3.2 Migration iDAIfield

3.2.1 Ausgangssituation

Der Datenbestand von iDAI.field verteilt sich auf verschiedene Filemaker-Datenbanken, die jeweils von verschiedenen Projekten genutzt werden. Obwohl alle auf der gleichen Grundstruktur beruhen, haben sie sich mit der Zeit je nach Projektbedarf auseinanderentwickelt: Es kann in jeder Projektdatenbank unterschiedliche Tabellen, Felder, Wertelisten oder Layouts geben.

3.2.2 Überlegungen zur Migration

Filemaker bietet die Möglichkeit eines XML-Exports und verwendet dabei ein eigenes XML-Format. Aufgrund der vielen verschiedenen Inkarnationen von iDAI.field und der daraus resultierenden Notwendigkeit, später Migrationsarbeiten von verschiedenen, unterschiedlich technisch-bewanderten Personen durchführen lassen zu können, wurde entschieden, bei der Durchführung der Migration keine direkte Verbindung mit einer PostgreSQL-Instanz herzustellen. Diese wird stattdessen über ihre XML-Schnittstelle angesprochen, um die Eingabe fehlerhafter Daten zu verhindern und das Rollensystem nicht zu kompromittieren. Die Umwandlung des Filemaker- in das OpenInfRA-XML-Format soll dabei mit XSLT durchgeführt werden. Diese Vorgehensweise hat noch weitere Vorteile:

1. Durch die Umwandlung des Filemaker-Formats an sich wird eine allgemeine Schnittstelle zwischen Filemaker und OpenInfRA geschaffen, die von sämtlichen Filemaker-basierten Projekten aktuell und in Zukunft genutzt werden kann. Dies gilt natürlich auch für alle iDAI-field-Instanzen, die trotz ihrer heterogenen Strukturen den gleichen Migrationsvorgang durchlaufen können.
2. X-Technologien sind in der OpenInfRA-Welt bereits im Einsatz, beispielsweise innerhalb der XML-Schnittstelle. So wird vermieden, dass weitere Technologien bzw. Programmiersprachen eingesetzt werden müssen, deren Beherrschung eventuell nicht bei allen Projektteilnehmenden vorausgesetzt werden kann.
3. XSLT ist auch ohne große Programmierkenntnisse relativ einfach zu lesen und zu modifizieren. Falls zu einem späteren Zeitpunkt Bedarf bestehen sollte, könnten Änderungen vorgenommen werden, ohne den Programmcode eines Programms zu verändern, neu kompilieren zu müssen, etc.

3.2.3 Konkrete Umsetzung

Ein Problem von XML-Exporten von Filemaker ist, dass diese häufig dem XML-Standard nach illegale Zeichen enthalten und so nicht direkt umgewandelt werden können, da der XSLT-Prozessor sich weigert. Vor der Konvertierung ist es also notwendig, alle illegalen Zeichen programmatisch zu entfernen.
Danach kann die bereinigte XML-Datei dann mithilfe zweier XLST-Dateien (*filemaker_locale.xsl, filemaker.xsl*) in das OpenInfRA-Format umgewandelt werden.

Die erste Datei erzeugt sowohl das Wertelisten- als auch die Locale-File und ersetzt in dem Filemaker-XML alle Strings durch jene UUIDs, die auch im Locale-File verwendet werden. Außerdem parst es bereits Beziehungen zwischen Datensätzen heraus und konvertiert die Filemaker-Datei so in ein Zwischenformat.

Dieses Zwischenformat wird mithilfe der zweiten XSLT-Datei dann in valides OpenInfRA-XML umgewandelt. Dieses kann dann, zusammen mit Wertelisten- und Locale-Datei, von der XML-Schnittstelle verstanden und wiederrum zu SQL transformiert werden.

Der komplette Weg vom Filemaker-Export zum OpenInfRA-XML wird in einem Java-Programm umgesetzt, das eine GUI anbietet, damit auch weniger versierte Anwender nicht auf die Kommandozeile angewiesen sind. Es können alle XML-Dateien der zu migrierenden Datenbank ausgewählt werden, im zweiten Schritt können dann noch Dateien mit Wertelisteninformationen ergänzt werden. Momentan werden diese im XML-Format des Filemaker Database Design Reports erwartet.

Wertelisten werden aus dem "Database Design Report" von Filemaker Advanced übernommen, Überlegungen dzau s.: Filemaker-Wertelisten exportieren

Beim Klick auf "Konvertierung starten" wird dann die Konvertierung durchgeführt: die illegalen Zeichen werden entfernt und auf Basis von Saxon 9 die beiden XSLT-Transformationen duchgeführt.

Es ist auch möglich, die Migration ohne das Java-Tool nur anhand der XSLT-Dateien durchzuführen, die Dateien erwarten dabei folgende Parameter:

filemaker_locale.xsl:

- *files_to_transform:* mit Semikolon getrennte Dateipfade aller zu konvertierenden Dateien
- *value_list_files:* mit Semikolon getrennte Dateipfade aller zu einbeziehenden Wertelisten
- *path:* Pfad, in den die konvertierten Dateien geschrieben werden sollen
- *projectname*: Wie heißt das Projekt? Z.B. "Pergamon", etc.
- *locale_for_itf*: Soll eine zusätzliche Locale-Datei für das initiale Themengerüst erstellt werden? (Infos dazu) Wird später zu boolean umgewandelt, daher 0 oder gar nicht gesetzt = false, beliebige Zeichenkette oder Zahl > 0 = true

filemaker.xsl:

- *path:* Pfad, in den die konvertierten Dateien geschrieben werden sollen

Die erste Transformation wird auf eine der zu konvertierenden Dateien angewandt (die zu konvertierenden Dateien ergeben sich aber trotzdem aus *files_to_transform*).
Die zweite Transformation wird auf das Ergebnis der ersten Transformation angewandt.

Achtung: Die Transformation funktioniert ohne das Java-Tool nur, wenn die Ausgangsdateien bereits vorher auf einem anderen Weg von den illegalen Zeichen bereinigt wurden.

