Dinomeda Projekt

Projektbeschreibung

von

Kevin Krammer, Martin Oswald und Mattias Welponer

Institut für Informationsverarbeitung und Computergestützte neue Medien
(IICM),
Technische Universität Graz
A-8010 Graz

05. März 2003

Betreuer: DI Christof Dallermassl

1 Allgemein

Die Dinomeda Class Library ist als Subpackage des Dinopolis Projektes (`www.dinopolis.org`) angelegt und soll einer Applikation ermöglichen, Metadaten in Multimediaformaten zu bearbeiten, wobei als einzige Kenntnis über das Dateiformat deren MIME Type nötig ist.
Abhängig vom MIME Type werden über eine zentrale Komponente pluginbasierte Handlerklassen geladen.

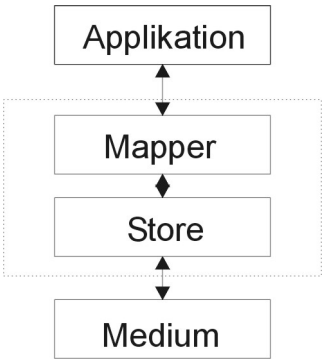

Abbildung 1: Schichtmodell der Metadatenbearbeitung

Die beiden für die Bearbeitung verantwortlichen Klassen sind der Store und der Mapper (siehe Abbildung 1).
Der Store stellt die Schnittstelle zum Medium bereit, hängt also davon ab, in welcher Form die Daten vorliegen.
Der Mapper abstrahiert die interne Struktur der Metadaten und bietet der Applikation den Zugriff über die im Dinomeda Schema [Krammer und Oswald 2003] definierten Elementnamen.

Die aktuellen Quellen sind jeder Zeit per anonymen (Passwort ist leer) CVS Zugriff erhältlich:

```
export CVSROOT=:pserver:anonymous@cvs.dinomeda.berlios.de:/cvsroot/dinomeda
cvs login
cvs -z3 co dinomeda
```

2 Systemanforderungen

Benötigt werden ein Java2 kompatibles JDK und, für die Stores des Dinomeda Provider Plugins, zwei externe Bibliotheken:

- PAT: www.javaregex.com

Version: 1.5.3, Lizenz: LGPL
Regular Expressions

- Sixlegs PNG decoder: www.sixlegs.com/software/png Version: 1.2.3, Lizenz: LGPL PNG Handling

Beide Bibliotheken sind in passender Version im CVS Baum verfügbar, um für die Entwickler eine einheitliche Ausgangsbasis zu gewährleisten.

3 Architektur der Pluginservices

Die Plugins im Dinomeda Framework sind Implementationen des DMDServiceProvider Interfaces.
Jeder Provider kann zwei Arten von Services zur Verfügung stellen: Stores und Mapper (Abb. 2)
Ein Store bearbeitet dabei das Medium, zum Beispiel die Datei oder den Stream, und erlaubt das Laden, Schreiben und Löschen von Metadaten im Medium.
Der Mapper passt die Namen der Metadatenfelder und das Format ihrer Inhalte an ein externes Schema an.
Das ermöglicht einer Applikation, die im externen Schema gleichwertigen Felder verschiedener Mediendatentypen auf gleiche Weise anzusprechen.

Die drei grundlegenden IO Operationen sind READ, UPDATE und WRITE:
READ bedeutet, dass die gewünschten Felder aus der Datei in den Speicherbereich des Stores geladen werden. Sind dort für die angeforderten Felder Daten vorhanden, werden sie durch die neu gelesenen Daten überschrieben.
UPDATE bedeutet, dass Daten, die im Store geändert wurden, an die entsprechenden Stellen in der Datei zurückgeschrieben werden.
Im Gegensatz dazu bedeutet WRITE, dass nach der Operation nur die angegebenen Felder in der Datei vorhanden sind. Alle dort verher vorhandenen Metadaten werden gelöscht.

