SHORE

1.1.0

Administrator-Handbuch

Dokumentversion 1.1

Norma Korta, Klaus Mayr, Tammo Schnieder,
Helge Schulz

sd&m AG
software design & management
Thomas-Dehler-Strasse 27
81773 München
(089) 6 38 12-0
(089) 6 38 12-150

www.sdm.de

SHORE Administratorhandbuch

Dieses Handbuch wird vom sd&m-Werkzeugteam gepflegt (mailto: V_P_SHORE@muc.sdm.de)

Das Verfahren der Verarbeitung von SHORE ist von der sd&m AG zum Patent angemeldet.

Der Name SHORE ist geschützt.

Zusammenfassung

Dies ist das Handbuch für Administratoren von SHORE (sd&m Hypertext Object Repository). Von der Installation und Einrichtung über das Systemmanagement bis zu routinemäßigen Tätigkeiten findet sich hier alles, was zum Betrieb von SHORE nötig ist.

Historie

Version	Status	Datum	Autor(en)	Erläuterung
1.1	in Arbeit	17.September 2001	Tammo Schnieder	ImportBatch-Kapitel erweitert, auf neue Parser-Verzeichnisstruktur umgestellt
1.0	freigegeben	03.Juli 2001	Tammo Schnieder	

Inhaltsverzeichnis

1 Einleitung

SHORE ist ein leistungsfähiges System, um strukturierte Projektdokumente und deren Beziehungen zu verwalten und zu präsentieren.

Dieses Handbuch hat die Aufgabe, Sie als SHORE-Administrator bei Ihrer Arbeit zu unterstützen, um SHORE möglichst effizient und reibungslos einsetzen zu können.

Dieses Dokument beschreibt, wie Sie SHORE installieren und welche Einstellungen Sie an SHORE vornehmen können. Sie bekommen Informationen zum Managen eines SHORE-Systems und Sie finden in diesem Handbuch typische Abläufe.

Das SHORE zugrundeliegende Konzept ist detailliert im SHORE-Anwenderhandbuch beschrieben.

Stilelemente dieses Handbuchs

Einige Teile des Textes heben sich optisch ab. Dies sind zum einen Anleitungen, die grau hinterlegt sind. Weitere verwendete Konventionen:

Konvention	Bedeutung
Schriftarten	Einige Textteile sind in eigenen Schriftarten dargestellt, beispielsweise: ● `Dateien und Pfadangaben` ● `Programmcode`
">" (Größerzeichen)	Das Größerzeichen wird verwendet, wenn Sie in einem Menü nacheinander verschiedene Einträge auswählen sollen. Beispiel: „Hilfe > Hilfethemen...“

Symbole dieses Handbuchs

Folgende Symbole werden in diesem Handbuch verwendet:

Achtung: Wenn Sie etwas besonders beachten sollten.

Hinweis: Weiterführender oder erläuternder Text
.

Tipp: Falls Ihnen etwas die Arbeit erleichtern kann oder eine andere Information an dieser Stelle nützlich erscheint.

Generell: die Verwendung der männlichen Form gilt für Männer und Frauen.

1.1 Anforderungen und Hilfe für den Administrator

1.1.1 Vorkenntnisse

Um SHORE effizient administrieren zu können, sollten Sie über folgende Kenntnisse verfügen:

Thema	vorausgesetzter Kenntnisgrad
allgemeine NT-Kenntnisse als Anwender	mittel, d.h. Sie sollten wissen, dass es verschiedene Berechtigungsstufen gibt und wozu sie gut sind. Sie sollten wissen, wo Sie Einstellungen zu Ihrer NT-Installation finden können.
Umgang mit der NT-Eingabeaufforderung (DOS-Fenster), NT-Befehle	mittel bis hoch, d.h. Sie sollten wissen, wie man sich in Verzeichnissen zurechtfindet und wie man Programme ausführt. Von Vorteil ist es, wenn Sie wissen, wie man eine Batchdatei erstellt.
Installation von Programmen unter NT	mittel, d.h. Sie sollten schon einige Programme unter NT installiert haben und wissen, was die Registry ist und sich zutrauen, dort evtl. Änderungen vorzunehmen.
Prolog	Grundkenntnisse (optional), um SHORE effizient zu nutzen, werden Sie Anfragen formulieren.

1.1.2 Aufgaben

Ein reibungsloser Einsatz von SHORE erfordert einige Aufgaben, die gelöst werden müssen.

projektorganisatorische Aufgaben

Wie für jeden gut laufenden Server benötigen Sie auch für die Betreuung eines SHORE-Servers einen Administrator und einen Stellvertreter.

Als SHORE-Administrator sind Sie die erste Anlaufstelle für Fragen aus dem Projektteam. Sie leisten somit "1st-Level-Support". Wenn Sie Fragen nicht beantworten können, können Sie sich an das Werkzeugteam wenden.

SHORE-organisatorische Aufgaben

Für die Pflege Ihres SHORE-Systems fallen folgende Aufgaben an:

- SHORE und SHORE-Updates installieren

- Mit dem Werkzeugteam zusammenarbeiten
  Als SHORE-Administrator ist es vorteilhaft, wenn Sie eng mit dem Werkzeugteam zusammenarbeiten. Das Werkzeugteam bietet Leistungen für Sie an, wie die Beantwortung von Fragen, Beratung, Fehlerbehebung und Weiterentwicklung von SHORE. Die Art der Zusammenarbeit können Sie mit dem Werkzeugteam absprechen.
- Erweiterungen an SHORE fließen zum Werkzeugteam zurück.
  Wenn Sie für SHORE Erweiterungen vornehmen möchten - beispielsweise Anpassungen an einem Parser oder Schreiben eines Skriptes zur Automatisierung des Imports - so sprechen Sie sich idealerweise mit dem Werkzeugteam ab. Das Werkzeugteam koordiniert sämtliche Weiterentwicklungen von SHORE, so können Doppel- bzw. Parallelentwicklungen vermieden oder auch Unterstützung gegeben werden.

technische Aufgaben

Damit SHORE auf Ihrem Server aus technischer Sicht reibungslos läuft, führen Sie als Administrator im wesentlichen folgende Arbeiten aus:

- Regelmäßige Kontrolle von Logdateien
  Kontrollieren Sie in regelmäßigen Abständen die Logdateien auf Fehlermeldungen und löschen Sie alte Dateien aus dem entsprechendem Verzeichnis.
  Anfangs möglichst nach jedem Import!
- Plattenplatz beobachten
  Achten Sie darauf, dass auf den Platten des SHORE-Servers genügend Platz vorhanden ist, da die Menge der von SHORE verwalteten Dokumente häufig schnell wächst.
- Systemstörungen erkennen und beheben, Bedienfehler vermeiden
  Machen Sie sich mit dem SHORE-Server vertraut, damit Sie Systemstörungen frühzeitig erkennen und beseitigen können.

1.1.3 Treffen der SHORE Anwender (SURF)

Das Werkzeugteam veranstaltet in unregelmäßigen Abständen "SHORE User Relation Forums", kurz SURF-Meetings. Dort erfahren Sie Neuigkeiten, hören Erfahrungsberichte und können sich mit anderen Anwendern von SHORE austauschen. Den nächsten Zeitpunkt eines SURF-Meetings erfahren Sie vom Werkzeugteam.

1.1.4 Fehler und Erweiterungen

Damit Sie mit dem Werkzeugteam optimal zusammenarbeiten können, haben wir ein Change Request-Verfahren installiert. Offene Punkte, gewünschte Erweiterungen und Fehler in Bezug auf SHORE können Sie direkt einsehen, wenn Sie Zugang zum sd&m-weiten Intranet (sww) haben.

Sie finden dazu Informationen auf der SHORE-Projekt-Homepage `http://sww.sdm.de/prj/shore` unter "Requests".

1.1.5 Projektspezifische Erweiterungen durch das Werkzeugteam

Wenn Sie für Ihr Projekt umfangreichere Anpassungen rund um SHORE vornehmen möchten, oder wenn Sie Unterstützung benötigen, wenden Sie sich bitte an das Werkzeugteam. Auf der Homepage des Werkzeugteams finden Sie weitere Details zum Einsatz des Werkzeugteams in Projekten. Per mail können Sie das Werkzeugteam unter der Adresse V_P_SHORE@muc.sdm.de erreichen.

2 Konzept

Damit Sie SHORE besser kennenlernen, können Sie anschließend lesen, aus welcher Idee heraus SHORE geboren wurde und mit welcher Technik es realisiert wurde. Dieses Wissen benötigen Sie, wenn Sie Änderungen an Ihrem SHORE-System vornehmen möchten.

2.1 Die Idee von SHORE

Was ist SHORE?

SHORE ist ein Repository für Projektdokumente.
SHORE ist offen für Projektdokumente beliebigen Typs, die inhaltlich in einem Zusammenhang stehen, wie beispielsweise die Ergebnisdokumente und Programme, die in einem Software-Entwicklungsprojekt anfallen. SHORE verknüpft in einem Hypertextsystem die unterschiedlichsten Dokumente, die die unterschiedlichsten Datenformate und Informationsinhalte beinhalten können.

Wofür dient SHORE?

Der Inhalt eines Dokumentes wird von SHORE in Objekte zerlegt, die untereinander in Beziehungen stehen. Die Dokumente, ihre Objekte und Beziehungen werden mit SHORE verwaltet und stehen für Navigation und Recherchen zu Verfügung. Die Inhalte unterschiedlichster Dokumente können auf diese Weise miteinander verknüpft werden.

Welche Typen von Dokumenten eigenen sich für SHORE

Die Palette von Dokumenttypen, die sich nach SHORE importieren lassen, reicht von schwach strukturierten Textdokumenten - wie Anforderungen, Testfälle, Fehler- und Änderungsprotokolle, Konzepte und Entwürfe (als Instanzen einer klar geregelten Spezifikationssprache) - bis hin zu streng formalisierten Dokumenten - wie Schnittstellenbeschreibungen und Programm-Quelltexte mit klar definierter (und allerdings manchmal komplizierter) Syntax. Mit SHORE können Sie also alle Phasen eines Entwicklungsprojektes abdecken.

Welche Rollen gibt es aus Sicht von SHORE?

Zunächst gibt es den **Administrator**. Seine Aufgaben werden in diesem Handbuch beschrieben. Der SHORE-Administrator kümmert sich um die korrekten projektspezifischen Einstellungen des SHORE-Systems. Dazu gehört z.B. das Verwalten von dokumenttypspezifischen Parsern und projektspezifischen Anfragen an die SHORE-Datenbasis. Der Administrator kümmert sich außerdem darum, dass neue Projektdokumente in das SHORE-System gestellt werden.

Der Administrator stellt ein SHORE-System bereit, dass die sogenannten **Surfer** nutzen können. Die Surfer navigieren durch die Dokumente und stellen Anfragen um bestimmte Fragestellungen anhand der Objekte und Beziehungen beantworten zu können. Surfer sind z.B. Mitarbeiter der auftraggebenden Fachabteilung oder QS-Beauftragte.

Je nach Projektorganisation existiert gegebenenfalls die Rolle eines **Entwicklers**. Dies ist ein Benutzer, der zusätzlich zu den Interaktionen des Surfers auch Dokumente importiert. sd&m-Mitarbeiter in Projekten sind typische Entwickler im SHORE-Sinne. Oftmals wird der Import von Dokumenten vom Administrator organisiert, so dass keine einzelnen Importe notwendig oder erlaubt sind.

2.2 technische Aspekte

Grundsätzliche Architektur

SHORE ist als Client-Server-System konzipiert. Die Serverkomponente läuft als Programm oder Dienst auf einem NT-Server. Der Server speichert die Projektdokummente und die dazugehörigen Objekte und Beziehungen. Der Administrator steuert das System mit dem Kommandozeilen-Client. Dies ist ein Java-Programm, das in einem DOS-Fenster bedient wird.

Der Client für die Surfer läuft als Java-Applet in einem Browser.

Wie ist SHORE in eine Software-Entwicklungsumgebung eingebunden?

SHORE benötigt keine spezielle Einbindung in die Entwicklungsumgebung. Dank des Kommandozeilen-Clients kann der Im- und Export von Dokumenten flexibel gesteuert werden. Z.B. kann der Import über ein Skript mit dem Konfigurationsmanagement des Projekts gekoppelt werden, so dass bei der Freigabe einer Dokumentversion (check in) immer ein Import erfolgt.

Für spätere Stufen ist eine engere Kopplung mit der Entwicklungsumgebung vorstellbar, so dass z.B. Build-Vorgänge und die Steuerung des Konfigurationsmanagements aus der SHORE Oberfläche heraus erfolgen.

Wie gelangen Ihre Dokumente nach SHORE?

Um die unterschiedlichen Dokumente in ein einheitliches Datenformat zu bekommen, werden sie von Parsern in das universelle Dokumentformat XML (eXtensible Markup Language) konvertiert und auf die für eine Hypertext-Navigation relevanten Objekte und Beziehungen hin analysiert.

Wie werden Objekte und Beziehungen in den Dokumenten erkannt?

Software-Entwickler interessieren sich für Strukturen und Zusammenhänge. Ein Erkennen dieser Zusammenhänge ist nicht immer leicht. Um Querbeziehungen so genau wie möglich zu erkennen, müssen komplexe Parser für die betreffenden Dokumente entwickelt werden, die eine auf Symboltabelleninformation gestützte semantische Analyse durchführen. Nur so können Namenskonflikte aufgelöst und die richtige Definitionsstelle eines Objektes, von dem man nur einen Teil seines Namen kennt, gefunden werden.

Es stehen verschiedene Parser bereit: Java, Cobol und C-Quelltexte können komplett geparst werden. XML-Konvertierung von Rational-Rose-Dokumenten, Word- und Excel-Dokumenten, Access-Tabellen existieren ebenfalls. Darüber hinaus gibt es Generatoren, die Sie in die Lage versetzen, sich einfache Parser auf Basis von Pattern-Matching selbst zu schreiben.

Wie sind Objekte und Beziehungen abgelegt?

Die in einem Dokument von einem Parser erkannten Objekte und Beziehungen werden mit XML-Tags markiert. Beim SHORE-Import werden die Objekte und Beziehungen später wieder extrahiert und dann in einer Datenbank abgelegt. Das Datenbankschema ist bei SHORE im Gegensatz zu anderen Repositories frei definierbar. Der Anwender kann in seinem Metamodell festlegen, welche Objekte und Beziehungen er in SHORE sehen möchte. Jedes Projekt kann die Struktur seines Metamodells frei gestalten, und entsprechende Parser nach seinen eigenen Bedürfnissen entwickeln.

Wie präsentiert SHORE die Dokumente, Objekte und Beziehungen?

Alle Dokumente, die nach SHORE importiert wurden, erscheinen direkt als Hypertext im Browser des Anwenders. Der HTML-Hypertext wird dynamisch aus jedem XML-Dokument generiert. Bei dieser Umwandlung werden die Einträge aus der Datenbank, Objekte(rot) und Beziehungen(blau) farblich ausgezeichnet und miteinander verlinkt, damit eine Hypertext-Navigation über sie möglich ist.