Konkret sieht das Muster für eine Migration ohne Java-Tool so aus:
(Für Mac/Linux, es müssen dafür "saxon9.jar", filemaker_locale.xsl" und "filemaker.xsl" im aktuellen Ordner vorhanden sein.)

> *java -cp saxon9.jar net.sf.saxon.Transform -s:/pfad/zu/Datei.xml -xsl:filemaker_locale.xsl -o:transformed.xml files_to_transform="/pfad/zu/Datei.xml;/pfad/zu/Datei2.xml;/pfad/zu/Datei3.xml;" value_list_files=/pfad*

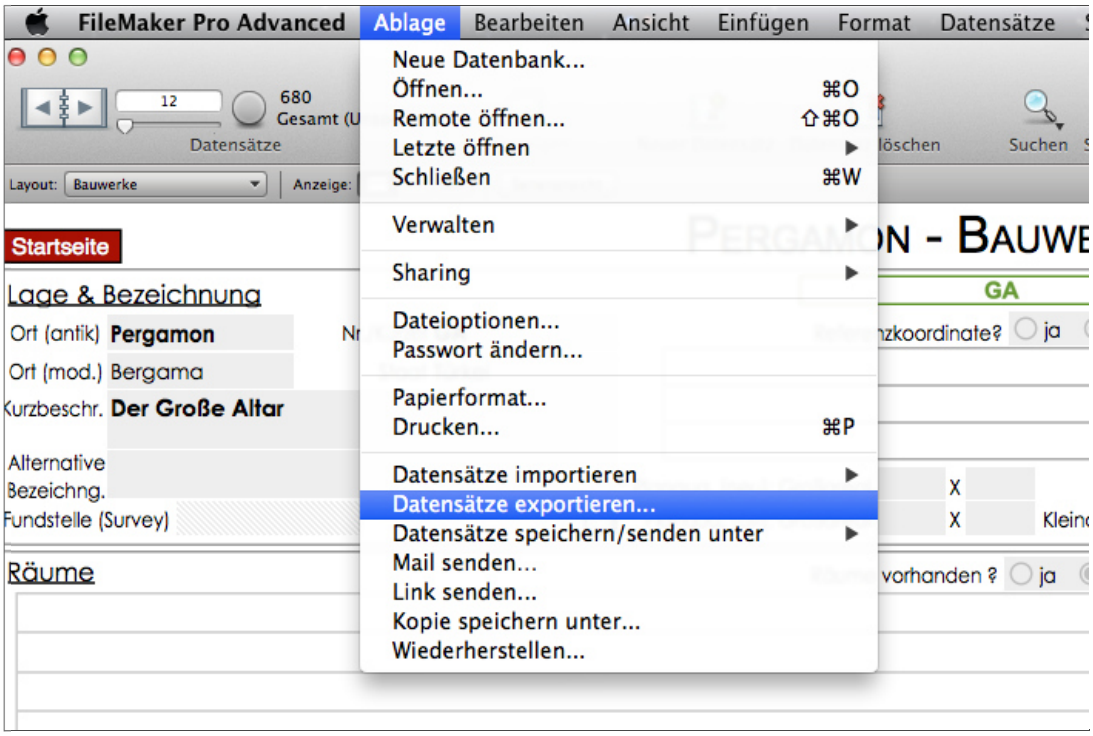
> */zu/DAISeriennummer_fp7.xml path=/pfad/zum/speichern/der/transformierten/dateien projectname=Projektname; java -cp saxon9.jar net.sf.saxon.Transform -s:/pfad/zu/transformed.xml -xsl:filemaker.xsl path=/pfad/zum/speichern /der/transformierten/dateien*

Für sehr viele sehr große Dateien ist es sinnvoll, nicht das Java-Tool zu benutzen, sondern die Migration über ein paar Tage auf einem Server laufen zu lassen. (Z.B. mithilfe von nohup)