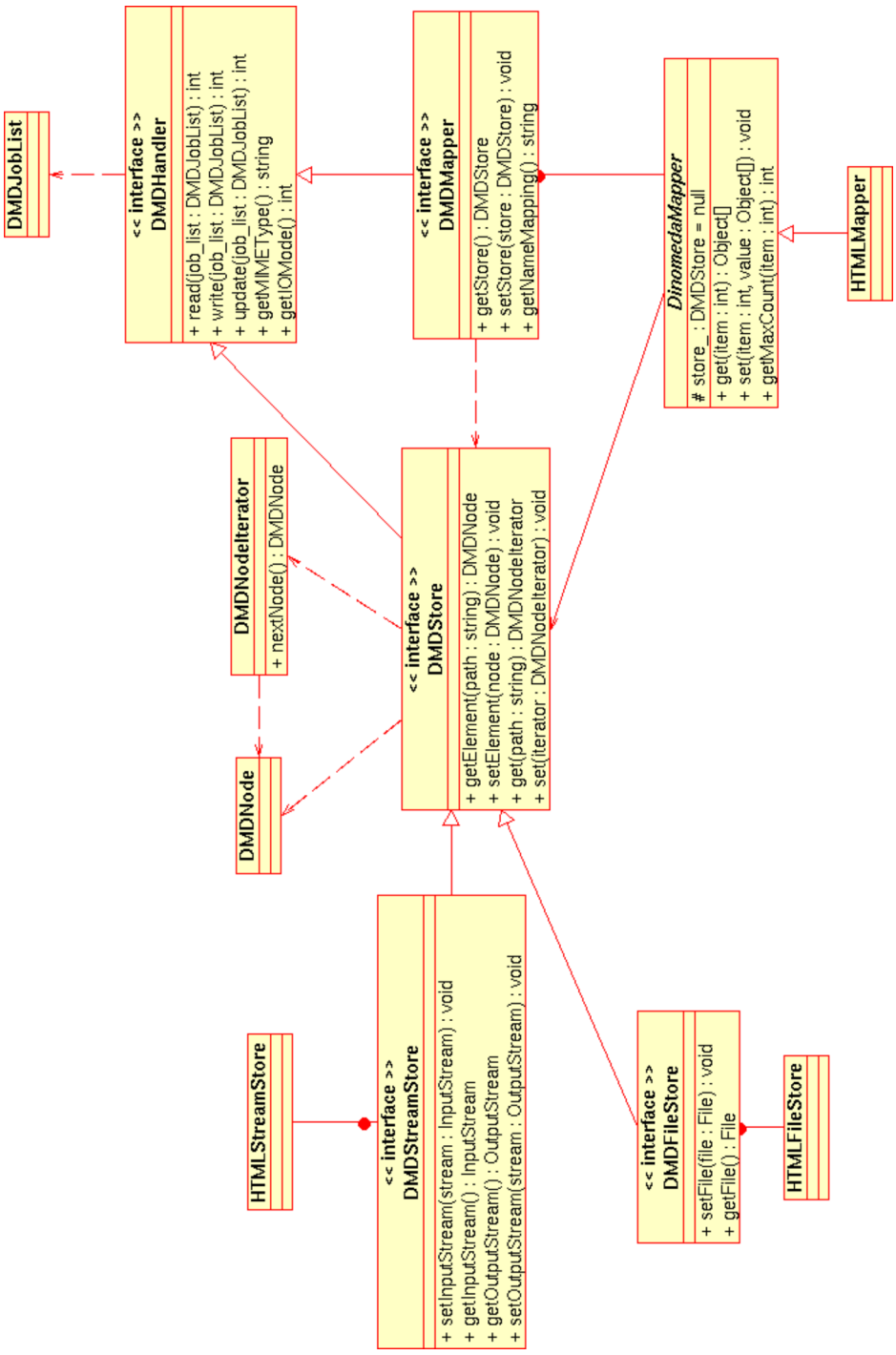

Abbildung 2: Klassendiagramm der Plugins

Abbildung 3 zeigt den Ablauf eines gemappten Zugriffs. Die Applikation fordert eine Instanz eines Dinomeda Mappers für HTML an und einen HTML Store für Dateizugriff.
Dann wird der Erzeuger des Dokuments angefordert, der im Dinomeda Schema Creator heißt. Der Mapper setzt nun den Elementnamen auf “author” um, da dies die korrekte Bezeichnung für den HTML Store ist.