Mit jedem Klick auf eine Hypertext-Beziehung (Link) wird intern eine Anfrage an die SHORE-Datenbank gerichtet, die ihrerseits den Browser dazu veranlasst, das Dokument mit dem gesuchten Ziel zu präsentieren. Aufgrund der Tatsache, dass die Links in einer Datenbank gespeichert werden, ist damit sogar die Navigation in die umgekehrte Richtung möglich.

Welche Abfragen sind möglich?

Durch die Anbindung der logischen Programmiersprache Prolog haben Sie außerdem die Möglichkeit, über Navigations-Anfragen hinaus weitere komplexe Abfragen an die Datenbank zu stellen, deren Ergebnisse dann wieder dynamisch in HTML-Seiten konvertiert werden.

Besonders hervorzuheben ist, dass SHORE auch rekursive Abfragen erlaubt. Damit ist man in der Lage, zum Beispiel baumartige Strukturen aus dem Beziehungsnetz zu extrahieren.

Typische komplexe Abfragen, die häufig gewünscht werden und in verschiedenen Projekten bereits häufig verwendet wurden, sind:

- wie sehen die Aufrufhierarchien aus?
- wie sehen die Vererbungshierarchien aus?
- wo gibt es nicht verwendete Objekte?
- wo gibt es Objekte, die mehrfach definiert sind?
- wie sieht die Nachbarschaft eines Objekts aus ... eingeschränkt auf ...?
- usw.

Zur Formulierung solcher Anfragen benötigt man eine genaue Kenntnis des Metamodells. Denn dazu muss man dem System sagen, von welchen Objekten welchen Objekttyps es über welche Beziehungen welchen Beziehungstyps bei der Suche navigieren soll. Je konkreter / allgemeiner (in der Vererbungshierarchie) hierbei die angegebenen Typen sind, desto präziser / umfassender sind auch die Antworten, die SHORE bei diesen Abfragen zurückliefert.

Was bietet mir die Kopplung mit einer Suchmaschine?

Sie können über Ihre Dokumente eine Volltextsuche durchführen, wenn Sie zusätzlich zu SHORE eine Suchmaschine installieren. Es wird momentan das Produkt Information Server search 97 Version 3.6.1 der Firma verity unterstützt.

3 Installation und Einrichtung

Sie haben sich dafür entschieden, SHORE zu installieren. Zuerst sollten Sie prüfen, ob die für SHORE notwendigen Programme zur Verfügung stehen. Dazu gehört,

- dass Sie eine Java-Laufzeitumgebung der Version 1 (ab 1.1.5) oder 2 (ab 1.2.2) installiert haben, die sich über ein Programm java.exe starten läßt (wird vom Kommandozeilen-Client benötigt, der von SHORE installiert wird). Im Pfad zur Java-Laufzeitumgebung dürfen keine Leerzeichen enthalten sein (z.B. wie in "`Program Files`")
- dass Sie einen Browser ab Netscape 4.5 oder Internet Explorer 5.0 zur Verfügung haben, in dem Java ausgeführt werden kann.

Alle Installationen müssen Sie unter Windows NT als Benutzer mit Administrator-Rechten vornehmen.

3.1 Wie eine Installation abläuft

Damit Sie einen Überblick gewinnen, lesen Sie vor der eigentlichen Installation zunächst, was zur Installation von SHORE dazugehört. Die Installation ist in drei Phasen aufgeteilt:

1. Programme installieren
2. projektspezifische Anpassungen planen
3. SHORE-Instanz(en) einrichten

3.1.1 Phase 1: Programme installieren

In dieser Phase installieren Sie die für SHORE benötigte Software.
Zunächst ein kurzer Überblick als Ablaufdiagramm:

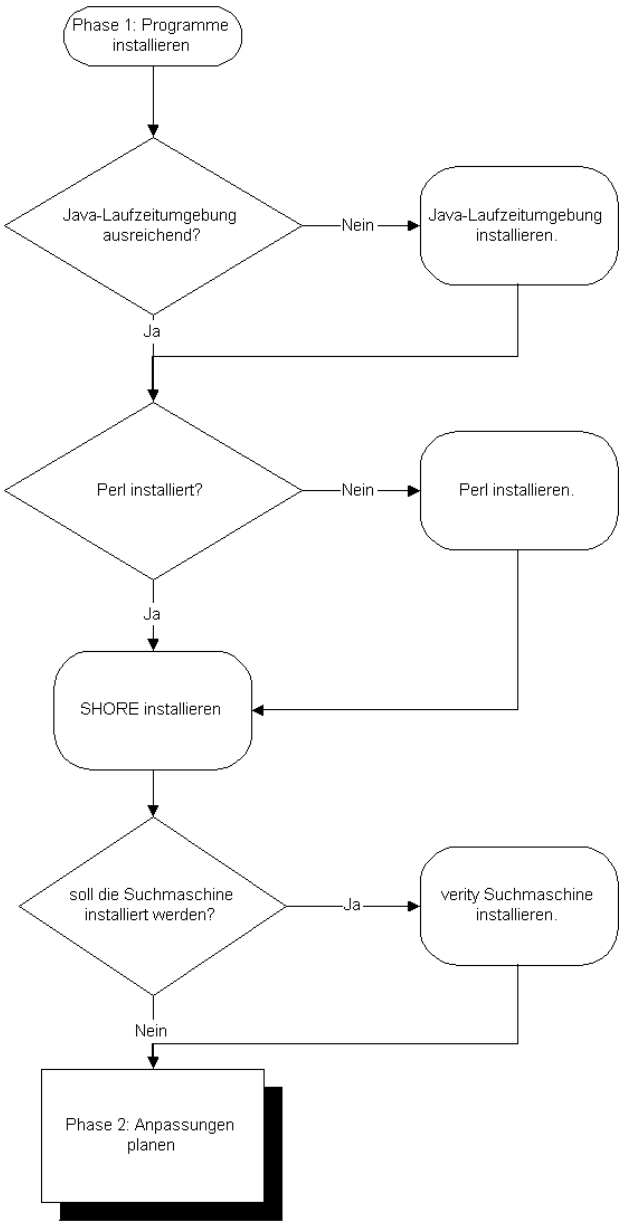


Abbildung 1: Schritte der Phase 1

Zur Installation von SHORE auf einem Windows NT 4.0 Server gehören folgende Schritte:

1. Prüfen Sie, ob Sie Perl ab Version 5 installiert haben. Falls nicht, installieren Sie Perl.
2. Prüfen Sie, ob eine geeignete Java Laufzeitumgebung existiert. Falls nicht, müssen Sie eine installieren und einrichten. Details dazu finden sich im- Abschnitt “Installation der Java-Laufzeitumgebung” auf Seite 16.
3. SHORE installieren. Details hierzu: Abschnitt “Installation von SHORE” auf Seite 17
4. Wenn Sie eine Lizenz der verity Suchmaschine besitzen, können Sie diese anschließend installieren und einrichten. Siehe “Installation der verity-Suchmaschine” auf Seite 19

Tipp: Für die spätere Arbeit mit SHORE empfiehlt es sich, auf dem Arbeitsplatzrechner des Administrators ein Laufwerk mit dem SHORE-Verzeichnis zu verbinden um z.B. Logdateien zu kontrollieren. Außerdem ist in manchen Situationen die Installation eines "Fernsteuertools" wie z.B. VNC oder pcAnywhere hilfreich.

3.1.2 Phase 2: Instanz einrichten

Nun geht es darum, dass Sie sich Gedanken machen, wie SHORE für Sie angepasst werden soll.

Wenn Sie mehrere Instanzen von SHORE laufen lassen möchten, dann müssen Sie für jede Instanz zumindest einen Datenbanknamen und eine Portnummer festlegen.

Weitere Details finden Sie in Abschnitt “So planen Sie die Einrichtung einer neuen Instanz” auf Seite 63

Jede einzelne Instanz richten Sie wie folgt ein:

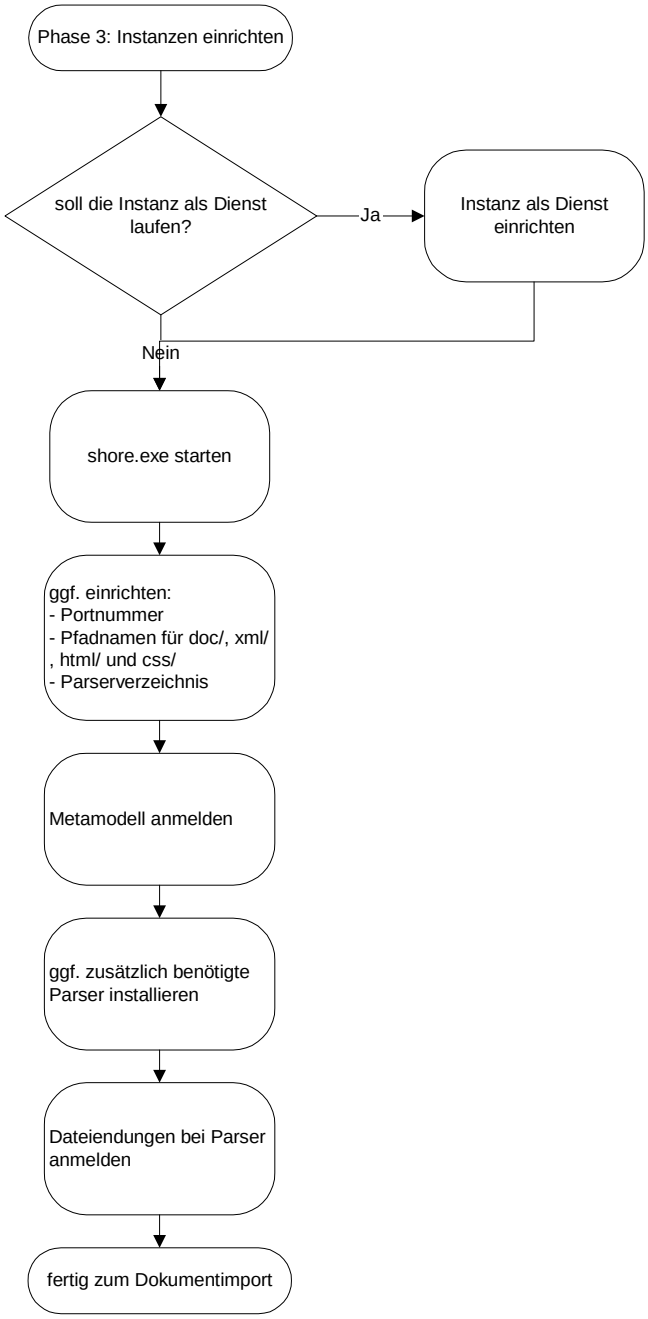


Abbildung 2: Phase 2

1. Wenn die Instanz als Dienst laufen soll, dann richten Sie sie ein wie in Abschnitt “Einrichten als Dienst” auf Seite 55 beschrieben.
2. Starten Sie SHORE ggf. mit einem Datenbanknamen als Parameter (siehe “SHORE starten” auf Seite 71).
3. Ändern Sie die Portnummer und ändern Sie ggf. die Verzeichnisse (siehe “Einstellen der Projektdokumentablage” auf Seite 60).
4. Melden Sie das Metamodell an (siehe “So importieren Sie das Metamodell” auf Seite 28).
5. Falls Sie zusätzliche Parser für Ihr Projekt installieren wollen, dann tun Sie dies jetzt (siehe “Parser in SHORE installieren” auf Seite 46).

6. Melden Sie die Dateiendungen bei den entsprechenden Parsern an (siehe “So melden Sie den Parser für Ihre Dokumenttypen über das Browserfenster bei SHORE an.” auf Seite 47).

Nun haben Sie SHORE komplett eingerichtet und können Ihre Dokumente importieren. Dies wird in Kapitel “Projektdokumente” auf Seite 75 ausführlich diskutiert.

3.2 Hardware-Voraussetzungen

- Betriebssystem
  SHORE läuft unter Windows NT 4.0 Workstation oder Server der 386-Familie (ab Servicepack 4).
- Hauptspeicher
  Je nach eingesetztem Parser und Menge der Daten, die zu importieren sind, ist der Hauptspeicher zu dimensionieren. Mit 128 bis 256 MB kann ein SHORE-Server laufen. Bei einem umfangreichen Dokumentenbestand wird ein System mit 512 MB empfohlen.
- Festplatten
  Aus Performance-Gründen sollten Sie mindestens mit zwei physischen Festplatten arbeiten. Die zweite Platte sollte die Logfiles der ObjectStore-Datenbank speichern.

  Die Größe der Hauptplatte errechnet sich ungefähr wie folgt:
  - ca. 60 MB für das SHORE-System
  - plus die Größe der zu importierenden Dateien multipliziert mit 40 bis 50 (abhängig von den zu importierenden Dateitypen).

  Zusätzlich wird für die Logdateien Platz benötigt. In der Regel sind das einige Megabyte, aber zum umfangreichen Verfolgen von Fehlern sollten Sie ca. 100-200 MByte berücksichtigen.

3.3 Installation

3.3.1 Installation von Perl

Sie benötigen Perl Version 5.005 (oder neuer) auf dem Rechner, auf dem Sie Ihre Parser laufen lassen.

So testen Sie, ob ein ausreichendes Perl installiert ist

1. Geben Sie in der Eingabeaufforderung „`perl -v`“ ein.

Falls Perl nicht installiert ist, bekommen Sie die Meldung
`Der Befehl ist entweder falsch geschrieben oder konnte nicht gefunden werden.`
`Bitte überprüfen Sie die Schreibweise und die Umgebungsvariable 'PATH'.`
Falls Perl installiert ist, sehen Sie die installierte Versionsnummer. Wenn die Versionsnummer größer oder gleich 5.005 ist, kann SHORE damit arbeiten. Bei einer kleineren Version müssen Sie dieses Perl deinstallieren und eine neuere Version installieren.

So installieren Sie Perl 5.6

Auf der SHORE-CD finden Sie Perl Version 5.6, die Sie installieren können Stellen Sie sicher, dass auf dem Rechner kein Perl installiert ist. Für die Installation der Version 5.6 benötigen Sie den Windows Installer (eine Version für Windows NT 4.0 ab Service Pack 3 ist ebenfalls auf der CD enthalten).

1. Klicken Sie auf `ActivePerl-5_6_1_629-MSWin32-x86-multi-thread.msi` damit wird die Installation gestartet.
   Das Fenster „ActivePerl Build 629 Setup“ öffnet sich. Klicken Sie „Next“
2. Das Fenster „ActivePerl Build 629 License Agreement“ öffnet sich. Sie müssen die Lizenz akzeptieren um Perl installieren zu können. Dazu wählen Sie die Option „I accept the terms in the Licence Agreement“ und klikken auf „Next“
3. Das Fenster „ActivePerl Build 629 Setup“ mit dem Text "Custom Setup" zeigt eine Baumstruktur, in der Sie den Umfang der Installation und das Installationsverzeichnis wählen können.
   SHORE benötigt zumindest den Punkt "The Perl scripting language", sie können aber auch den Rest mit Installieren.
   Um das Installationsverzeichnis zu ändern, müssen Sie im Baum den Eintrag `ActivePerl` markieren. Klicken Sie auf „Browse“.