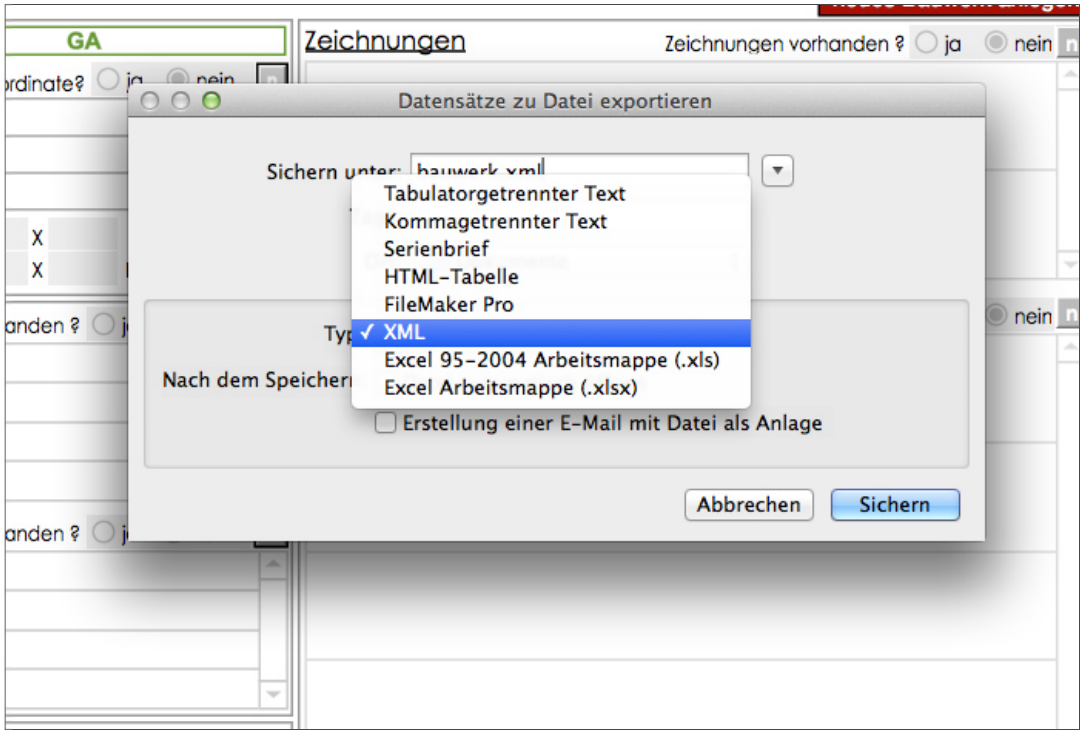
3.2.4 Migrationsanleitung

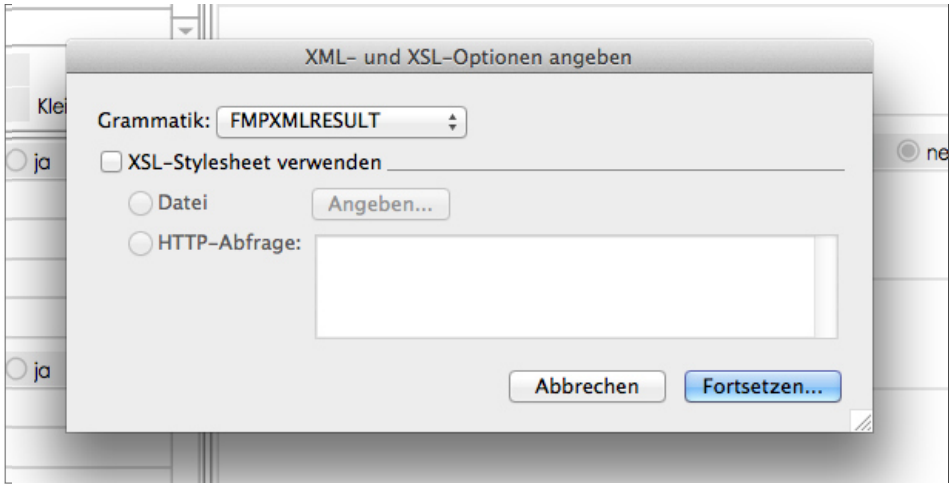
Export aus Filemaker

Zunächst müssen alle Tabellen, die migiriert werden sollen, aus Filemaker exportiert werden.



Dabei sollte XML als Dateiformat ausgewählt werden, mit der Grammatik "FMPXMLRESULT".

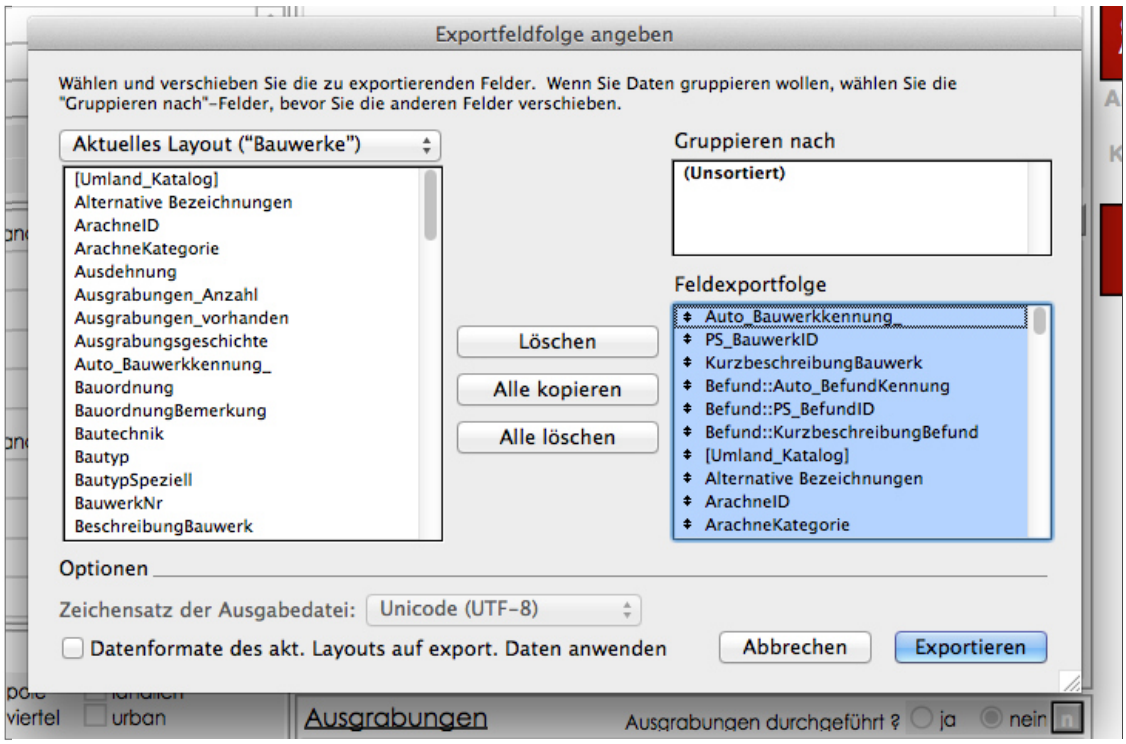




Alle Felder, die migriert werden sollen, müssen ausgewählt werden.
Dabei können folgende Attribute ausgelassen werden:

- Auto_ - Automatisch generierte Formelfelder
- [] - Automatisch generierte Formelfelder
- :: - Attribute aus Bezugsdatensätze
- FS_ - Fremdschlüssel zu verknüpften Datensätzen (nur dann auslassen, wenn die Verknüpfung nicht mitmigriert werden soll)

