues Plugin zu schreiben, muss im Package `com.hdm.scanscore.application.ui.module` eine neue Modulklasse angelegt werden, die von `ScanScoreGUIModule` erbt. Falls das neue Plugin Oberflächenelemente enthält, die regelmäßig upgedatet werden müssen, muss die die neue Modulklasse die Methode `public void updateDataBaseElements()` überschreiben.

Die Modulklasse ist nur als Wrapper für das eigentliche Panel gedacht, das dann die gewünschten Oberflächenelemente enthält. Dieses Panel muss von JPanel erben und das Interface ScanScoreView implementieren. Ausserdem muss das Panel zwei statische Methoden implementieren:

```
public static void setApplication(ScanScoreApplication
application) {
        _application = application;
}
public static StatisticsSelectionPanel getView() {
        if (_view == null) {
            _view = new StatisticsSelectionPanel();
        }       return _view;

}
```

Das gewünschte Panel kann dann im Konstruktor der Modulklasse wie folgt eingebunden werden:

```
StatisticsSelectionPanel.setApplication(getApplication());

setView(StatisticsSelectionPanel.getView());
```

Die Hauptanwendung von ScanScore wird über die Klasse ScanScoreApplication.java abgebildet. Sie bildet den Rahmen für alle Plugins, hier werden auch neue Plugins hinzugefügt.

Für ein neues Modul muss schließlich am Anfang der Klasse eine neue Konstante mit dem Namen der Klasse eingefügt werden, ausserdem muss diese Konstante dem modules Array hinzugefügt werden. Das Plugin wird dann über reflection geladen.

```
      private final String MASTER_DATA = "MasterDataEditorModule";

      private final String SCAN_PERSPECTIVE = "ScanPerspectiveModule";

      private      final      String      EVALUATIONSHEET_EDITOR      =
"EvaluationSheetEditorModule";

      private      final      String      STATISTICS_SELECTION      =
"StatisticsSelectionModule";

      String[]    modules    =    {    MASTER_DATA,    SCAN_PERSPECTIVE,
STATISTICS_SELECTION };
```

Zu beachten ist ausserdem, dass eine strenge Trennung von Oberfläche und Logik einzuhalten ist. Sämtliche logische Aktionen, wie etwa das Speichern von Daten werden über sogenannte ActionCommand Klassen getriggert (siehe z.B. die Klasse `EvaluationSheetActionCommands`).

7 Bereits realisierte Plugin-Module

7.1 Pflege von Evaluierungsbögen

Das Modul „Pflege von Evaluierungsbögen“ dient zum Erstellen, Bearbeiten, Speichern und Drucken von Evaluierungsbögen. Das Modul ist über ein JSplitpane in zwei Bereiche untergliedert, wobei in beiden Bereichen modifizierte JTrees verwendet werden, um sowohl die Gesamtzahl erstellter Bögen, als auch die einzelnen Bögen betrachten und bearbeiten zu können.

Die Modulklasse hierfür heisst `EvaluationSheetEditorModule`, das entsprechende Panel findet sich im Konstruktor der Modulklasse.

7.2 Stammdatenpflege

Über das Modul „Stammdatenpflege“ können Vorlesungen, Dozenten, Semester und Bewertungszeitpunkte erstellt und verändert werden. Hierfür wurden editierbare JTables verwendet, so dass schnell und einfach editiert werden kann.

Die Modulklasse hierfür heisst `MasterDataEditorModule`, das entsprechende Panel findet sich im Konstruktor der Modulklasse.

7.3 Scan der Daten

Über das Scan Modul wird der Scanner mit Mehrfacheinzug angesprochen, sowie die Speicherung der gescannten Daten angestoßen.

Zunächst werden alle im Scanner vorhandenen Bögen eingescannt und temporär zwischengespeichert. Anschließend werden die gespeicherten Bilder nacheinander in den Speicher geladen und über die verwendete Bildbearbeitungsbibliothek weiterverarbeitet. Hierfür wird in allen 4 Ecken des Dokumentes nach dem Barcode gesucht. Ist dieser gefunden, werden über die drei weiteren Markierungen die Skalierungen und Drehungen berechnet und ausgeglichen (falls die Abweichungen nicht zu groß sind). Im Anschluss wird die im Barcode codierte Version des Bogens aus der Datenbank geladen und in einen Baum mit Positionierungsinformationen überführt. Dieser Baum wird verwendet, um die in der aufbereiteten Bilddatei vorhandenen Markierungen anzusteuern und auszuwerten. Die Grauwerte der einzelnen Auswertungskästchen werden berechnet und verglichen, das Kästchen mit dem höchsten Grauwert über einer gewissen Toleranzgrenze (bzw. das mit

dem zweithöchsten bei zwei hohen Grauwerten (Antwort durchgestrichen)) wird verwendet und diese Information wird in der Datenbank abgespeichert.

Die Modulklasse hierfür heisst `ScanPerspectiveModule`, das entsprechende Panel findet sich im Konstruktor der Modulklasse.

7.4 Auswertung der Daten

Mithilfe des Moduls „Auswertung der Daten“ können die gescannten und gespeicherten Daten anhand verschiedener Kriterien ausgewertet werden. So wird zunächst über verschiedene Auswahlmöglichkeiten bestimmt, welche Vorlesung aus welchem Semester, und welche Beurteiler daraus (nur weiblich, nur männlich…) zur Auswertung betrachtet werden sollen. Ist diese Auswahl getroffen, werden die entsprechenden Daten aus der Datenbank geladen und im Speicher gehalten. Der Benutzer kann anschließen wieder über einen Baum auswählen, welche Auswertungen er betrachten will, die dann neben dem Baum angezeigt werden. Ausserdem bietet dieses Modul die Möglichkeit, die ausgewählten Daten in einem Report im PDF Format abzuspeichern.

Das Hautpanel hat ist in zwei Subpanels untergliedert, eins für die Auswahl und Anzeige der Daten und eines für die Buttons zur Navigation bzw. Ausführung von Aktionen. Beide Panels haben ein CardLayout, bei dem je nach Auswahl ein anderes Panel angezeigt wird. Das Panel zur Anzeige der Daten ist ein JSplitpane, wobei wie bei der Pflege der Evaluierungsbögen links ein modifizierter JTree verwendet wird und rechts ein Standardpanel mit einem TabbedPane, das die einzelnen Daten gliedert.

Die Modulklasse hierfür heisst `StatisticsSelectionModule`, das entsprechende Panel findet sich im Konstruktor der Modulklasse.