4. Das Fenster „ActivePerl Build 629 Setup“ mit dem Text "Change current destination folder" öffnet sich. Wählen Sie hier den Pfad aus, wo Perl installiert werden soll wie z.B. `D:\Programme\Perl`
   Klicken Sie auf „OK“ und das Fenster schließt sich.
5. Klicken Sie im Fenster „ActivePerl Build 629 Seetup“ mit dem Text "Custom Setup" auf „Next“
6. Das Fenster „ActivePerl Build 629 Setup“ zeigt nun den Inhalt mit der Überschrift "Choose Setup Options" an. Die beiden Optionen „Add Perl to the PATH environment variable“ und „Create Perl file extension association“ sollten ausgewählt sein.
   Klicken Sie auf „Next“.
7. Das Fenster „ActivePerl Build 629 Setup“ zeigt nun den Inhalt mit der Überschrift "Ready to Install" an. Klicken Sie auf „Install“ um mit der Installation zu beginnen.
8. Der Fortschritt der Installation wird angezeigt. Am Ende der Installation öffnet sich das Fenster „ActivePerl Build 629 Seetup“ mit dem Titel "Completing the ActivePerl Build 629 Setup Wizard". Sie können hier noch wählen, ob Sie die release notes sehen wollen.
   Klicken Sie auf „Finish“ um die Installation zu beenden.

Sie haben nun Perl 5.6 installiert.

3.3.2 Installation der Java-Laufzeitumgebung

Sie benötigen eine Java-Laufzeitumgebung auf dem Rechner, auf dem Sie mit dem SHORE-Kommandozeilen-Client arbeiten. Dies ist zunächst auf dem Server, auf dem Sie auch SHORE installieren werden. Sie können später noch weitere Kommandozeilen-Clients auf anderen Arbeitsplatzrechnern installieren. Den Kommandozeilen-Client benötigen Sie unter anderem, um SHORE einzustellen oder um Projektdokumente nach SHORE zu importieren (siehe “Betrieb eines SHORE-Servers” auf Seite 71).

Wenn Sie keine erforderliche Laufzeitumgebung auf Ihrem Rechner installiert haben, so können Sie die mitgelieferte Version 1.1.8 installieren.

So installieren Sie Java 1.1.8

1. Starten Sie von der CD im Verzeichnis `installation\jdk-1.1.8` die dort liegende ausführbare Datei und folgen Sie den Anweisungen.
2. Setzen Sie die Umgebungsvariable PATH, unter dem Menü Start > Einstellungen > Systemsteuerung > System > Register „Umgebung“.

Syntax <JAVA-Installationsverzeichnis>\bin
Beispiel d:\programme\jdk1.1.8\bin

Achtung: Das JAVA-Installationsverzeichnis darf keine Leerzeichen enthalten (z.B. wie "`Program Files`").

Hinweis: Schreiben Sie diesen Eintrag abgetrennt durch ein Semikolon hinter die anderen Einträge der PATH-Variable.

3. Setzen Sie die Umgebungsvariable CLASSPATH

Syntax <JAVA-Installationsverzeichnis>\lib\classes.zip
Beispiel d:\programme\jdk1.1.8\lib\classes.zip

Besondere Hinweise:

Achtung: Höhere Versionen als 1.1.8 benutzen Swing-Klassen, die auch im CLASSPATH referenziert werden. Möchten Sie eine höhere Version verwenden, müssen Sie sich ein Skript schreiben, welches den CLASSPATH neu setzt.

Tipp: Falls Sie schon eine ältere Version von Java auf dem Server-Rechner installiert hatten, und für andere Anwendungen weiternutzen wollen, so können Sie sich für die alte Version eine CMD- oder BAT-Datei schreiben, in der PATH und CLASSPATH entsprechend auf die alte Version gesetzt wird.

3.3.3 Installation von SHORE

Melden Sie sich auf dem Rechner, auf dem Sie SHORE installieren wollen mit einem Benutzer an, der mindestens lokale Administrationsrechte besitzt.
Bevor Sie das Installationsprogramm starten, empfehlen wir Ihnen, eine bereits installierte Version von SHORE und ObjectStore zu deinstallieren.

Hinweis: Versionen vor 1.0.6 müssen manuell gelöscht werden, da Sie nicht durch Installshield installiert wurden, sondern durch Kopieren der Dateien.

Hinweis: Wenn Sie eine vorhandene Version von SHORE nicht vorher deinstallieren, installiert das Installshield SHORE zwar korrekt, bei einer anschließenden Deinstallation wird jedoch das ObjectStore dann nicht mehr gelöscht. Daher unsere Empfehlung, eine alte Version von SHORE vorher zu deinstallieren.

Prüfen Sie, ob eine Java-Laufzeitumgebung installiert ist, denn Sie ist Voraussetzung für die Installation von SHORE.

So installieren Sie SHORE

1. Starten Sie die Installation von SHORE mit der Datei `Setup.exe`.
   Sie finden die Datei auf der SHORE-Installations-CD im Verzeichnis `installation\shore-<Versionsnummer>`.

Hinweis: Wenn auf Ihrem System ObjectStore bereits installiert ist, dann beginnt Ihr Installationsprogramm mit dem Fenster „SHORE Installationspfad“ (siehe unten).

Sie werden vom Fenster „Willkommen“ begrüßt. Klicken Sie auf „Weiter“.

2. Stellen Sie im Fenster „ObjectStore Installationspfad“ den Zielordner ein.
   Der Pfad und der Zielordner dürfen keine Leerzeichen und keine Sonderzeichen enthalten (z.B. wie in "`Program Files`").

Syntax <Laufwerk>:\<Verzeichnis>.
Beispiel `D:\Programme\ostore`.
Klicken Sie auf „Weiter“.

3. Stellen Sie im Fenster „Logdateipfad“ den Zielordner ein, wohin die Logdateien geschrieben werden sollen.

Hinweis: Wie schon früher erwähnt, wählen Sie aus Geschwindigkeitsgründen ein Verzeichnis auf einem anderen Laufwerk aus. Dieses Laufwerk sollte sich auf einer zweiten physischen Festplatte befinden.

Tipp: Sie können im Fenster „Ordner auswählen“ ein neues Verzeichnis erstellen, indem Sie es im Feld „Pfad“ eingeben und mit „OK“ bestätigen.

Klicken Sie auf „Weiter“.

4. Im Fenster „SHORE Installationspfad“ stellen Sie den Zielordner ein.
   Der Pfad und der Zielordner dürfen keine Leerzeichen und keine Sonderzeichen enthalten.

Syntax <Festplatten-Laufwerk mit Ordner>\shore
Beispiel `D:\Programme\shore`
Klicken Sie auf „Weiter“.

5. Im Fenster „SHORE Instanznamen festlegen“ geben Sie einen Instanznamen ein. Dieser Name darf keine Leerzeichen enthalten.
   Klicken Sie auf „Weiter“.
   Die Dateien werden kopiert. Dabei öffnen und schließen sich einige DOS-Fenster.
6. Wählen Sie im Fenster „Setup abgeschlossen“ aus, ob der SHORE-Server und die SHORE-Oberfläche im Browser sofort gestartet werden sollen oder nicht und bestätigen Sie mit „Beenden“

Das Installationsprogramm von SHORE ist damit beendet.

Hinweis: In SHORE wird die lizenzpflichtige Datenbank ObjectStore verwendet. Für den Einsatz in sd&m-Projekten fallen keine zusätzlichen Lizenzgebühren an, bei einem Einsatz außerhalb sprechen Sie bitte mit dem Werkzeugteam.

3.3.4 Installation der verity-Suchmaschine

Nachdem Sie SHORE installiert haben, können Sie jetzt optional die Volltextsuchmaschine installieren. Die Suchmaschine bietet Ihnen die Möglichkeit in Ihren Dokumenten nach beliebigem Text zu suchen.

Für den Einsatz einer verity-Suchmaschine benötigen Sie die entsprechenden Lizenzschlüssel und Installationsmedien für den "verity Information Server search 97 Version 3.6.1" und für die "verity locales englishx Version 2.4.1" (für eigene Projekte besitzt sd&m eine firmenweit gültige Lizenz).

Zur Installation der verity-Suchmaschine ist folgendes nötig

- Deinstallieren Sie eine bereits vorhandenen Installation der verity-Suchmaschine.
- Installieren der verity-Suchmaschine,
- Installieren und Anpassen der verity-Locales und
- Anpassen der verity-Suchmaschine an SHORE.

Installation des Information Server search97 3.6.1 von verity

Denken Sie daran, dass Sie nur die Version: IS S97 3.6.1 verwenden können und dass Sie eine entsprechende Lizenz benötigen. Ausserdem benötigen Sie eine für die Version passende Lizenz für die "verity locales englishx“. Eine Lizenz für den "verity spider" wird nicht benötigt.

So installieren sie die verity Suchmaschine

1. Falls beim Einlegen der CD verity Information Server Search 97 Version 3.6.1 (IS S97 3.6.1) das Installationsprogramm nicht automatisch ausgeführt wird, starten Sie die Installation mit der Datei `Setup.exe`.
   Sie finden die Datei auf der verity-Installations-CD im Verzeichnis `S97is361\_nti31`.
2. Lesen Sie die Lizenzvereinbarungen und akzeptieren Sie diese; dann erst können Sie die Suchmaschine installieren.
3. Wählen Sie im Fenster „Installation Options“ den Punkt „Install full Version“ aus.
4. Ignorieren Sie die Meldung No CGI-compliant Web Server Found.
5. Geben Sie in Fenster „Proxy Host Information“ nichts ein.
6. Geben Sie im Fenster „Administration Information“ beliebige Daten ein. Dabei sind die Daten über die SHORE-Anbindung unbedeutend, da SHORE keine administrativen Funktionen der Suchmaschine nutzt.
7. Stellen Sie im Fenster „Choose Destination Location“ das SHORE-Verzeichnis `...\shore\verity\is` ein.

Achtung: Die Suchmaschine muss unter dem Verzeichnis installiert werden, in dem Sie SHORE installiert haben.

8. Beantworten Sie die Frage, ob man weiter machen möchte, da das Verzeichnis schon existiert, mit „Ja“ oder „Yes“.
9. Wählen Sie den Programm Ordner aus und bestätigen Sie mit „Ja“ oder „Yes“.
   Die Dateien werden kopiert.
10. Geben Sie im Fenster „Registration“ die Lizenzinformationen des "License Key Sheet" von verity ein.
    Eine Spider-Lizenz wird nicht benötigt, da der Spider von SHORE nicht genutzt wird.
11. Ignorieren Sie die Aufforderung, den Webserver neu zu starten („OK“), da es für SHORE nicht relevant ist.
12. Ignorieren Sie die Mitteilung den Server neu zu booten (No), da es für SHORE nicht relevant ist.
13. Der Dienst "Search97 IS Admin" wird gestartet.

Tipp: Dass der Dienst gestartet wurde, ist nur unter Systemsteuerung > Dienste zu erkennen.

14. Deaktivieren Sie diesen Dienst, unter Start > Einstellungen > Systemsteuerung > Dienste, mit „Beenden“ (oder „Stop“) und der Einstellung Startart "manuell".
    SHORE benötigt diesen Dienst nicht.

Die Installation für die verity Suchmaschine ist abgeschlossen, nun müssen die locales installiert werden.

Einstellungen bei der Installation der verity Locales

SHORE verwendet unter anderem den UTF-8-Zeichensatz. Die Suchmaschine unterstützt diesen Zeichensatz nur, wenn die sogenannten locales installiert werden.

So Installieren Sie die verity Locales

1. Starten Sie die Installation von verity Locales mit der Datei `Setup.exe`.
   Sie finden die Datei auf der verity-Installations-CD im Verzeichnis `_nti40\INX200`.
2. Stellen Sie im Fenster „Choose Destination Location“ das SHORE-Verzeichnis `...\shore\verity\is` ein.
3. Im Fenster „License Key“ geben Sie den Lizenzschlüssel für das Feld „Lokalisierung“ "englishx" ein. Der Lizenzschlüssel wird von verity auf einem eigenen "License-Sheet" ausgeliefert.
4. Verschieben Sie die zwei Dateien (`*.dll`) aus dem Verzeichnis `...\verity\is\_nti40\bin` nach `...\verity\is\_nti31\bin`. Diese Dateien wurden durch die Locales-Installation leider im falschen Verzeichnis abgelegt.

Hinweis: Falls dies nicht Ihre erste verity-Installation auf dem Rechner ist, können hier auch mehr Dateien existieren. Dann kopieren Sie alle `*.dll`-Dateien.

Die Installation für die verity Locales ist abgeschlossen. Nun müssen Sie die installierten Locales in der Suchmaschinen-Installation anmelden und die Installation anpassen.

Einstellungen in der Konfigurationsdatei

Um die Erweiterung der Suchmaschine zu nutzen, müssen Sie in der Konfigurationsdatei `inetsrch.ini` die Erweiterung englishx anmelden.
Damit SHORE im Fehlerfall die Meldungen formatiert ausgibt, ändern Sie dafür die Einstellungen der Locales.

1. Öffnen Sie im Verzeichnis `...\verity\is\_nti31\bin` die Datei `inetsrch.ini` in einem Text-Editor.
2. Ändern Sie für die Erweiterung englishx, unter der Sektion `[Common]` den Eintrag
   `locale=english` (oder den Wert, der dort steht) in
   `locale=englishx`
3. Ändern Sie für die SHORE-konforme Fehlermeldung in der Sektion `[S97IS\Primary\SearchDefaults]` den Eintrag
   `ResultErrorTemplate = serror.hts` in
   `ResultErrorTemplate = SHOREerror.hts`.

4. Speichern Sie die Datei.

Die Installation der Suchmaschine ist nun abgeschlossen.

3.3.5 Installation des Kommandozeilen-Clients

Auf dem Server wird zusammen mit SHORE auch der Kommandozeilen-Client installiert. Wenn Sie den Kommandozeilen-Client auch auf anderen Rechnern nutzen wollen, müssen Sie ihn dort manuell installieren.
Der Kommandozeilen-Client ist ein Java-Programm. Damit Sie ihn nutzen können, muss auf dem Rechner eine Java-Laufzeitumgebung oder ein JDK ab Version 1.1.7 installiert sein (vielleicht funktionieren auch ältere Versionen, das haben wir nicht ausprobiert).

1. Kopieren Sie vom Server das Verzeichnis `...\shore\client` einschließlich aller enthaltenen Dateien auf den Zielrechner.
2. Erweitern Sie die Systemvariable `Path` um das neu angelegte Verzeichnis.

Damit ist der Kommandozeilen-Client installiert und Sie können ihn sofort nutzen.

3.4 SHORE einrichten

Nachdem Sie SHORE erfolgreich installiert haben, können Sie damit beginnen, SHORE für sich anzupassen. In den folgenden Abschnitten wird exemplarisch das auf Ihrer SHORE-CD enthaltene Taschenrechnerbeispiel eingerichtet.

3.4.1 Einstellungen am Kommandozeilen-Client

Prüfen Sie noch einmal, ob Sie alle Software-Voraussetzungen erfüllt haben. Dazu gehört ein Netzwerkzugriff auf den SHORE-Server, sowie einen Browser, in dem Java ausgeführt werden kann (Sicherheitseinstellung), wie beispielsweise Netscape ab Version 4.5 oder Internet Explorer ab Version 5.0.