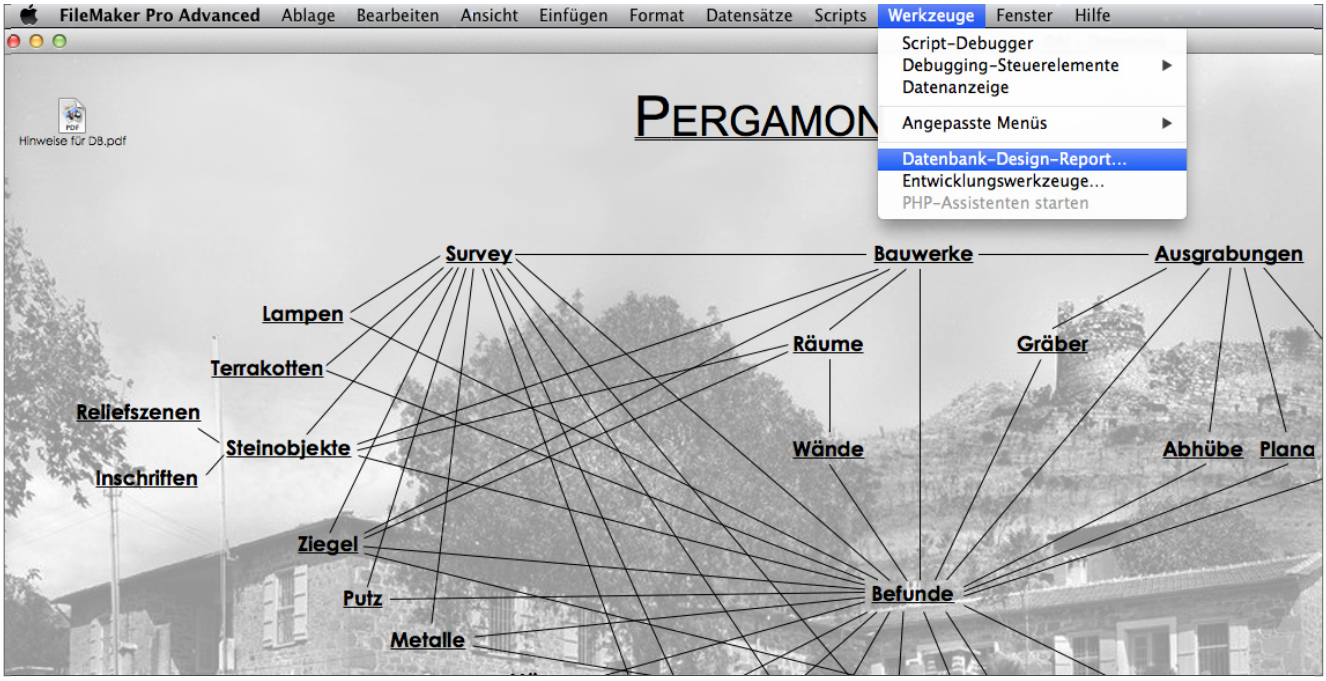
Ggfs. muss bei den exportierten Dateien die Endung .xml hinzugefügt werden.



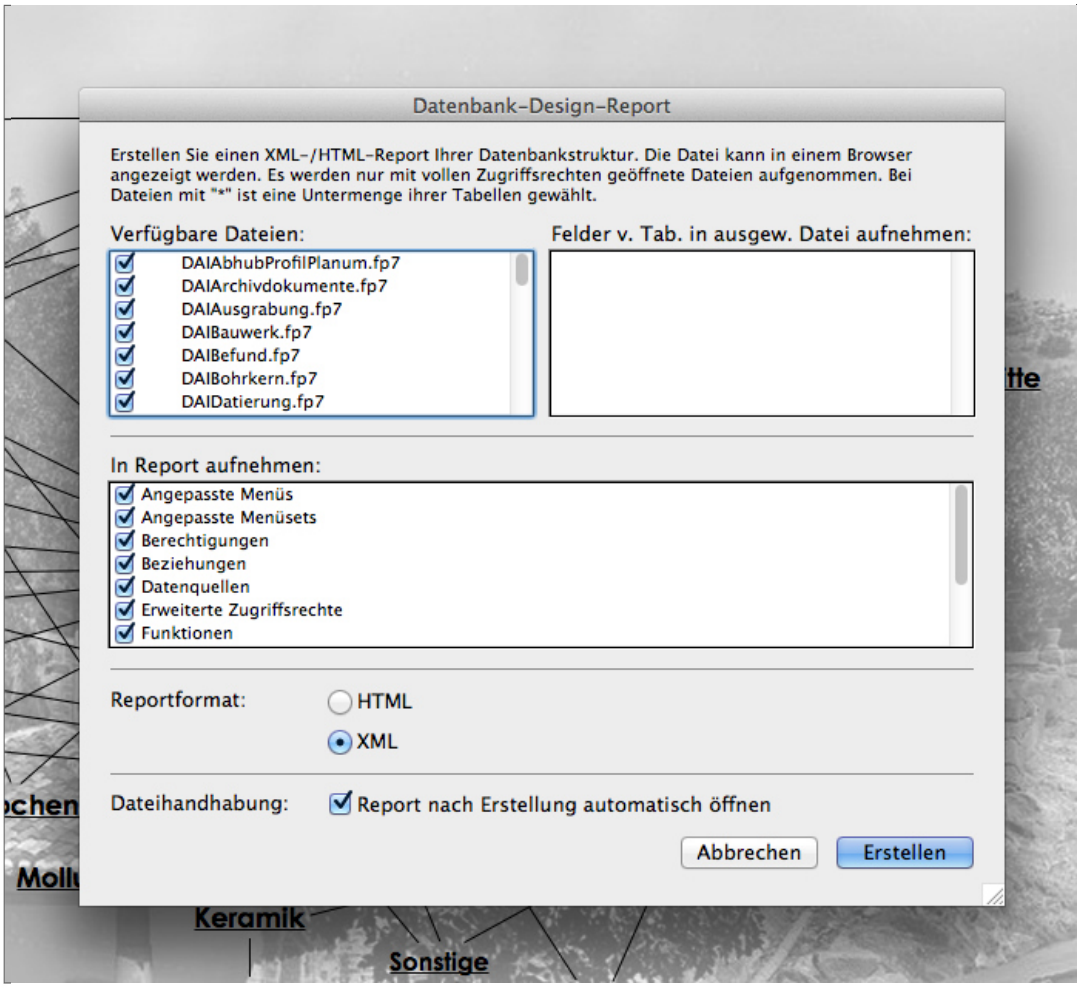
Erstellen von Dateien mit Wertelisten

Achtung: *folgende Vorgehensweise funktioniert nur mit einer* ***Advanced*** *Version von Filemaker!*

Die Wertelisten können aus Filemaker Pro Advanced mithilfe des Database Design Reports exportiert werden. Dazu auf "Werkzeuge/Datenbank-Design-Report..." gehen.