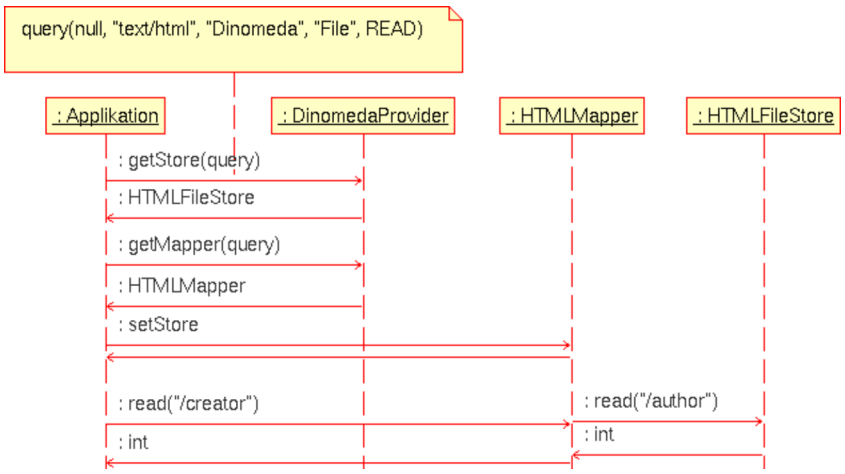

Abbildung 3: Beispiel eines gemappten Zugriffs

4 Auffinden und Laden der Plugins

Damit die Applikation keine konkrete Impementierung von Plugins kennen muss, sondern über Anforderungen ein geeignetes Plugin suchen und laden kann, stellt die Bibliothek eine zentrale Stelle zum Auffinden und Laden zur Verfügung.
Diese Stelle ist der Trader, dessen Schnittstelle in DMDTrader deklariert ist.
Die beteiligten Klasse sind in Abbildung 4 abgebildet.