Der Kommandozeilen-Client ist vom Prinzip her eine Fernbedienung für SHORE. Sie können den Kommandozeilen-Client auf dem Server und auf jedem anderen Rechner, der über das Netzwerk mit dem Server verbunden ist, einsetzen. Mit ihm können Sie alle wichtigen Funktionen ausführen.
Zunächst müssen Sie den Kommandozeilen-Client einstellen, danach können Sie ein Metamodell und die dazu passenden Parser anmelden. Dann endlich können Sie Dokumente in SHORE importieren und mit ihnen arbeiten.

So rufen Sie den Kommandozeilen-Client auf

Um mit dem Kommandozeilen-Client arbeiten zu können, muss der SHORE-Server laufen.

Tipp: Der SHORE-Server läuft, wenn Sie im Taskmanager unter dem Register „Prozesse“ den Eintrag „shore-server.exe“ finden können.

Start des SHORE-Servers

1. Sie Starten den SHORE-Server mit `shore.exe`.
   Sie finden die Datei im Verzeichnis `...\shore\bin`.

Um aus einem beliebigen Verzeichnis heraus mit dem Kommandozeilen-Client arbeiten zu können, empfiehlt es sich, das Verzeichnis, in dem sich der Kommandozeilen-Client befindet, mit in den PATH aufzunehmen.

So fügen Sie das Verzeichnis des Kommandozeilen-Clients zur Systemvariablen PATH hinzu

1. Öffnen Sie den Dialog „Systemeigenschaften“: Start > Einstellungen > Systemsteuerung > System und Wählen Sie das Register „Umgebung“.
   Sie sehen Ihre System- und Benutzervariablen
2. Klicken Sie auf die Systemvariable „Path“. Im untern Teil des Fensters sehen Sie im Feld „Variable“ den Eintrag „Path“ und unter „Wert“ Ihre Einträge.
3. Setzen Sie den Cursor im Feld „Wert“ an das Ende Ihrer Einträge.
4. Fügen Sie durch ein Semikolon abgetrennt den absoluten Pfad des Verzeichnisses hinzu, in dem der Kommandozeilen-Client steht.

Syntax ;...\shore\client
Beispiel ;D:\Programme\shore\client

5. Klicken Sie auf „Setzen“ und schließen Sie das Fenster mit „OK“

Aufruf des Kommandozeilen-Clients

1. Öffnen Sie das DOS-Eingabefenster und wechseln Sie in das Laufwerk des Servers.

2. Geben Sie das Kommando `shore` ein und bestätigen Sie mit „`Enter`“.

Achtung: Sie dürfen sich nicht im Verzeichnis `...\shore\bin` befinden. Sonst wird ein zweiter SHORE-Server gestartet.

Der SHORE Kommandozeilen-Client stellt sich mit seinen Kommandos und den aktuellen Einstellungen vor.

```
D:\>shore

SHORE Kommandozeilen-Client 1.9 (Java) vom 31.8.2001

Benutzung: shore <Kommando> <Parameter>

 Kommando       ¦ Parameter
----------------+-------------------------------------
 import         ¦ [-binary¦-text] <projektdoks>
                ¦ [-parser <parser>]
                ¦ [-grouped [n] ] [-noindex]
 imp_mm         ¦ <metamodelldatei> [<zielverzeichnis>]
 imp_mm_rescan  ¦ <metamodelldatei> [<zielverzeichnis>]
 imp_ss         ¦ <style-sheet> <dokumenttyp>
 imp_parser     ¦ <parserdatei> <name> <endungen>
 export         ¦ <projektdokument>
 exp_xmldok     ¦ <projektdokument>
 exp_ss         ¦ <dokumenttyp>
 move           ¦ <name_alt> <name_neu>
 reindex        ¦
 delete         ¦ [-noindex] <projektdoks>
 del_pdok       ¦ [-noindex] <dokumenttyp>
 del_ss         ¦ <dokumenttyp>
 stdquery       ¦ <stdquery> (<paramname>=<wert>)*
 query          ¦ <queryfile> [<prologregel>]
 dump           ¦
 backup         ¦ <zielverzeichnis>
 restore        ¦ <quellverzeichnis>
 admin          ¦ [-server <adresse>] [-port <port>]
                ¦ [-wf <pfad>]
 exit           ¦

Die aktuellen Einstellungen:
        Server              : localhost
        Port                : 8080
        Arbeitsverzeichnis  : /

D:\>_
```

Abbildung 3: Kommandozeilen-Client

So ändern Sie die Servereinstellung im Kommandozeilen-Client

Der Kommandozeilen-Client muss wissen, mit welchem Server er kommunizieren soll.

1. Die Einstellung für den Server ändern Sie mit dem Kommando `shore admin` und dessen Parametern.

2. Sie stellen den Server mit dem Parameter „-server“ ein.
Syntax „shore admin -server <Adresse des Servers>“
Beispiel „shore admin -server localhost“
Bestätigen Sie mit „<Enter>“.

Hinweis: Den Namen des Servers den Sie als Adresse in der URL angeben können, finden Sie heraus, indem Sie auf dem Server in einem DOS-Fenster den Befehl `ipconfig /all` eingeben. Der Name erscheint dann unter Host-Name.

Mit einem weiteren Aufruf des Kommandos `shore` zeigt das Eingabefenster die geänderten Einstellungen an.
Der angegebene Pfad beim Working Folder (Arbeitsverzeichnis) zeigt nun zum Wurzelverzeichnis der Projektdokumente auf der SHORE-Client-Maschine.

So ändern Sie den Port im Kommandozeilen-Client

Über den Port steuern Sie in der Regel, mit welcher Instanz der Kommandozeilen-Client auf dem Server kommunizieren soll.
Die Standardeinstellung für den Port ist 8080.

1. Stellen Sie den Port mit dem Parameter „-port“ ein.
Syntax shore admin -port <Portnummer>
Beispiel „shore admin -port 8766“
Bestätigen Sie mit „<Enter>“.

So ändern Sie das Arbeitsverzeichnis im Kommandozeilen-Client

Das Arbeitsverzeichnis zeigt auf das Verzeichnis, unter dem die zu importierenden Projektdokumente stehen. Es sollte für eine Instanz möglichst gleich bleiben und auf das Verzeichnis zeigen, dass gleichnamig eines eventuell vorhandenen Projektverzeichnisses ist. Für das Taschenrechnerbeispiel legen Sie bitte ein eigenes Verzeichnis an und stellen es als Arbeitsverzeichnis ein.

1. Stellen Sie das Arbeitsverzeichnis mit dem Parameter „-wf“ ein.
Syntax „shore admin -wf <Pfad>“
Beispiel „shore admin -wf D:\Taschenrechnerbeispiel“
Bestätigen Sie mit „Enter“.

3.4.2 Öffnen des SHORE-Browser-Fensters

Kontrollieren Sie nun, ob SHORE im Browser geladen wird.

1. Starten Sie Ihren Browser und geben Sie die URL ein.

Syntax http://<Rechnername des SHORE-Servers>:<Port des SHORE-Servers>/shore

Beispiel `http://shoredemo.muc.sdm.de:8080/shore`

Hinweis: Den Rechnernamen finden Sie im DOS-Eingabefenster - wenn Sie den Kommandozeilen-Client ohne weitere Parameter aufrufen - unter
"Die aktuellen Einstellungen: Server:" und den Port unter "Port:".
Funktioniert die Einstellung localhost nicht, ersetzen Sie im Kommandozeilen-Client und im Browser den Eintrag „`localhost`“ durch den Hostnamen Ihres Rechners, durch „`127.0.0.1`“ oder durch Ihre IP-Adresse .

Die Oberfläche von SHORE wird geladen.

Tipp: Das Laden der Oberfläche kann je nach Rechner etwas länger dauern, da das Java-Applet unter Umständen im Browser kompiliert werden muss.

Bestätigen Sie mit „`Enter`“.

3.4.3 Einrichten einer Instanz

Nun geht es ins Eingemachte - jetzt wird exemplarisch das Taschenrechnerbeispiel eingerichtet.

Dafür kopieren Sie erstmal die Projektdateien des Beispiels in Ihr Arbeitsverzeichnis. Das Kopieren der Dateien ist für den Import später in Ihrem Projekt nicht zwingend notwendig. Sie können auch im Kommandozeilen-Client den Arbeitsordner auf das Verzeichnis einstellen, wo sowieso die Dokumente bereits liegen.

Dann importieren Sie das Metamodell und melden einen Parser an. Für Ihr Projekt müssen Sie sich überlegen, welche Parser sie benötigen und wie Ihr Metamodell aussehen soll. Details dazu finden Sie im Kapitel “Metamodell und Parser” auf Seite 31. Für das Taschenrechnerbeispiel wird ein "dummy"-Parser angemeldet (`xml2xml.exe`), der SHORE-konformes XML akzeptiert. Diesen können Sie für bereits zum Metamodell passend ausgezeichnete XML-Dateien verwenden.

Abschließend importieren Sie die Projektdokumente nach SHORE.

Kopieren Sie die Projektdateien in das Arbeitsverzeichnis

1. Entpacken Sie vom Installationsverzeichnis bzw. von der CD alle Dateien aus der Datei `\beispiele\shore-beispiel-oop-2000.zip` mit Ihren Unterverzeichnissen unter das von Ihnen eingestellte Arbeitsverzeichnis.

So importieren Sie das Metamodell

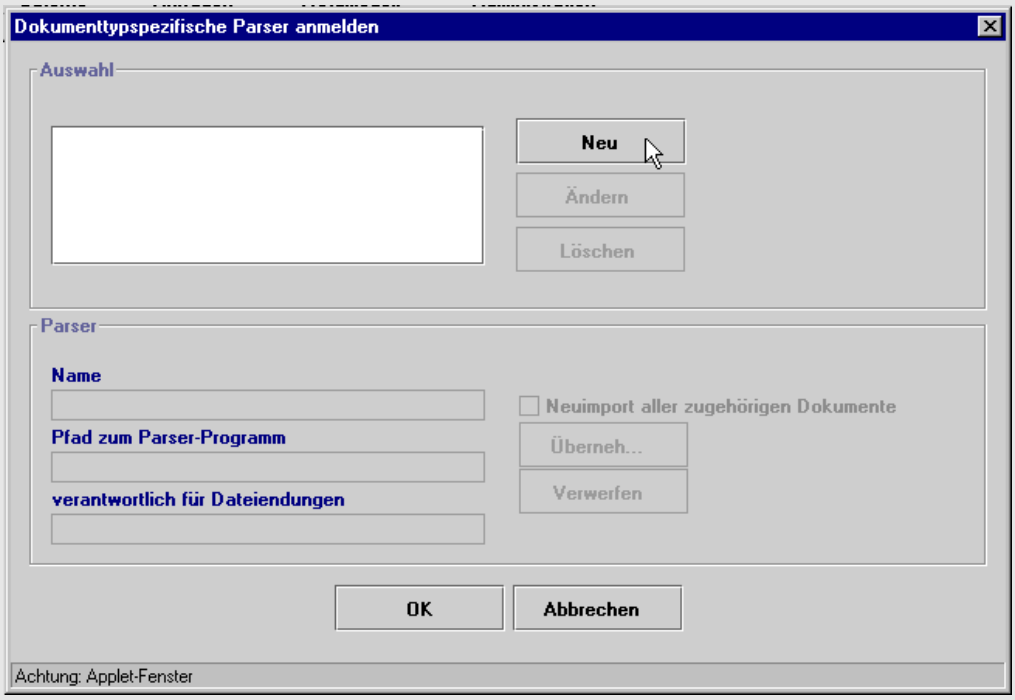
1. Importieren Sie das Metamodell `oop.meta` mit dem Kommandozeilen-Client im DOS-Eingabefenster mit dem Kommando imp_mm und seinen Parametern. Dazu wechseln Sie in das Verzeichnis, in dem die Metamodelldatei gespeichert ist.

Syntax shore imp_mm <Datei>
Beispiel `shore imp_mm oop.meta`
Bestätigen Sie mit „`Enter`“

Hinweis: Der Dateiname kann Wildcards und den relativen Pfad zum Working Folder enthalten.

So melden Sie einen Parser an

1. Wechseln Sie in das SHORE-Browser-Fenster.
2. Wählen Sie im Menü Administration > Dokumentenspezifische Parser... . Das Fenster „Dokumentenspezifische Parser anmelden“ öffnet sich.



3. Klicken Sie auf den Button „Neu“
4. Geben Sie im Feld „Name“ den Namen des Parsers ein.
   „`dummy`“

5. Geben Sie im Feld „Pfad zum Parser-Programm“ den Pfad und Dateinamen des Parsers ein.
   „xml2xml.exe“

Hinweis: hier wird der Pfad relativ zu der Angabe „Standardverzeichnis für Dokumenttyp-spezifische Parser“ des Fensters „Servereinstellungen bearbeiten“ (im Menü Administration>Ändere Einstellungen...) eingetragen.

6. Geben Sie im Feld „verantwortlich für Dateiendungen“ die Kürzel der Dateiendungen ein. Diese können mit Leerzeichen, Komma oder Semikolon getrennt werden.
   „cbl java middle uc“
7. Klicken Sie auf „Übernehmen“. Der Parser erscheint im Bereich „Auswahl“ und ist jetzt angemeldet.
8. Klicken Sie auf „OK“.

So importieren Sie die Projektdokumente

1. Wechseln Sie in das DOS-Eingabefenster und wechseln Sie in das Verzeichnis, in dem die zu importierenden Dokumente liegen.
   Dieses Verzeichnis muss im oder unter dem Arbeitsverzeichnis liegen.
2. Importieren Sie die Dateien mit dem Kommando `shore import` und seinen Parametern.
   Den erfolgreichen Import zeigt der Kommandozeilenclient für jede Datei mit einem "ok" an.

Syntax <Pfad des Arbeitsverzeichnisses>shore import <Datei>
Beispiel „shore import .\java*.java“

Hinweis: Der Dateiname kann Wildcards und den relativen Pfad zum Working Folder enthalten.

Mit dem Beispiel importieren Sie alle Dateien aus dem Verzeichnis java die auf .java enden. Bestätigen Sie mit „Enter“

Nun haben Sie das Taschenrechnerbeispiel eingerichtet und alle java-Dateien importiert. Wiederholden Sie den Import auch für die restlichen Verzeichnisse (`uc`, `middle` und `cobol`). Damit ist das Taschenrechnerbeispiel dann komplett in SHORE geladen.
Nun können Sie - wie im Anwenderhanbuch beschrieben - durch die Dokumente navigieren und Auswertungen erstellen.

4 Arbeiten mit SHORE

4.1 Systemmanagement

Zum Systemmanagement zählen alle Handlungen am SHORE-System, die nicht regelmäßig durchgeführt werden. Hier finden Sie auch weitergehende Informationen rund um den SHORE-Server.

4.1.1 Metamodell und Parser

Welche Dokumenttypen existieren und welche Dinge samt ihrer Beziehungen untereinander für die Navigation und Anfragen relevant sind, bestimmt ein frei definierbares Metamodell in Verbindung mit den projektspezifischen Parsern. Weitere Informationen zum Metamodell entnehmen Sie bitte dem Fachkonzept von SHORE.

Mit SHORE sind Sie in der Lage die Metamodelle den Erfordernissen Ihres Projekts flexibel anzupassen.