Als Reportformat sollte XML ausgewählt werden:

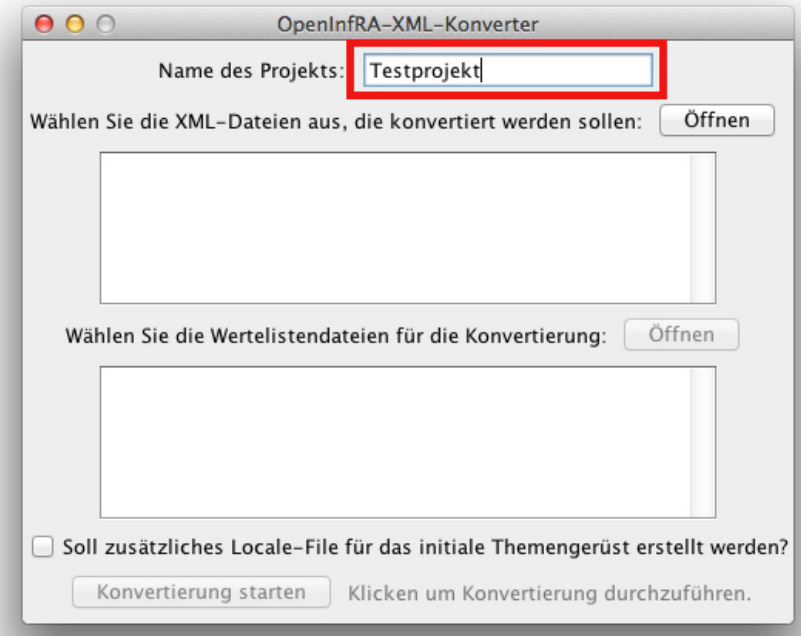


Im Falle der Pergamondatenbank befinden sich dann alle Wertelisten in der Datei "DAISeriennummer_fp7.xml".

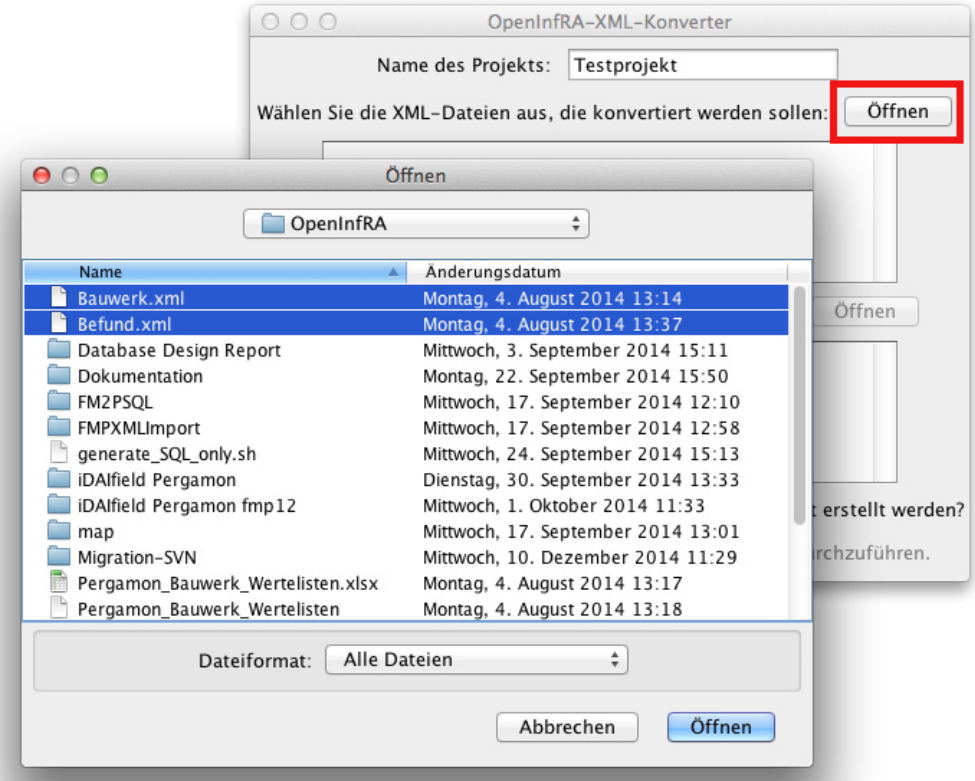
Ausführung des Java-Tools zur Konvertierung

Das Java-Tool befindet sich im iDAIfield-Migrations-SVN.
Check-Out des SVN-Repositories: `svn co https://saffron.informatik.tu-cottbus.de/repos/OpenInfRA_Migration-iDaiField/Ordnername`
Das jar-File muss ausgeführt werden um das Tool zu starten.