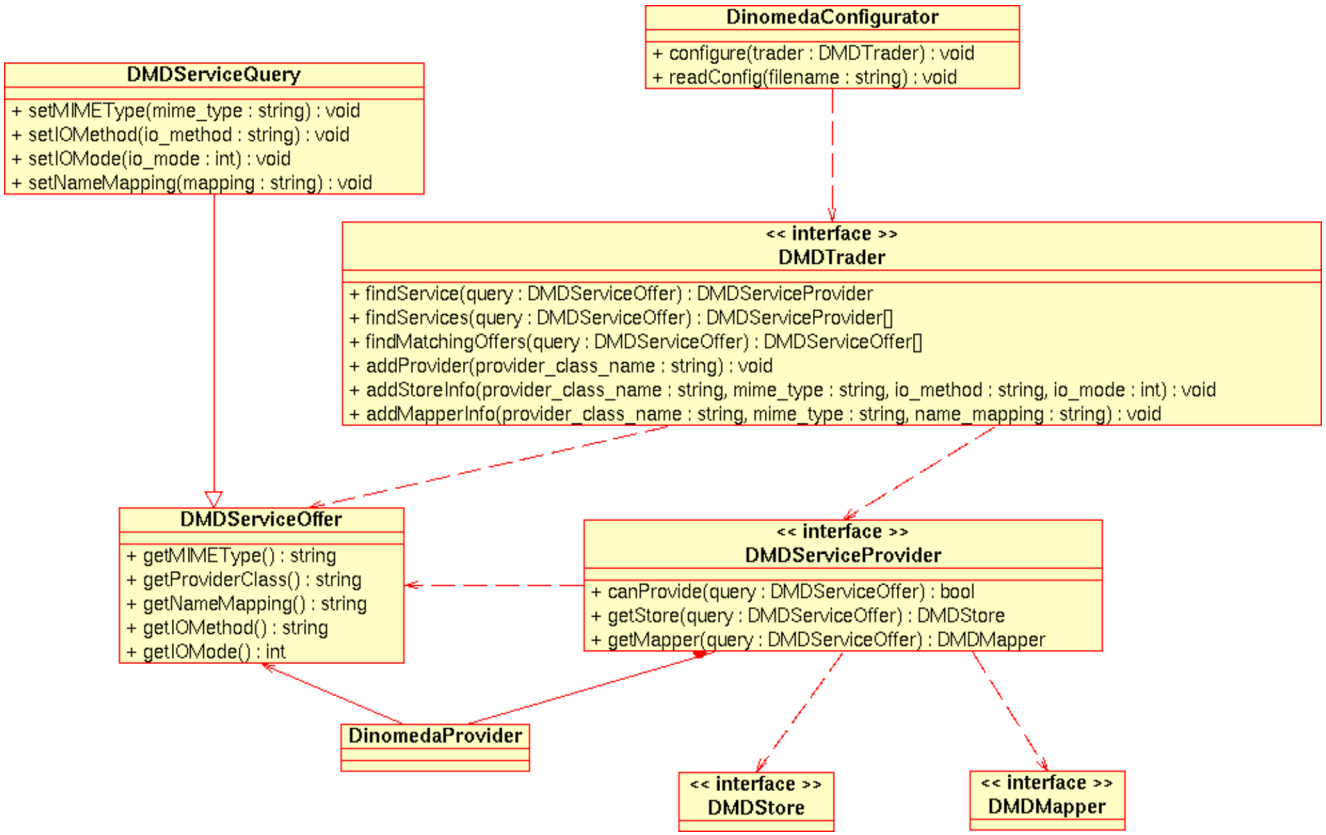

Abbildung 4: Plugin-Verwaltungsklassen

Der Trader wird von der Applikation oder einer Konfigurationsklasse mit Daten über Service Provider (Plugins) und deren Services ausgestattet (siehe Abb. 5), anhand deren in Folge Anfragen nach Plugins beantworten kann.

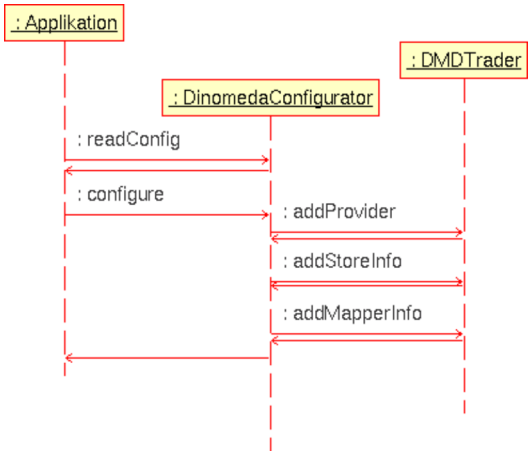

Abbildung 5: Konfiguration eines Traders

Weiters kann die Applikation einen konfigurierten Trader dazu benutzen, einen geeigneten Service Provider zufinden und auch zu erzeugen (Abb. 6)