Metamodelle - Einführung

Die Metamodelle von SHORE finden Sie in Dateien mit dem Suffix `.meta`. Als Beispiel für ein solche Metamodellbeschreibung werfen Sie bitte einen Blick in die Datei `general.meta`, die Sie im Verzeichnis `shore\parser\meta\` finden. In dieser Datei ist das SHORE Standard-Metamodell für Programmiersprachen festlegt, dazu später noch genaueres.

Metamodell-Inhalt

Im Metamodell wird festgelegt,

- welche Dokument-, Objekt- und Beziehungstypen es gibt,
- welches die Quell- und Zielobjekttypen der Beziehungstypen sind,
- welche Vererbungsbeziehungen es bei den einzelnen Typen gibt, sowie
- welche Objekttypen in welchen Dokumenttypen definiert werden und welche Beziehungstypen in welchen Dokumenttypen vorkommen dürfen

.

Ein Metamodell ist jeweils projektspezifisch und muss dem SHORE-System in einer Metamodellspezifikation gemäß folgender Notation als Dokument zur Verfügung gestellt werden.

Metamodell-Syntax

Eine Metamodellspezifikation besteht aus drei Typen von Einträgen:

Metamodellspezifikation

- Deklaration von Dokumenttypen mit optionaler Angabe von Dokumenttyp-Supertypen und der Information, ob Dokumente dieses Typs für die Volltextsuche indiziert werden, in der Form
  Dokumenttyp `Dokumenttypname`
  `(` **ist** `(`**ein**`|`**eine**`) Dokumenttypname )*`
  `[` **ist nicht volltextindizierbar** `]`
- Deklaration von Objekttypen mit Angabe der Dokumenttypen, in denen sie definiert werden und optionaler Angabe von Objekttyp-Supertypen in der Form
  Objekttyp `Objekttypname`
  `(` **ist** `(`**ein**`|`**eine**`) Objekttypname )*`
  `(` **ist definiert in** `Dokumenttypname )*`
- Deklaration von Beziehungstypen mit Angabe der Quell- und Zielresource und deren Kardinalitäten , mit Angabe der Dokumenttypen, in denen sie definiert werden können und optionaler Angabe von Beziehungstyp-Supertypen in der Form
  Beziehungstyp `Beziehungstypname [` **alias** `Aliasname ]`
  von `n1` **bis** `n2|* Resourcetypname`
  nach `m1` **bis** `m2|* Resourcetypname`
  `(` **ist Teilmenge von** `Beziehungstypname )*`
  `(` **ist definiert in** `Dokumenttypname )*`

Reihenfolge, Whitespace und Kommentare

Die Reihenfolge der Einträge ist dabei beliebig, Schlüsselwörter und Bezeichner werden durch beliebige Whitespace-Zeichen (Newline, Tab, Blank) voneinander getrennt, mit '#' an beliebiger Stelle einer Zeile kann der Rest der Zeile inklusive des Zeilenumbruchs als Kommentar gekennzeichnet werden.

Deklaration und Namensgebung

Alle vorkommenden Typen müssen in genau einem Eintrag deklariert werden. Jeder Typ muß durch seinen Namen eindeutig gekennzeichnet sein. Namen mit gleicher Buchstabenreihenfolge aber unterschiedlicher Groß-/Kleinschreibung gelten als gleich (nicht "case sensitive"). Für Prolog werden die Anfangsbuchstaben der Namen ggf. in Kleinbuchstaben umgewandelt.

Vererbung

Bei der Spezialisierung von Beziehungstypen ist zu beachten, daß die zugehörigen Quell- und Zielressourcen entweder gleich bleiben oder ebenfalls spezialisiert werden. Zyklische Vererbungsbeziehungen sind nicht erlaubt.

implizite Transitivität

Transitiv erschließbare Vererbungsbeziehungen müssen nicht explizit angegeben werden (z.B. 'C-Funktion ist eine Funktion' und 'Funktion ist ein Programmelement' impliziert 'C-Funktion ist ein Programmelement').

Parser - Einführung

Das Metamodell gibt die Typen der nach SHORE zu importierenden Objekte und Beziehungen vor. Parser werden dazu geschrieben, diese Objekte und Beziehungen in den Dokumenten erkennen. Um den Import vorzubereiten, wandeln die Parser die nach SHORE zu importierenden Dokumente in XML-Dateien um.

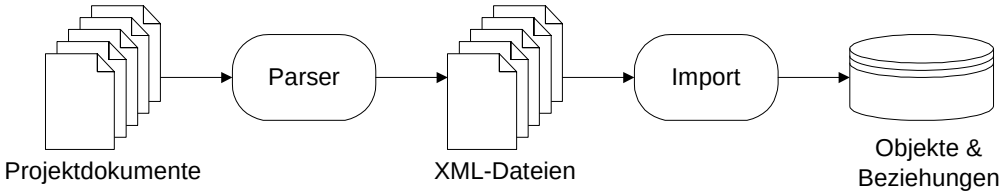


Abbildung 4: Schema, wie Dokumente nach SHORE gelangen

Beim Import dieser Dokumente nach SHORE wirkt das vorher festgelegte Metamodell wie ein Filter. Nur die Objekte und Beziehungen, die im Metamodell festgelegt sind, werden in die SHORE-Datenbank übernommen.

Für jeden Objekttyp in den erzeugten XML-Dokumenten muss das Attribut `Name` angegeben sein. Entsprechend werden für jeden Beziehungstyp die Attribute `Quelltyp`, `Zieltyp`, `Quellname` und `Zielname` verlangt.

Verträglichkeitsbedingungen

Damit sich das, was die Parser an XML-Dokumenten erzeugen, auch ohne weiteres importieren läßt, müssen einige weitere Verträglichkeitsbedingungen erfüllt sein.

- Zunächst sind sämtliche Objekt- und Beziehungstypen, von denen geerbt wird, abstrakt. Objekte oder Beziehungen eines solchen abstrakten Typs dürfen damit in den XML-Dokumenten nicht vorkommen. Vielmehr gilt: In den XML-Dokumenten sind nur Objekte oder Beziehungen erlaubt, von deren Typ nicht geerbt wird.
- Weiterhin muss der Quelltyp und der Zieltyp einer jeden vom Parser erkannten Beziehung ein Untertyp der für den entsprechenden Beziehungstyp festgelegten Quell- und Zieltypen sein oder mit diesen übereinstimmen.

Steuern der Parser und des Imports

Sie haben verschiedene Möglichkeiten, das Parsen und den Import von Dateien zu steuern. Hier werden einige Szenarien beschrieben, es sind aber noch weitere Varianten denkbar. In einer Instanz können durchaus diese Szenarien gemischt vorkommen.

Welches Szenario zur Anwendung kommt, bestimmt unter anderem die Abhängigkeit der Dateien untereinander. Das bedeutet, dass, wenn in einer Datei eine Änderung gemacht wurde, andere - abhängige - Dateien ebenfalls importiert werden müssen.

Szenario 1: eine Dateiendung - ein Parser - keine Abhängigkeiten

Die Situation

In einem Verzeichnisbaum liegen Dateien unterschiedlichen Typs. Zu jedem Typ können Sie genau einen Parser zuordnen und zwischen den Dateien existieren nur wenig Abhängigkeiten.

Die Lösung

Dann melden Sie die Dateiendungen und Parser direkt in SHORE an und importieren ganz normal mit dem Kommandozeilen-Client (siehe “So importieren Sie Dokumente mit dem Kommandozeilen-Client” auf Seite 77).

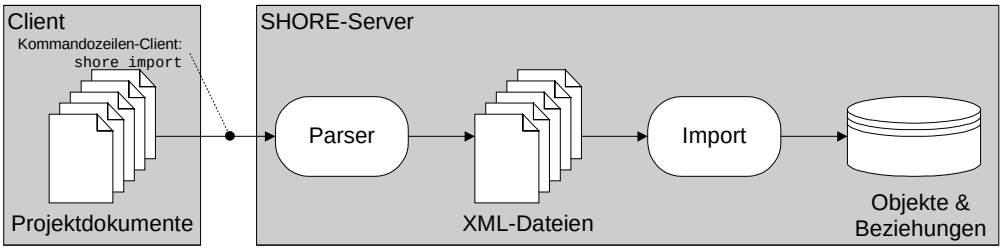


Abbildung 5: Schema eines einfachen Imports

Szenario 2: eine Dateiendung - mehrere Parser - keine Abhängigkeiten

Die Situation

In einem Verzeichnisbaum (auf einem SHORE-Client) liegt eine Menge von Dateien unterschiedlichen Typs (mit zum Teil gleichen Dateiendungen, wie z.B. Worddokumente) und es ist Ihr Wunsch, diese Dateien der Reihe nach zu importieren. Unter Umständen aber reicht die Dateiendung allein nicht aus, um genau den richtigen Parser auszuwählen oder den SHORE-Dokumenttyp daraus zu erschließen. Um die richtige Unterscheidung zu treffen, müssen Sie auch den Dateinamen noch heranziehen.

Die Lösung

In diesem Fall empfiehlt es sich

- für die Dateiendung das Programm `dispatcher.exe` als Parser bei SHORE anzumelden. Sie nutzen damit einen bereits vorhandenen, noch zu konfigurierenden aber leicht konfigurierbaren dispatcher-Mechanismus, der zum Parsen der Dateien auf verschiedene andere Perl-Parser-Module verzweigt. Diese Module können auch anhand des Dateinamens den richtigen Parser starten.
- Eine ganze Reihe bereits existierender Perl-Module können Sie für sich anpassen und ansteuern:
  - zur XML-Konvertierung von Word-Dokumenten
  - zur XML-Konvertierung von Excel-Dokumenten
  - zum Parsen von XML-Tabellen
  - zum Parsen von Rational Rose Modellen

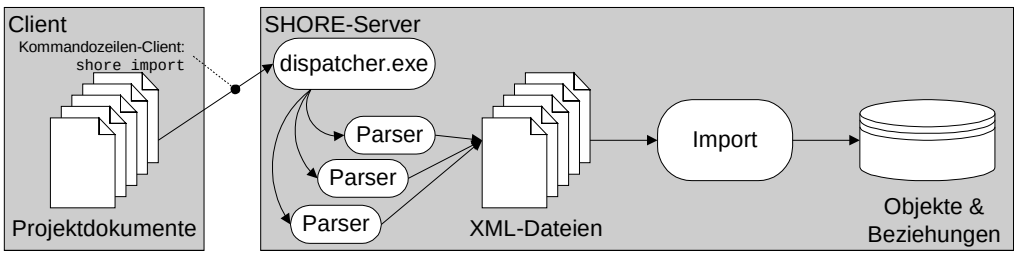


Abbildung 6: Einsatz von dispatcher.exe

Beispielsweise können Sie in Word vorliegende Dokumente des Fachkonzepts oder DV-Konzepts separat parsen, wenn alle Teile des Fachkonzepts dem Muster `FK_*.doc` entsprechen und alle Dokumente des DV-Konzepts dem Muster `DVK_*.doc`. Das Programm `dispatcher.exe` fragt dann genau auf dieses Muster ab und die jeweils angemeldeten Perlmodule parsen dann die Dokumente.

Szenario 3: Parsen von Dateien mit Abhängigkeiten

Die Situation

In mehreren Verzeichnisbäumen (auf dem SHORE-Server) liegen Quelltexte, die voneinander in einer Weise abhängig sind, dass beim SHORE-Import auch Abhängigkeiten zu berücksichtigen sind.

Beispiel: Änderungen, die in einem C-Header-File gemacht wurden, wirken sich darauf aus, dass sämtliche C-Sourcen neu geparst werden müssen, die dieses Header-File inkludieren. Die Parser müssen hier nach den selben Regeln vorgehen wie Compiler.

Die Lösung

Für diese Situation werden Parser eingesetzt, die ein "Gedächtnis" besitzen. Dieses Parsergedächtnis ist ein Verzeichnis, in dem alle bereits geparsten und noch zu parsenden Sourcen gespeichert sind. Die Parser entscheiden dann anhand des Datums, welche Dateien bei einem Import verarbeitet werden sollen.
Will man das Netz der Objekte und Beziehungen in SHORE auf Dauer konsistent halten, so muss man akzeptieren, dass die Parser sehr sensibel auf die Tatsache reagieren, dass sich bestimmte Dateien geändert haben. Falls Sie also das SHORE-Repository inkrementell auf dem neusten Stand halten und nicht jede Nacht sämtliche Sourcen ihres Projekts von Grund auf neu importieren, so können Sie den schon vorher beschriebenen dispatcher-Mechanismus nutzen oder den ImportBatch-Mechanismus verwenden.

Für den Import mit ImportBatch gehen Sie wie folgt vor:

1. Kopieren Sie die zu importierenden Dateien in das im Batchfile benannte `projektVerzeichnis`
2. Starten Sie den Importbatch mit `shore import <myBatch.batch>`.

Der Parser für diese Batchdatei stößt dann (ähnlich wie der oben beschriebene Dispatcher) die weiteren Parser-Module an, die sich ihrerseits erst einmal aus dem Projektverzeichnis die zu parsenden Dateien heraussuchen, indem sie das Datum der letzten Änderung dieser Dateien mit dem Datum des letzten erfolgreichen Imports vergleichen und dann ermitteln, welche Dateien von diesen geänderten Dateien noch abhängen.

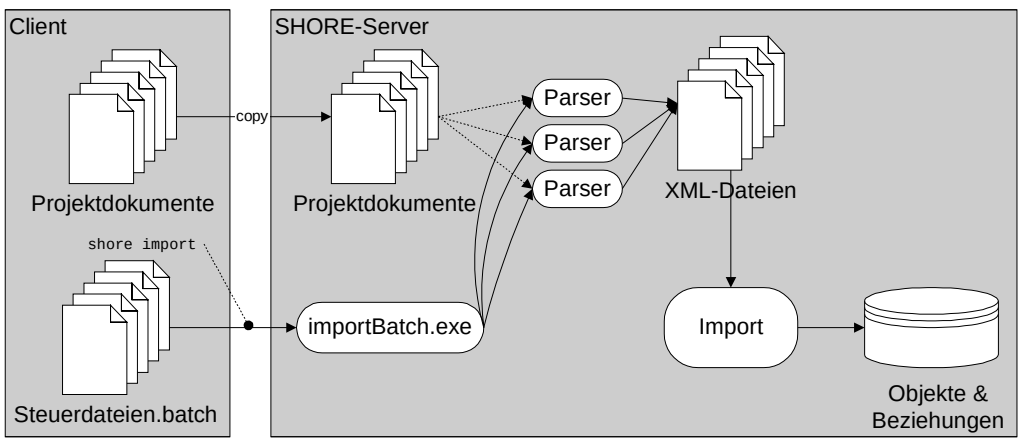


Abbildung 7: Import abhängiger Dateien mit importBatch.exe als Parser

Wie kann ich sehen, ob meine Parser richtig arbeiten?
Wenn Sie einen Parser bei SHORE anmelden, so können Sie die XML-Dateien, die der Parser erzeugt hat, nur auf dem Server sehen. Wenn Sie die Dateien lokal sehen wollen, so müssen Sie den Parser mit Ihren Dateien so wie in SHORE aufrufen, ihm aber als letztes Argument ein Ausgabeverzeichnis mitgeben. Sie starten also den Parser dann nicht über `shore import`, sondern explizit. Damit Sie Ihre Dateien nach SHORE importieren können, müssen Sie nun als Parser `xml2xml.exe` anmelden. Dann können Sie die erzeugten XML-Dateien mit dem Kommandozeilen-Client nach SHORE importieren.