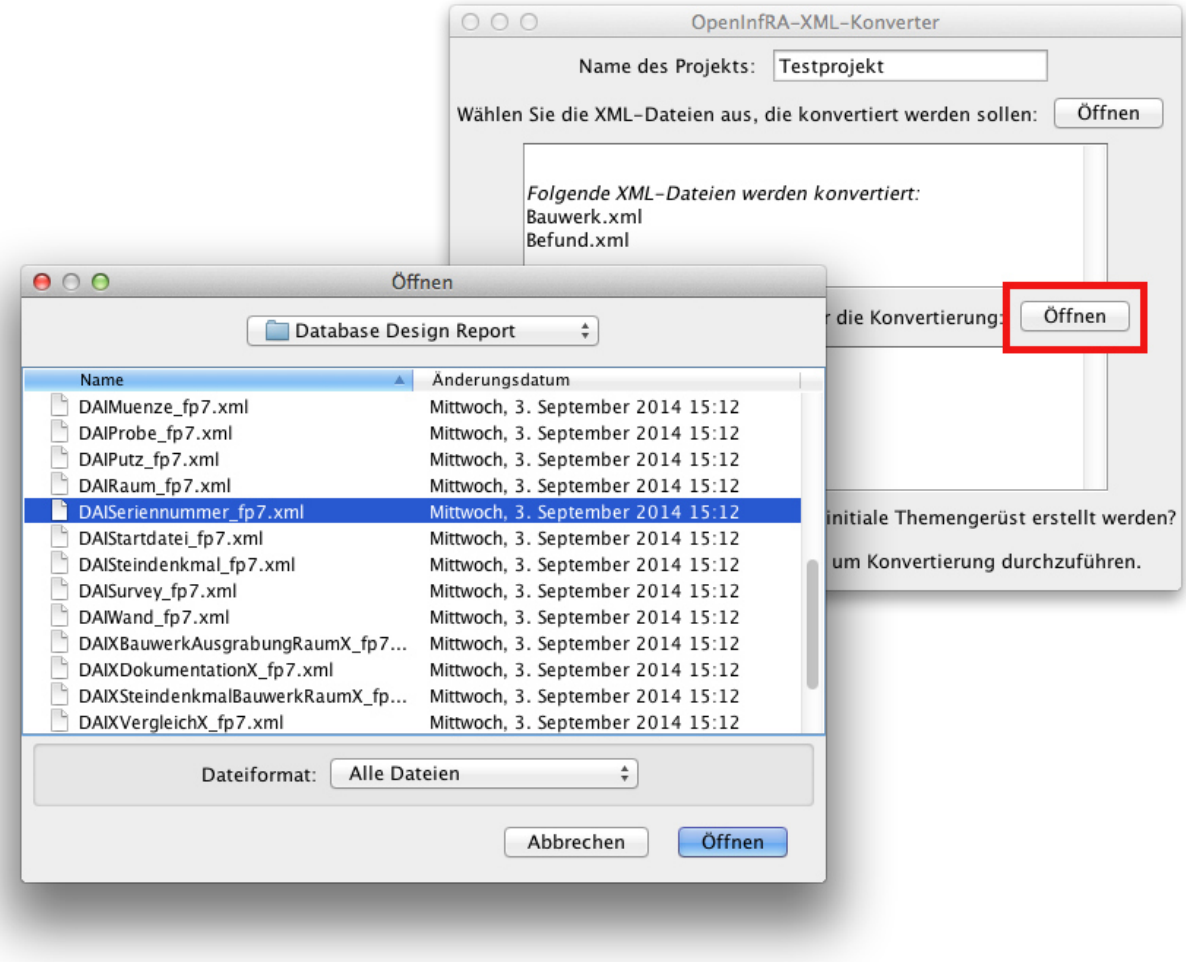
Wenn das Tool gestartet ist, sollte man als erstes in das Textfeld ganz oben den Namen des zu migrierenden Projektes schreiben.



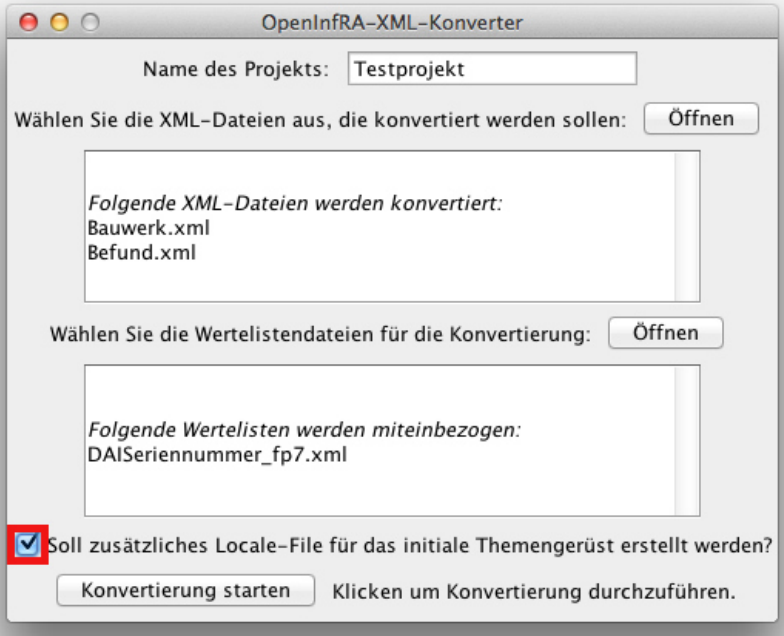
Danach müssen mit dem Tool die XML-Exporte der Tabellen aus Filemaker ausgewählt werden, die konvertiert werden sollen. Dazu wird der erste "Öffnen"-Button betätigt und danach im Filesystem die entsprechenden Dateien ausgewählt. **Wichtig:** Alle Tabellen müssen gleichzeitig migriert werden, damit die Verknüpfungen erstellt werden können!



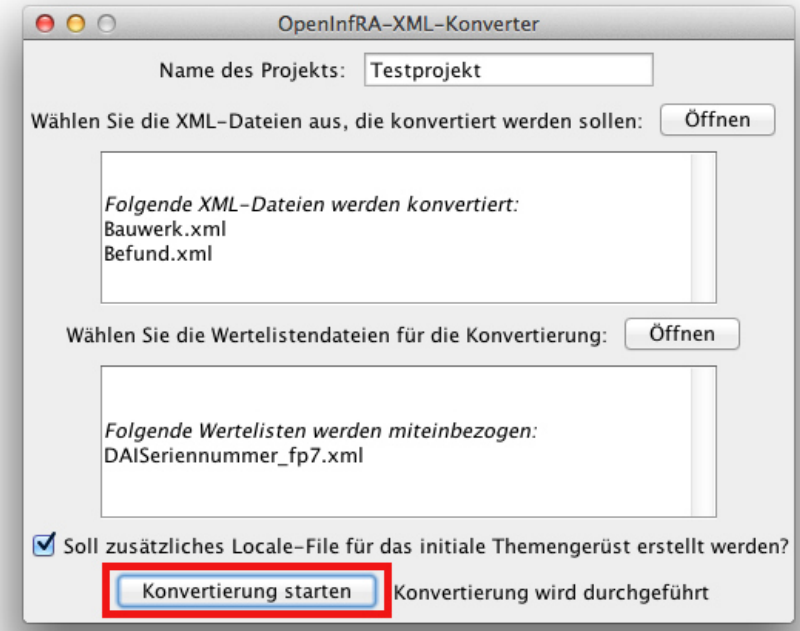
Im nächsten Schritt wird auf die gleiche Weise mit dem zweiten "Öffnen"-Button die Datei mit den Wertelisten ausgewählt, die durch den Database Design Report erstellt wurde.