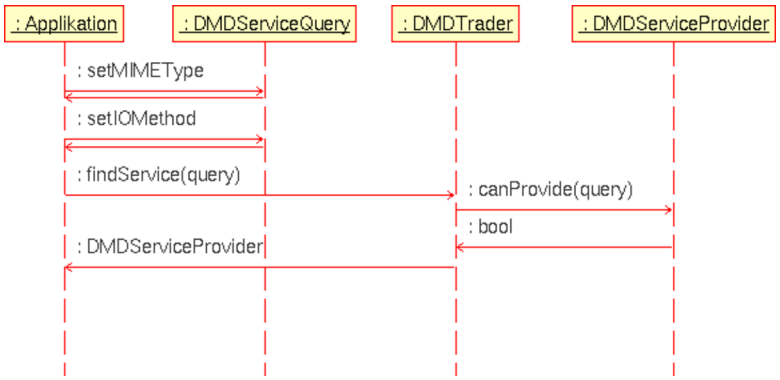

Abbildung 6: Anfrage an einen Trader

5 Beispielimplementation des Dinomeda Teams

Abbildung 7 zeigt die Struktur des Beispielplugins der Dinomeda Entwickler. Es enthält Stores und Mapper für HTML, PNG, PDF und MP3(ID3V1, ID3V1.1). Das externe Schema der Mapper ist das Dinomeda Daten Schema.

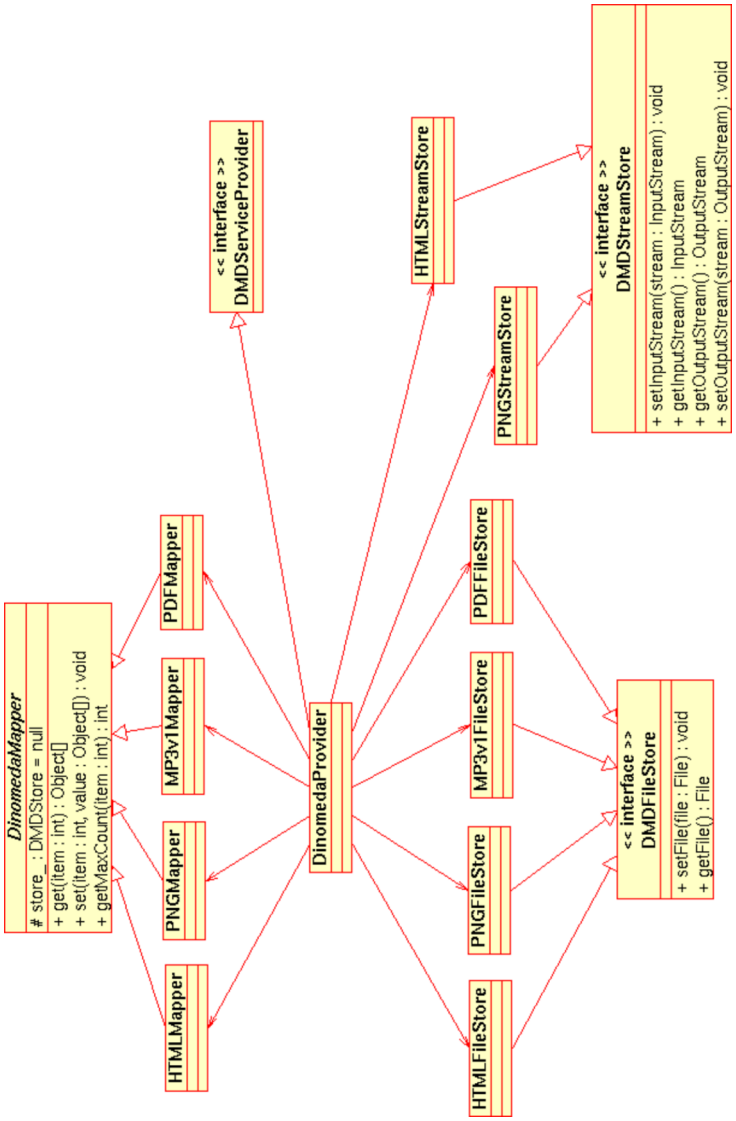

Abbildung 7: Beipielimplementation eines Plugins

Abbildungsverzeichnis

Literatur

[Krammer und Oswald 2003] Kevin Krammer und Martin Oswald: *Metadaten und Metadatenstandards* January 31, 2003