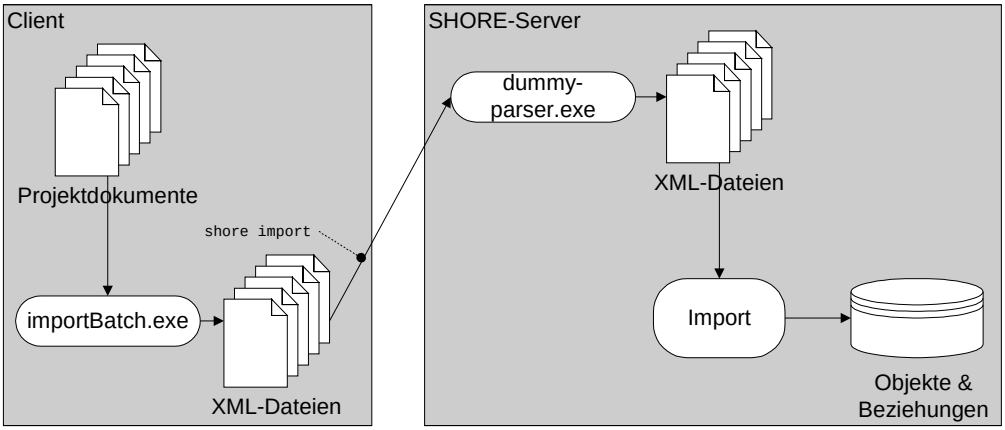


Abbildung 8: Import abhängiger Dateien mit manuell aufgerufenem importBatch.exe

Anbindung von SHORE an ein KM-System

Sehr häufig haben Sie die aktuellen Quellen, die Sie mit SHORE navigierbar machen wollen, in einem Konfigurationsmanagement-System eingecheckt.

Setzen Sie, wenn möglich, einen Trigger, der dieses Einchecken damit verknüpft, dass die entsprechende Source im entsprechenden Verzeichnis auf dem shore-Server landet oder direkt ein Import nach SHORE erfolgt.

Alternativ können Sie das at-Kommando verwenden um regelmäßig die Sourcen auf den SHORE Server zu kopieren.

Die Sourcen sollten im Verzeichnis `shore\projects\projektname\` sein, die zugehörigen Instanzen, die Sie mit den Sourcen aus den Projektverzeichnissen bestücken, sollten Sie im Verzeichnis `shore\docstore\instanzenname\` unterbringen.

Mittels der Skriptumgebung haben Sie Wahlfreiheit, wie Sie Ihre Sourcen aus den unterschiedlichen Projekten auf die Instanzen verteilen.

Triggerung über at-Kommandos oder cron-jobs

Wenn Sie einen regelmäßigen Import durchführen wollen, können Sie einen Scheduling-Mechanismus installieren. Auf einem NT-Rechner beispielsweise können Sie das AT-Kommando verwenden, um zu bestimmten Zeitpunkten mit Hilfe des Kommandozeilen-Clients ihre Dateien zu importieren. Sie können auch von anderen Plattformen aus den Kommandozeilen-Client starten, da er in Java implementiert ist (unter Unix z.B. per entsprechendem Eintrag in der crontab).

Sie können einen Scheduling-Mechanismus auf mehrere Arten nutzen. Hier zwei Varianten:

1. Auf einem Rechner, der alle Sourcen beherbergt, wird regelmäßig ein Import aller oder nur der neuesten Dateien angestoßen. Die importierten Dateien werden nicht gelöscht.
2. Die Projektmitarbeiter können die zu importierenden Dateien in ein Verzeichnis auf einem Rechner speichern. Ein Batchjob importiert zu bestimmten Zeiten alle dort abgelegten Dateien und löscht die Dateien aus dem Verzeichnis.

Das Parserframework

Zusammen mit SHORE bekommen Sie bereits existierende Parser, die automatisch mit installiert werden.

Grundsätzlich ist es egal, wie ein Parser intern funktioniert, für SHORE ist wichtig, dass er SHORE-konformes XML generiert. Die vom Werkzeugteam zur Verfügung gestellten Parser sind

- ausführbare Programme oder
- `class`-Files in einem `jar`-File oder
- Perl-Module oder
- SHORE-PatternMatcher-Skripten oder
- LIM/XSLT-Skripten.

Alle Parser werden von einer Skriptumgebung angesteuert, in die Sie auch eigene Parser einhängen können. Die Skriptumgebung besteht aus

- einer Menge von Perl-Modulen, die unterhalb des Verzeichnis `shore\parser\lib\perl` zu finden sind
- einem Perlskript `importBatch.pl`
- einem Perlskript `dispatcher.pl`

Wie Sie die einzelnen Parser und die Skriptumgebung nutzen, steht in einem eigenen Dokument, dem Parserkochbuch.

ImportBatch einrichten

ImportBatch-Mechanismus

Der mitgelieferte Parser `importBatch.exe` ist kein Parser im herkömmlichen Sinn. Er bekommt keine Dokumente zum Parsen, sondern Anweisungen, wo die Dateien gespeichert sind, die geparst werden sollen und was für ein Parser dafür verwendet werden soll. Es existieren folgende Parser:

- java2xmlstartet den Java-Parser
- cobol2xmlstartet den Cobol-Parser
- xcobol2xmlstartet den Cobol-Parser
- limInput2xmlstartet den LIM/XSLT-Patternmatcher
- pmInput2xmlstartet den SHORE-Patternmatcher

So richten Sie importBatch.exe ein

1. Melden Sie das Programm `importBatch.exe` für die Files mit dem Suffix `.batch` als Parser an.
2. Melden Sie das Programm `xml2xml.exe` für alle Dateiendungen an, die über `importBatch.exe` importiert werden sollen, beispielsweise `java; doc; cob`
3. Schreiben Sie Batchdateien mit Statements der Form

   ```
   something2xml
   <projektVerzeichnis>
   <suffixes> ... <weitere Argumente>
   target
   ```

   Details finden Sie im Anschluß an diese Anleitung.
   Ein Batchfile `myBatch.batch` sieht beispielsweise so aus:

   ```
   java2xml
   $SHORE/projects/meinProjekt/quellen/java/src
   java
   D:/de/sdm/meinProjekt;D:/de/sdm/Basis
   $SHORE/projects/meinProjekt/quellen/java/xmlm
   all;
   ```

Aufbau der Batchdatei

Die Anzahl der Argumente der Batchdateien ist je nach Parser unterschiedlich. Grundsätzlich steht `something2xml` für den jeweiligen Parser. Unter `suffixes` geben Sie die Dateieindungen getrennt durch einen senkrechten Strich an, die der Parser berücksichtigen soll.
Beispiel: `cob|cbl`
Dabei können Sie primäre und sekundäre suffixes angeben. Primäre sind die eigentlichen Source-Dateien und Sekundäre bezeichnen Include-Dateien. Wenn Sie diese Möglichkeit nutzen wollen, gebrauchen Sie folgende Konvention: Primäre Dateiendungen stehen in runden Klammern, Sekundäre stehen in eckigen Klammern.
Beispiel: `(c|cpp)[h|hh]`
Das Argument`target` kann einen der folgenden Werte annehmen:
`xml`, wenn alle Dateien geparst werden sollen (keine Abhängigkeitsanalyse)
`delta`, wenn nur neue Dateien geparst werden sollen, wobei Abhängigkeiten berücksichtigt werden
`all`, wenn alle Dateien neu geparst werden sollen und Berechnungen zur Abhängigkeitsanalyse durchgeführt weren sollen.

Achtung: Die Pfadangaben müssen Sie immer absolut eingeben (also mit Laufwerkbuchstaben) oder Sie verwenden den speziellen Platzhalter $SHORE. Dieser wird von der Parserskriptumgebung als das Verzeichnis, in dem SHORE installiert ist interpretiert.

Der Dispatcher erkennt anhand seiner Konfiguration aus Dateinamen und Dateiendung die zugehörigen SHORE-Dokumenttypen und ruft die zugehörigen Parser auf. Die Konfiguration stellen Sie in sogenannten Frontend-, Dokumenttyp- und Backend-Tabellen ein. Der Dispatcher arbeitet zweistufig. Um Dokumente nach SHORE zu importieren, erzeugt zunächst ein Frontend aus einem proprietären Format XML-Dokumente. Danach reichert ein Backend diese XML-Dokumente mit zusätzlichem Markup an. Die Frontend / Backend-Architektur können Sie auf vielfältige Weise und für vielfältige Zwecke einsetzen.

So nutzen Sie den Dispatcher

1. Melden Sie den Dispatcher bei SHORE als Parser für Ihre unterschiedlichen Dokumenttypen an.
2. Konfigurieren Sie den Dispatcher

Dabei müssen die von Dispatcher angesteuerten Module den vom ihm vorgegebenen Schnittstellen genügen (und Funktionen zur Verfügung stellen, die vom Dispatcher-Modul gerufen werden).

Zusammen mit SHORE erhalten Sie bereits Module, die vom Dispatcher angesteuert werden können. Dazu gehören beispielsweise anpassbare Parser für Worddokumente und Exceltabellen.

So können Sie mit dem Kommando "shore import" beliebige Dateien importieren.

Damit Ihr Dispatcher richtig arbeitet spezifizieren Sie folgende Dinge:

- Ihre Frontend-Tabelle

- Ihre Dokumenttyp-Tabelle

- Ihre Backend-Tabelle

Die Belegung der Tabellen-Inhalte erfolgt in einem kleinen Perl-Skript. Dieses Skript initialisiert das Dispatcher-Modul und ruft dann dessen run-Methode auf. Die Frontend-Tabelle und die Dokumenttyp-Tabelle müssen Sie als Perl-Arrays, die Backend-Tabelle dagegen als Hash realisieren. Diese Implementierungsdetails sind aus der mitgelieferten Vorlage `/shore/bin/Dispatcher.pl` ersichtlich, im folgenden sprechen wir weiterhin von Tabellen. Sie können sowohl an dieser Datei Ihre Änderungen machen, als auch eine Kopie von Dispatcher.exe anfertigen. Die kopierte Datei erwartet dann eine namensgleiche Perl-Datei im selben Verzeichnis in der die Einstellungen stehen.

So konfigurieren Sie den Dispatcher

1. `Dispatcher.exe` erwartet eine Konfigurationsdatei `Dispatcher.pl`. Wenn Sie Dispatcher.exe umbenennen oder kopieren, dann müssen Sie auch die Konfigurationsdatei entsprechend umbenennen oder kopieren.
2. Öffnen Sie die Konfigurationsdatei in einem Texteditor und stellen Sie die Tabellen wie folgt ein.
3. Die Frontend-Tabelle
   Die Zeilen dieser Tabelle haben folgende Form:
   `[Suffix , Dokumenttyp , Multiple-Output-Flag , Frontend-Modul , Argumente ]`
   Durch Zeilen dieser Form legen Sie fest,
   - für welche Dateiendungen (Spalte 1)
   - welche Dokumenttypen zu generieren sind (Spalte 2)
   - welche Frontend-Parser (Spalte 4)
   - mit welchen Argumenten aufgerufen werden (Spalte 5) und
   - ob der Frontend-Parser nur eine oder mehrere Zwischendateien erzeugt (Multiple-Output-Flag, Spalte 3)

   Beispiele:
   ```
   ["cls|bas|frm|vbp" , "VBQuelltext" , undef , "Importer::Copy" , undef ]
   ["lim" , "LimQuelltext" , undef , "Importer::Copy" , undef ]
   ["doc" , undef , undef , "Importer::Copy" , undef ]
   ```
4. Die Dokumenttyp-Tabelle
   Die Zeilen dieser Tabelle haben folgende Form:
   `[ Muster , Dokumenttyp , Multiple-Output-Flag]`
   Wird in der Frontend-Tabelle für eine Dateiendung der Dokumenttyp nicht definiert (explizit auf `undef`), so müssen Sie den Dokumenttyp über die Dokumenttyp-Tabelle festlegen. Dort geben Sie an:
   - zu welchen Mustern von Dateinamen (Spalte 1)
   - welcher Dokumenttyp (Spalte 2) gehört und
   - ob der Frontend-Parser nur eine oder mehrere Zwischendateien erzeugt (Multiple-Output-Flag, Spalte 3)

   Beispiele:
   ```
   ["FK.*" , "Fachkonzept" , undef  ]
   ["DVK.*" , "DVKonzept" , undef  ]
   ["Glossar" , "Glossar" , undef  ]
   ```

5. Die Backend-Tabelle
   Die ersten beiden Tabellen müssen so beschaffen sein, dass der Dispatcher in der Lage ist, für jede Datei den gewünschten Dokumenttyp zu berechnen. Um Dokumente für den SHORE-Import zu erzeugen, müssen die Dateien, die von den Frontends erzeugt werden, ggf. durch einen Parser noch weiter transformiert (und mit Markup angereichert) werden. Dies wird in der Backend-Tabelle definiert.
   Ihre Zeilen sind von der Form:
   `Dokumenttyp => [ Backend-Modul, Argumente]`Durch die Tabelle legen Sie fest,
   - für welchen Dokumenttyp (Spalte 1)
   - welches Backend-Modul (Spalte 2)
   - mit welchen Argumenten (Spalte 3)
   aufgerufen wird.
   Beispiel:

```
"BatchFile" => ["Importer::ImportBatch", undef],
"Glossar" => ["Importer::Skip", undef],
"VBQuelltext" => ["Importer::Skip", undef],
"LimQuelltext" => ["Importer::Skip", undef],
```

An der Schnittstelle zwischen Frontend und Backend beachten Sie bitte folgendes:

- wenn Sie kein Frontend-Modul für ein Dokument spezifizieren, so nimmt das Backend-Modul das Original-Dokument als Eingabe

- wenn das Frontend-Modul aus einem Dokument mehrere Zwischendateien erzeugt, so wird der Dokumenttyp für das Hauptdokument aus der Dateieendung und dem Dateinamen des Eingabedokuments errechnet. Die Dokumenttypen der weiteren Dokumente sind an den Namen dieser Dokumente zu erkennen. Die Namen sind von der Form `<document-name>.<document-type>.xml`

Schnittstelle zu den Frontends und Backends

Wenn Sie eigene Module einhängen wollen: von den Frontend-Modulen, die vom Dispatcher aufgerufen werden können, wird erwartet dass sie eine Funktion `convert` zur Verfügung stellen, die die folgende Signatur hat:

```
convert($p, $doctype, \%args);
```

Von den Backend-Modulen, die vom Dispatcher aufgerufen werden können, wird erwartet dass sie eine Funktion `markup` zur Verfügung stellen, die die folgende Signatur hat:

```
markup($tmpfile, $outputpath, $type, \%args );
```

Als letztes Argument wird in beiden Fällen eine Reverenz auf einen anoymen Hash erwartet, den man in Perl nach folgendem Schema initialisert:

```
{ argName1 => $value1, ... argNameN => $valueN,}
```

`$p` ist eine Instanz auf ein `Shore::Interface` Objekt, über dessen Methoden man auf Dinge wie den namen des zu dispatchenden Files herankommt.