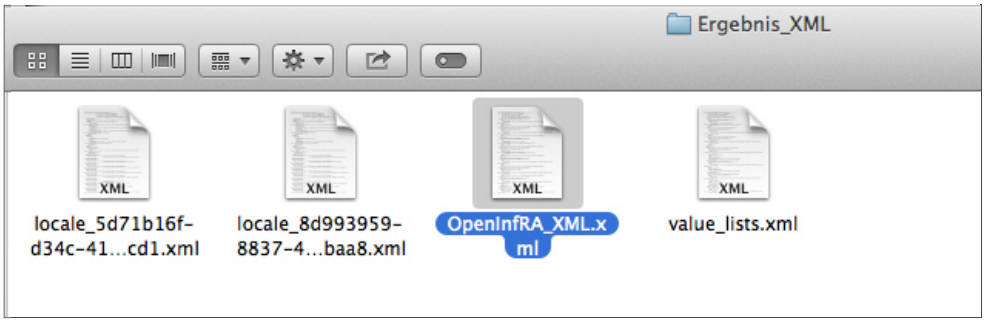
Falls man möchte, dass eine zusätzliche Localedatei für das initiale Themengerüst erstellt wird, sollte man den Haken dafür setzen. Dies ist nur dann erforderlich, wenn man ein initiales Themengerüst automatisiert erstellen möchte und die Datenbank vorher entsprechend bereinigt hat, s. Überlegungen zur automatischen Migration des initialen Themengerüsts



Um die Konvertierung in das OpenInfRA-XML-Format zu starten, muss jetzt "Konvertierung starten" angewählt werden. Die Konvertierung kann einige Zeit in Anspruch nehmen!



Im Ordner, in dem die Filemaker-Exporte liegen, entsteht ein neuer Ordner namens "valid", der die gleichen Exporte ohne illegale XML-Zeichen enthält. In diesem wiederrum gibt es einen Ordner namens "Ergebnis_XML", der die resultierenden konvertierten Dateien im OpenInfRA-XML-Format enthält. Diese können nun über die XML-Schnittstelle migriert werden!



Migration über die XML-Schnittstelle

Mithilfe der XML-Schnittstelle können die XML-Datei jetzt nach SQL konvertiert werden, auch dies funktioniert mithilfe von XSLT.

Als Erstes müssen die soeben erstellten Dateien aus dem Ordner "Ergebnis_XML" innerhalb der Ordnerstruktur der XML-Schnittstelle in den Ordner "DB_Import/XML_Dokumente" kopiert werden.
Je nach Betriebssystem muss jetzt entweder "import.bat" (Windows) oder "generate_SQL_only.sh" (Mac/Linux) ausgeführt werden.
"import.bat" kann die konvertierten Dateien direkt in eine lokale PostgreSQL-Datenbank einlesen, kann aber auch vorher abgebrochen werden. Das Bash-Skript kann lediglich konvertieren, in die Datenbank muss dann händisch eingelesen werden.
Die SQL-Dateien befinden sich dann im Ordner "SQL_Script". Beim händischen Import in PostgreSQL ist folgendes zu beachten:

- Es muss bereits eine Projektdatenbank mit dem im Script angegebenen Schemanamen existieren
  - Installation s. hier
  - Kurzfassung: Zuerst diese Datei (project_schema.sql) einlesen
  - dann diese (project_constraints.sql)
  - Schemanamen beachten!
- In diese Projektdatenbank müssen die statischen Wertelisten eingelesen werden (**Achtung:** Schemanamen anpassen!)
- Beim eigentlichen Import muss folgende Reihenfolge eingehalten werden: Zunächste die locale-Dateien, dann Import_wertelisten.sql, dann Import_instanz.sql. (**Achtung**: Einlesen muss über psql erfolgen, s.u.)

Probleme (und Lösungen)

Statische Wertelisten (d.h. die statischen UUIDs nicht nur für Wertelisten) werden von der XML-Schnittstelle nicht unterstützt:
Das ist vor allem ein Problem für die locale-Dateien, denn es gibt feste UUIDs für character_code, language_code und country_code, die nicht beachtet werden, stattdessen generiert die Schnittstelle eine Zufalls-UUID. Außerdem kommt der character_code UTF8 für jede locale-Datei extra vor, was nicht mit den Integritätsbedingungen überein geht.

Lösung: Wenn vorher die statischen Wertelisten eingelesen wurden, muss der Import der Dateien aus "SQL_Script" über psql/die Kommandozeile erfolgen, nicht über pgadmin. Der Grund dafür ist, das aufgrund doppelt verwendeter UUIDs Fehler auftreten werden, die bei pgadmin zu einem Abbruch führen. Psql kann den Import trotz der Fehler fortsetzen. Im Normalfall sehen die Befehle in etwa so aus:

```
psql  -f Import_locale_bd0fcf55-547a-44da-b7c7-915e6e4ec40e.sql -U benutzername -o log.txt datenbankname
psql  -f Import_locale_c0d76ff3-a711-42af-920d-09132a287015.sql -U benutzername -o log.txt datenbankname
psql  -f Import_wertelisten.sql -U benutzername -o log.txt datenbankname
psql  -f Import_instanz.sql -U benutzername -o log.txt datenbankname
```

Achtung: In der Windows-Eingabeaufforderung kann es zu Zeichensatzproblemen beim Import von UTF-8-Daten kommen. Um dies zu verhindern, müssen hier vor dem Import folgende Befehle eingegeben werden:

```
SET PGCLIENTENCODING=utf-8
chcp 65001
```

In OpenInfRA soll es möglichst vermieden werden, dass Strings innerhalb der gleichen Locale mehrfach vorkommen. In iDAIfield kommen aber sehr viele Strings mehrfach vor, die beim Import auch erstmal so in die Datenbank geschrieben werden.
Ein weiteres Problem der iDAIfield-Daten ist daher geschuldet, dass der von Filemaker angegebene Datentyp nicht verlässlich ist, d.h. Filemaker unterscheidet zwar zwischen TEXT, NUMBER und DATE, in NUMBER-Feldern kommen aber teilweise auch andere Zeichen als Zahlen vor, z.B. "B 622-287, 024". Solche Felder können logischerweise in OpenInfRA nicht als integer importiert werden, bei der XSL-Transformation werden daher alle Felder als Text behandelt.