Parser installieren

Zur Installation eines Parsers muss die jeweilige Laufzeitumgebung eingerichtet werden.

Der Java-Parser verlangt

- eine vorherige Installation von Java 1.2 (JDK oder JRE)
- und eine Belegung der Umgebungsvariablen `JAVA12_CP` mit dem Pfad zur jar-Datei `rt.jar` sowie`JAVA12_EXE` mit dem Pfad zu `java.exe`

Für den Import erstellen Sie eine Batchdatei mit der Endung `batch`, die Sie später per Kommandozeilen-Client importieren werden. Als Parser für die Dateiendung ist `importBatch.exe` angemeldet.

So richten Sie den Java-Parser ein

1. Erstellen Sie mit einem Texteditor eine Batchdatei (z.B. `meinProjektJavaImport.batch`) nach folgendem Schema:

Syntax
```
java2xml
projektVerzeichnis
DateiEndungen
projektClassPath
projektXmlmVerzeichnis
target;
```

Beispiel
```
java2xml
$SHORE/projects/meinProjekt/quellen/java/src
java
D:/de/sdm/meinProjekt;D:/de/sdm/Basis
$SHORE/projects/meinProjekt/quellen/java/xmlm
all;
```

Achtung:

- `projektVerzeichnis` und `projektXmlmVerzeichnis` sind absolut anzugeben.
- Das Projektverzeichnis muss auf die oberste Package-Ebene zeigen. In unserem Beispiel folgt unter `D:/Programme/shore/projects/meinProjekt/quellen/java/src/` eine Verzeichnisstruktur des Musters `de/sdm/myProjekt/...`
- In `projektClasspath` stehen alle `jar` und `zip`-Dateien oder Verzeichnisse, die von den zu parsenden Dateien verwendet werden. Hier muss ggf. auch das Verzeichnis der zu parsenden Dateien angegeben werden. Unter Unix sind die Elemente des `projektClasspath` durch ":" zu trennen.
- Unter NT sind die Elemente des `projektClasspath` durch ";" zu trennen.
- Statt des Targets `all` ist auch das target `delta` zugelassen.
- Im Projektverzeichnis selbst werden nur Packages (und darunter ggf Sub-Packages) aber keine Java-Sourcen erwartet.

Für das Projektverzeichnis muss folgendes zutreffen:

- Zu jedem File `x.java` muss im selben Verzeichnis ein File `x.class` vorhanden sein.
- Die `class`-Files der im Projekt verwendeten Bibliotheken sollten über den `projektClassPath` erreichbar sein, ggf als Elemente von Zipfiles.

So importieren Sie Ihre Java-Sourcen

1. Kopieren Sie Ihre Projektdateien in das `projektVerzeichnis` (z.B. `D:/Programme/shore/projects/meinProjekt/quellen/java/src`).
2. Importieren Sie die Batchdatei mit dem Kommandozeilen-Client: `shore import meinProjektJavaImport.batch`
   Details zum import-Befehl: Siehe “So importieren Sie Dokumente mit dem Kommandozeilen-Client” auf Seite 77.

Cobol-Parser installieren

Für den Import erstellen Sie eine Batchdatei mit der Endung `batch`, die Sie später per Kommandozeilen-Client importieren werden. Als Parser für die Dateiendung ist `importBatch.exe` angemeldet.

So richten Sie den Cobol-Parser ein

1. Erstellen Sie mit einem Texteditor eine Batchdatei (z.B. `meinProjektCobolImport.batch`) nach folgendem Schema:

Syntax
```
cobol2xml
projektVerzeichnis
DateiEndungen
projektXmlmVerzeichnis
target;
```

Beispiel
```
cobol2xml
$SHORE/projects/meinPrj/quellen/Cobol/src
cbl
$SHORE/projects/meinPrj/quellen/cobol/xmlm
all;
```

Achtung:

- `projektVerzeichnis` und `projektXmlmVerzeichnis` sind absolut anzugeben.
- Statt des Targets `all` ist auch das target `delta` zugelassen.

So importieren Sie Ihre Cobol-Sourcen

1. Kopieren Sie Ihre Projektdateien in das `projektVerzeichnis` (z.B. `D:/Programme/shore/projects/meinProjekt/quellen/Cobol/src`).
2. Importieren Sie die Batchdatei mit dem Kommandozeilen-Client: `shore import meinProjektCobolImport.batch`
   Details zum import-Befehl: Siehe “So importieren Sie Dokumente mit dem Kommandozeilen-Client” auf Seite 77.

LIM/XSLT-Pattern-Matcher

Dies ist ein Parser, mit dem Sie mittels Patternmatching Strukturen erkennen können und mit XML anreichern können. Das Patternmatching erfolgt zweistufig (siehe nachfolgende Abbildung))

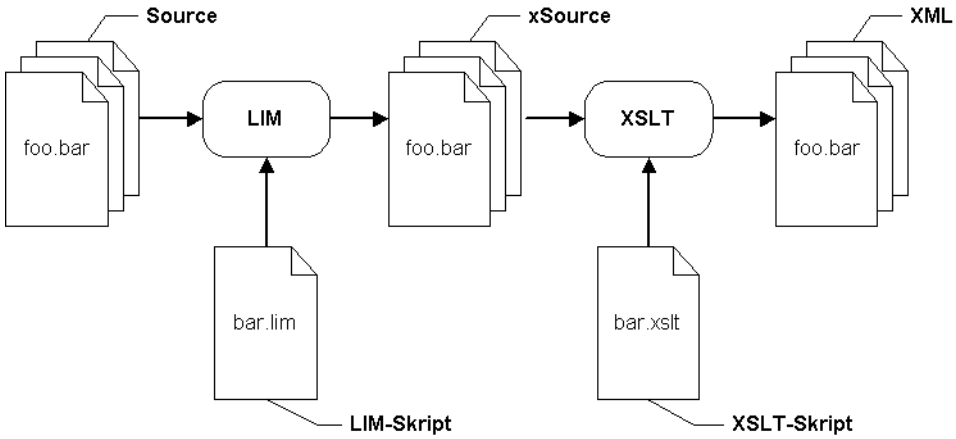


Abbildung 9: Umwandeln der Sourcen in XML mit LIM/XSLT

:

1. mit der Patternmatching-Sprache LIM wird aus einer Datei einfaches XML erzeugt und in einer Zwischendatei gespeichert (xSource).
2. Die Zwischendatei (xSource) wird mittels XSLT in SHORE-konformes XML umgewandelt.

Wie Sie mit LIM/XSLT ein Patternmatching programmieren können, finden Sie in “Anpassen eines LIM/XSLT-Scanners” auf Seite 49.

Für den Import erstellen Sie eine Batchdatei mit der Endung `batch`, die Sie später per Kommandozeilen-Client importieren wird. Als Parser für die Dateiendung `batch` wird `importBatch.exe` angemeldet. Für die Endungen der zu verarbeitenden Dateien (z.B. `4D`)tragen Sie den `xml2xml.exe` ein.

So richten Sie den LIM/XSLT-Patternmatcher ein

1. Erstellen Sie mit einem Texteditor eine Batchdatei (z.B. `meinProjekt4DImport.batch`) nach folgendem Schema:

Syntax
```
limInput2xml
projektVerzeichnis
Dateiendungen
Pfad zum LIM-Skript
Pfad zum XSLT-Skript
Verzeichnis, in dem die Zwischendateien gespeichert werden
target;
```

Beispiel
```
limInput2xml
D:/quellen/4D/src
4D
D:/Programme/shore/projects/4Dproject/4D2x4D.lim
D:/Programme/shore/projects/4Dproject/x4D2xml.xslt
D:/Programme/shore/projects/4Dproject/x4D
xml;
```

Alle Pfade sind absolut anzugeben.
Das `projektVerzeichnis` zeigt auf das Verzeichnis, unter dem die Sourcen stehen, die importiert werden sollen. Es werden die Dateien importiert, die einen in `Dateiendungen` angegeben Suffix besitzen. Der `Pfad zum LIM-Skript` und `Pfad zum XSLT-Skript` geben an, welche Skripten vom Parser verwendet werden sollen. Im `Verzeichnis, in dem die Zwischendateien gespeichert werden`, werden während des Parsens noch nicht SHORE-konforme XML-Dateien von der LIM-Verarbeitung abgelegt. Anschließend erweitert die XSLT-Verarbeitung dieses in SHORE-konformes XML. Der letzte Parameter ist das `target`. Dieses wird beim LIM/XSLT-Parser auf `xml` eingestellt.

Hinweis: Im Logfile wird für den XSLT-Prozessor immer ein return-code 200 ("Bad file descriptor") gemeldet, obwohl die Verarbeitung fehlerfrei abläuft. Dies scheint ein Bug im verwendeten XSLT-Prozessor zu sein. Sobald wir Rückmeldung vom Support haben, was diesen Fehler auslöst, versuchen wir ihn zu beheben... Ansonsten sind wir aber mit dem XSLT-Prozessor zufrieden ;-)

So importieren Sie Ihre Sourcen

1. Kopieren Sie Ihre Projektdateien in das `projektVerzeichnis` (z.B. `D:/quellen/4D/src`).
2. Importieren Sie die Batchdatei mit dem Kommandozeilen-Client: `shore import meinProjekt4DImport.batch`
   Details zum import-Befehl: Siehe “So importieren Sie Dokumente mit dem Kommandozeilen-Client” auf Seite 77.

Betreiben eigener Parser

Wenn Sie selbst (in der vorhandenen Skriptumgebung) weitere Parser entwikkeln und einbetten, dann ist es günstig, sich ein Skript zu schreiben, welches den dispatcher.exe mit einem definierten Ausgabeverzeichnis aufruft. Die in diesem Ausgabeverzeichnis liegenden XML-Dateien könnnen sie dann noch einmal inspizieren, bevor Sie sie nach SHORE importieren.

Parser in SHORE einbinden

Parser in SHORE installieren

Wenn Sie einen neuen Parser (oder eine neue Version eines Parsers) in SHORE einbinden wollen, dann können Sie dies über den Kommandozeilen-Client tun oder manuell machen. Eine Installation mit dem Kommandozeilen-Client können Sie nur mit einem Parser durchführen, der aus einem einzigen ausführbaren Programm besteht. Sollte Ihre Parser noch DLLs oder andere ausführbare Programme benötigen, die Sie mit installieren müssen, dann wählen Sie die manuelle Installation.

So installieren Sie einen Parser über den Kommandozeilen-Client

1. Wechseln Sie in der Eingabeaufforderung in das Verzeichnis, in dem der Parser liegt.
2. Geben sie den entsprechenden Befehl des Kommandozeilen-Clients ein.

Syntax imp_parser <parserdatei> <name> <endungen>

Beispiel Beispiel:
imp_parser xml2xml.exe dummy cbl java middle uc

3. Wenn der Parser installiert wurde, erscheint "ok".

Parser manuell installieren

Bei der manuellen Installation sind folgende Schritte zu tun:

1. Gegebenenfalls können Sie alle vorher importierten Dokumente manuell löschen. Dies kann aus Geschwindigkeitsgründen vorteilhaft sein.
2. Installieren Sie den Parser entsprechend der mitgelieferten Anweisung in ein Verzeichnis.
3. Kopieren Sie das Parserprogramm, das beim import aufgerufen werden soll, in das für die SHORE-Instanz gültige Verzeichnis. Das Standardverzeichnis ist `...shore\parser\bin`.
4. Falls Sie noch keine Dokumenttypen für den Parser angemeldet haben oder zusätzliche Dokumenttypen mit dem neuen Parser verarbeiten wollen, müssen Sie diese anmelden.

So melden Sie den Parser für Ihre Dokumenttypen über das Browserfenster bei SHORE an.

1. Wählen Sie im Menü „Administration“ den Eintrag „Dokumentenspezifische Parser...“.
   Das Fenster „Dokumentenspezifische Parser anmelden“ öffnet sich.
2. Klicken Sie auf den Button „Neu“, um einen neuen Parser anzumelden.
   Der untere Teil des Fensters wird für die Eingabe freigegeben.
3. Geben Sie im Feld „Name“ den Namen des Parsers ein.
   Beispieldummy
4. Geben Sie im Feld „Pfad zum Parser-Programm“ den Pfad und Dateinamen des Parsers ein.
   Beispielxml2xml.exe
5. Geben Sie im Feld „verantwortlich für Dateiendungen“ die Kürzel der Dateiendungen ein. Diese können mit Leerzeichen, Komma oder Semikolon getrennt werden.
   Beispielcbl java middle uc
6. Klicken Sie auf „Übernehmen“. Der Parser erscheint im Bereich „Auswahl“ und ist jetzt angemeldet.
7. Klicken Sie auf „OK“.

Das Standardmetamodell für Programmiersprachen

Zusammen mit SHORE erhalten Sie ein Metamodell, das die Definitionen der Dokument-, Objekt und Beziehungstypen für die verfügbaren Parser enthält (`.../shore/parser/meta/general.meta`). In den nachfolgenden Abbildungen sind für die Programmiersprachen C, C++, Java, Cobol und XCobol die in der jeweiligen Sprache enthaltenen wichtigsten Objekt- und Beziehungstypen mit Ihren Vererbungsbeziehungen abgebildet.