Der Lösungsweg bei beiden Problemen ist eine **nachträgliche Bereinigung der Daten.**
Dies funktioniert via PL/pgSQL. In der Datei Pergamon Datenbereinigung.sql gibt es zwei Hauptfunktionen, die beide mehrfach verwendete Strings löschen. Außerdem wird für jeden Attributtyp im Nachhinein auf Basis der Daten der richtige Datentyp ermittelt.

- clean_up_data_system(varchar, varchar, varchar) ist dann zu verwenden, wenn es bereits eine gefüllte Systemdatenbank auf dem gleichen Rechner gibt, diese wird dann als Basis für die Dublettenlöschung genommen, d.h. die UUIDs aus der Systemdatenbank haben auf jeden Fall über allen anderen Priorität. Die Funktion erwartet den Namen der Systemdatenbank, den User und das Passwort als Argumente. (Diskussion hierzu in #1383)
- clean_up_data_static() funktioniert ohne Systemdatenbank, hier wird eine temporäre LCS-Tabelle mit statischem Inhalt (s. Datei) zur Grundlage des Dublettenlöschens verwendet.

Nach erfolgreicher Migration der Daten in die PostgreSQL-Datenbank sollte eine der beiden Funktionen durchgeführt werden.

Das initiale Themengerüst: Fehlt in dem hier beschriebenen Migrationsweg. Hier muss noch ein Workflow überlegt werden.
Überlegungen zur automatischen Migration des initialen Themengerüsts

3.2.5 Beziehungen

Bei der Migration wird nicht zwischen "richtigen" Tabellen und Kreuztabellen unterschieden, jede zu migrierende Datei wird zu einer neuen Themenausprägung.
Verknüpfungen werden auf Basis von Primär- und Fremdschlüsseln erstellt. Primärschlüssel müssen immer PS_TabellennameID heißen. Fremdschlüssel können sowohl FS_TabellennameID als auch Tabellenname::PS_TabellennameID (bei Bezugstabellen) heißen, beide Versionen werden erkannt.
Damit eine Verknüpfung zu Stande kommt, muss für jedes Fremdschlüssel-Feld eine Tabelle exisitieren, die einen gleichnamigen Primärschlüssel hat (und in der ein Datensatz mit der entsprechenden ID vorkommt).
Um Felder einer Kreuztabelle zu berücksichtigen, können diese beim Export einer Tabelle aus Filemaker als Felder einer Bezugstabelle mitaufgenommen werden.
Es gibt in iDAI.field keine explizite Beziehungssemantik wie bei OpenInfRA. Bei der Migration werden daher einfach alle Beziehungen dem Beziehungstyp "Eng" zugeordnet. Eventuell müssen die Beziehungstypen später über die GUI nachbearbeitet werden. (Um z.B. auszudrücken, dass eine Verknüpfung zwischen Bauwerk und Raum bedeutet, dass sich der Raum in dem Bauwerk befindet.)

3.2.6 Fehlende Daten aus iDAIfield

Bei der Migration von iDAIfield nach OpenInfRA werden aktuell noch nicht alle Möglichkeiten des Datenbankformates ausgenutzt. Das liegt größtenteils daran, dass bestimmte Informationen in iDAIfield einfach nicht bzw. nur implizit enthalten sind und daher nicht automatisch migriert werden können. Aktuell gibt es folgende Lücken:

- Keine Beziehungstypensemantik
- Attributtypgruppen (Müssen händisch nachgetragen werden, dies soll auch in Zukunft so bleiben und später über die GUI gemacht werden.)
- Beziehungen zwischen Attributtypen
- Beziehungen zwischen Wertelisten (und Wertelistenwerten, diese könnten aber im Rahmen des init. Themengerüsts nachgetragen werden)
- Geometrien
- Mehrsprachigkeit (Existiert teilweise in Wertelisten-Daten von S. Kruse)

1.png - Bild 1 - Export aus Filemaker (156,454 KB) Karen Schwane, 28.08.2014 12:02
4.png - Bild 3 - Grammatik "FMPXMLRESULT" (44,091 KB) Karen Schwane, 28.08.2014 12:02
3.png - Bild 2 - XML als Format (100,018 KB) Karen Schwane, 28.08.2014 12:02
5.png - Bild 4 - Alle benötigten Felder (118,956 KB) Karen Schwane, 28.08.2014 12:02
Bildschirmfoto 2014-09-22 um 15.40.02.png (43,204 KB) Karen Schwane, 22.09.2014 15:41
Bildschirmfoto 2014-09-22 um 15.50.07.png (188,855 KB) Karen Schwane, 22.09.2014 15:58
Bildschirmfoto 2014-09-22 um 15.49.47.png (559,39 KB) Karen Schwane, 22.09.2014 15:58
Bildschirmfoto 2014-12-10 um 11.30.48.png (43,007 KB) Karen Schwane, 10.12.2014 11:45

Bildschirmfoto 2014-12-10 um 11.32.13.png (103,688 KB) Karen Schwane, 10.12.2014 11:45
Bildschirmfoto 2014-12-10 um 11.33.13.png (53,236 KB) Karen Schwane, 10.12.2014 11:45
Bildschirmfoto 2014-12-10 um 11.32.49.png (111,946 KB) Karen Schwane, 10.12.2014 11:45
Bildschirmfoto 2014-12-10 um 11.33.28.png (52,552 KB) Karen Schwane, 10.12.2014 11:45