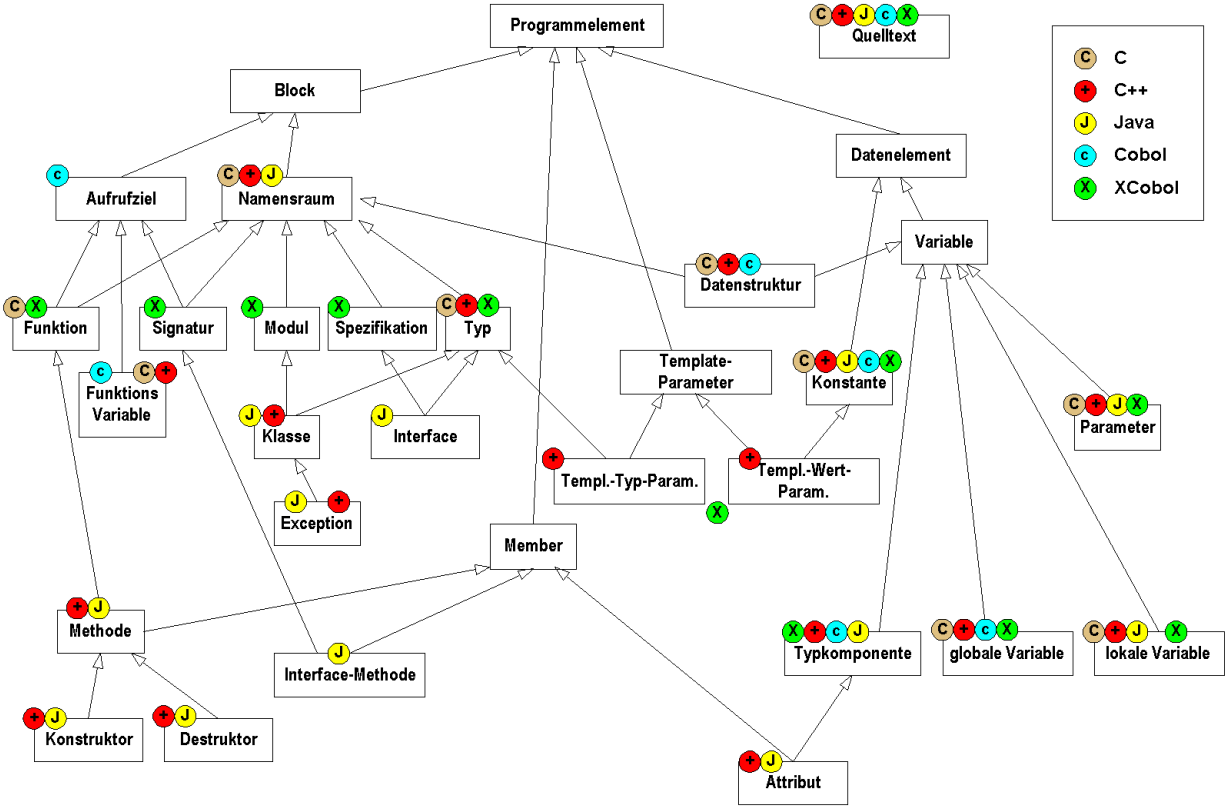


Abbildung 10: SHORE-Metamodell für Programmiersprachen: Objekttypen

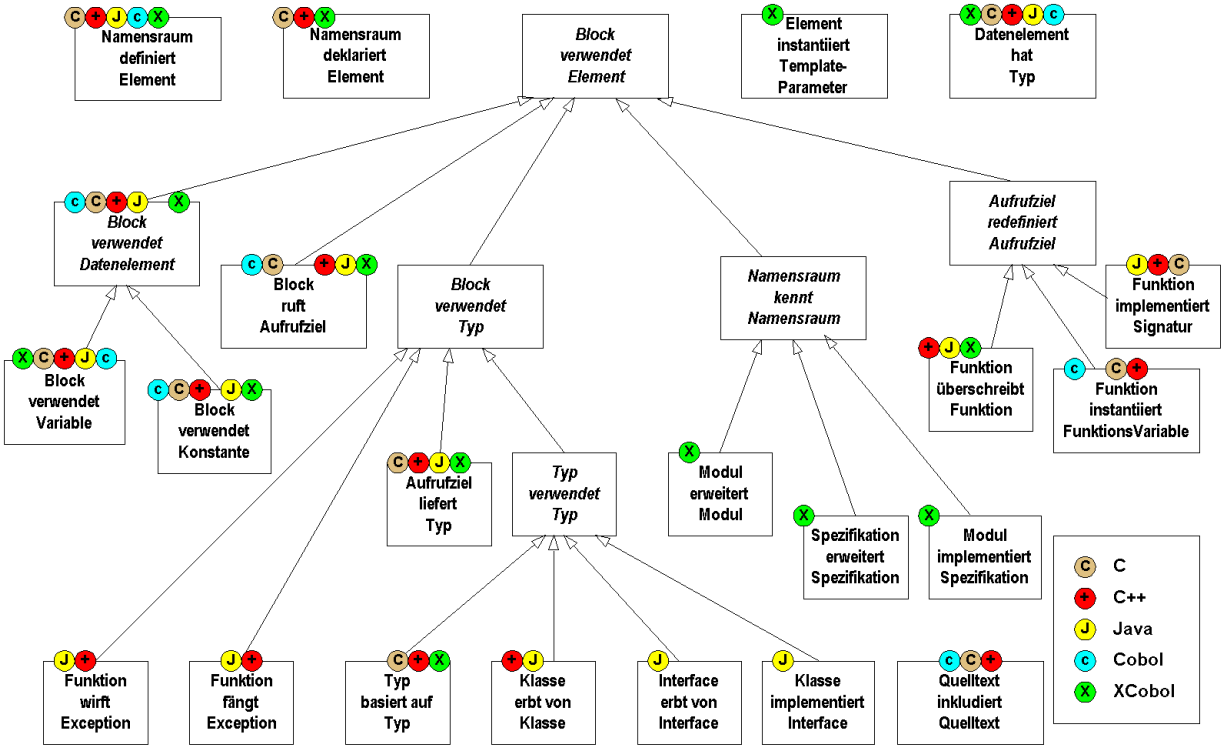


Abbildung 11: SHORE-Metamodell für Programmiersprachen: Beziehungstypen

Für Ihr Projekt können Sie der Übersichtlichkeit halber Teile des Metamodells löschen, die Sie nicht benötigen. Achten Sie dabei darauf, dass die Definitionen insgesamt konsistent bleiben.

Anpassen eines LIM/XSLT-Scanners

Einen LIM/XSLT-Scanner können Sie für relativ einfach zu erkennende Muster einsetzen. Dieser Scanner arbeitet zweistufig, d.h. dass zuerst mit der Programmiersprache LIM Muster erkannt werden und einfaches XML erzeugt wird. Danach erweitert XSLT das XML so, dass es in SHORE importiert werden kann.

Hier ist eine Übersicht, welche Arbeitsschritte notwendig sind. Eine detaillierte Beschreibung wie Sie einen Parser mit LIM/XSLT erstellen, ist im Parserkochbuch beschrieben.

Übersicht: Schritte zum Erstellen eines LIM/XSLT-Scanners

1. Richten Sie sich eine lokale Arbeitsumgebung ein, in der zum Testen Sourcen, Skripten, XML-Dateien und Logfiles gespeichert werden.
2. Richten Sie sich möglichst eine SHORE-Testinstanz auf Ihrem eigenen PC ein.
3. Melden Sie `xml2xml.exe` für die Dateiendungen der zu importierenden Dateien an.
4. Überlegen Sie sich, was Sie erkennen wollen und identifizieren Sie die entsprechenden Elemente im Quelltext.
5. Besorgen Sie sich eine entsprechende Grammatik oder definieren Sie z.B. in Bachus-Naur-Notation die Struktur der zu erkennenden Elemente.
6. Definieren Sie sich, wie ein Metamodell aussehen soll und erstellen oder erweitern Sie ein Metamodell.
7. Erstellen oder erweitern Sie ein LIM-Skript anhand der Grammatik und des Metamodells.
8. Erstellen oder erweitern Sie ein XSLT-Skript passen zum LIM-Skript. Das erzeugte XML muss zum Metamodell passen.
9. Importieren Sie das Metamodell in SHORE.
10. Importieren Sie probeweise die erzeugten XML-Dateien.
11. Wiederholen Sie die Schritte 4 bis 10 solange, bis Sie zufrieden sind.
12. Dann erstellen Sie eine `*.batch`-Datei zum direkten Import in SHORE.
13. Melden Sie `ImportBatch.exe` als Parser an.
14. Importieren Sie Ihre `*.batch`-Datei.

So richten Sie für das Erstellen eines LIM/XSLT-Scanners eine Arbeitsumgebung ein

Überprüfen sie, dass das Parserframework installiert ist (Siehe “Das Parserframework” auf Seite 37.).

1. Legen Sie sich ein Verzeichnis an, welches Ihr SHORE-Arbeitsverzeichnis wird. z.B. `D:\SHOREwork`

2. Legen Sie sich darunter Verzeichnisse für
   - Ihre Sourcen (z.B. `D:\SHOREwork\src`),
   - für Zwischendateien, die von LIM erzeugt werden (z.B. `D:\SHORE-work\xSrc`) sowie
   - für XML an (z.B. `D:\SHOREwork\xml`).

4.1.2 Verzeichnisse

Die Programme und Daten von SHORE stehen in unterschiedlichen Verzeichnissen. Die Verzeichnisstruktur der projektspezifischen Dateien können Sie über den Browser ändern (Siehe “Einstellen der Projektdokumentablage” auf Seite 60.).

Instanzunabhängige Verzeichnisse

Nachfolgend finden Sie die Verzeichnisse und deren Bedeutung, die unterhalb des SHORE-Verzeichnisses (z.B. `D:\shore`) bei der Installation als Standard eingestellt wurden:

Verzeichnis	Erläuterung
`...\shore\`	Das Hauptverzeichnis auf das die Umgebungsvariable SHORE zeigen muss.
`...\shore\bin`	Ausführbare Dateien des SHORE-Servers, z.B. `shore.exe`
`...\shore\client\`	SHORE-Kommandozeilenclient und dessen Hilfsdateien dazu.
`...\shore\conf\`	Konfigurationsdateien für Benutzerkonten
`...\shore\data\`	Dieses Verzeichnis wird vom SHORE-Server als Hauptverzeichnis für HTTP-Requests verwendet. Die SHORE-Leitseite `index.html` liegt hier, sie verweist auf Dateien im Unterverzeichnis html. Weiterhin werden hier je nach Konfiguration die jpg-Dateien angelegt, die beim Import von Ergebnisdokumenten anfallen.
`...\shore\data\css\`	Style Sheets für den SHORE-Server
`...\shore\data\html\`	Statische HTML-Dateien für die Leitseiten
`...\shore\data\html\pictures\`	Hier werden statische Bilder gespeichert, z.B. für die Leitseite.
`...\shore\data\java\`	Java-Klassen und Hilfsdateien für die Java-Applets des SHORE-Systems
`...\shore\data\xml\`	Statische XML-Dateien für die Leitseiten. Dieses Verzeichnis ist für zukünftige XML-fähige Browser vorgesehen, es ist derzeit leer.

Verzeichnis	Erläuterung
`...\shore\docstore\`	Hier legt SHORE die importierten Daten ab. Unter diesem Verzeichnis existiert pro Instanz ein Verzeichnis.
`...\shore\log\`	Hier liegen die Dateien `parser.log` und `server-<PID>-log`. Die Standardausgabe der Skriptumgebung wird hierhin umgelenkt und laufende `shore.exe` schreiben hier ihre Meldungen jeweils in ein Logfile (unterschieden nach der Process-ID). SHORE legt hier auch die Logdateien ab, die z.B. alle Requests und Imports bezeichnen.
`...\shore\projects\`	Standard-Verzeichnis für Projektdaten. Jedes Projekt sollte unter diesem Verzeichnis ein eigenes Unterverzeichnis haben.
`...\shore\projects\example`	Verzeichnis, in dem Beispiele zu den Parsern stehen.
`...\shore\projects\tutorial`	Verzeichnis, in dem Aufgaben und Lösungen aus dem Parser-Kochbuch zum Pattern-Matching stehen.
`...\shore\parser\bin`	Standard-Verzeichnis für die ausführbaren Parser-Programme.
`...\shore\parser\lib`	Verzeichnis, unter dem das Parserframework und andere Parserspezifische Dateien liegen.
`...\shore\store\`	Speicherung der Datenbanken für die Instanzen.
`...\shore\tmp\`	Temporäre Dateien für den Import. Hier bleiben bei bestimmten Programmabbrüchen der Parser Verzeichnisse stehen, die regelmäßig von Hand zu löschen sind.
`...\shore\verity\`	Installation der verity-Suchmaschine.
`...\shore\xsb\`	Verzeichnis und Unterverzeichnisse für den Prolog-Interpreter.
`...\shore\xsb\rules\`	Hier werden benutzerdefinierte Regeln (für Abfragen) abgespeichert.
`...\shore\xsb\syslib\tmpParsedFiles\`	Temporärer Zwischenspeicher für XML-Dateien während des Parsens. Bei bestimmten Programmabbrüchen während des Parsens (insbesondere bei fehlerhaft erzeugtem XML) bleiben hier Dateien stehen, die von Hand zu löschen sind.

Instanzabhängige Verzeichnisse

Wenn eine Instanz eingerichtet ist, gibt es noch weitere Verzeichnisse

Von SHORE verwaltete Verzeichnisse

In diesen Verzeichnissen legt SHORE Daten einer Instanz ab. Hier sollten Sie möglichst nichts ändern, es sei denn, Sie wissen, was Sie damit anrichten ;-)

Verzeichnis	Erläuterung
...\shore\docstore\<myDb-name>\css\	Verzeichnis für importierte projektspezifische Stylesheets. Dieses Verzeichnis wird von SHORE verwaltet.
...\shore\docstore\<myDb-name>\doc\	Verzeichnis für Kopien der Originaldokumente. Dieses Verzeichnis wird von SHORE verwaltet.
...\shore\docstore\<myDb-name>\html\	Verzeichnis für von SHORE aus den XML-Dateien erzeugte HTML-Dateien. Dieses Verzeichnis wird von SHORE verwaltet und wird nur gefüllt, wenn eine Suchmaschine installiert ist.
...\shore\docstore\<myDb-name>\index\	Verzeichnis für den Suchmaschinenindex. Pro Instanz wird eine "collection" angelegt, die als Namen eine Nummer erhält (die Nummer entspricht dem create-Timestamp der zugehörigen ObjectStore-Datenbankinstanz). Dieses Verzeichnis wird von SHORE verwaltet und wird nur gefüllt, wenn eine Suchmaschine installiert ist.
...\shore\docstore\<myDb-name>\xml\	Verzeichnis für XML-Dateien, die aus den importierten Ergebnisdokumenten erzeugt worden sind. Die XML-Dateien haben den selben Dateinamen (einschliesslich Dateityp) wie die Originaldokumente, obwohl sie grundsätzlich im XML-Format sind. Dieses Verzeichnis wird von SHORE verwaltet.

Projektabhängige Verzeichnisse

In diesen Verzeichnissen können Sie Ihre Projektdaten ablegen und z.B. auch Skripten, die den Import steuern.

Verzeichnis	Erläuterung
...\shore\projects\<myProject>	Von Ihnen selbst zu erstellendes Verzeichnis, in dem Sie z.B. Skripten zum Start eines Imports oder Skripten zum Parsen ablegen können. Sie können mehrere Projektverzeichnisse anlegen, die nicht 1:1 mit den Instanz-Verzeichnissen abbildaber sein müssen.
...\shore\projects\<myProject>\css\	Von Ihnen selbst zu erstellendes Verzeichnis, in dem Sie projektspezifische Stylesheets ablegen können.
...\shore\projects\<myProject>\doc\	Von Ihnen selbst zu erstellendes Verzeichnis, unter dem Sie Ihre Originaldokumente ablegen sollten.

Verzeichnis	Erläuterung
...\shore\projects\<myProject>\xdoc\	Verzeichnis, in der LIM/XSLT-Pattern-Matcher mit einfachem XML angereicherte Originaldokumente speichert, die aber noch nicht in SHORE-konformem XML sind. Dieses Verzeichnis legt SHORE nur dann an, wenn Sie LIM/XSLT-Pattern-Matching verwenden.
...\shore\projects\<myProject>\xmlm\	Verzeichnis für xmlm-Dateien. Dieses Verzeichnis legt SHORE nur dann an, wenn der Parser das XML mittels des XML-Mergers erzeugt.
...\shore\projects\<myProject>\xml\	Verzeichnis für XML-Dateien, die aus den importierten Ergebnisdokumenten erzeugt worden sind. Die XML-Dateien haben den selben Dateinamen (einschliesslich Dateityp) wie die Originaldokumente, obwohl sie grundsätzlich im XML-Format sind. Bei Verwendung von SHORE-Parsern wird dieses Verzeichnis automatisch angelegt. Sie können es auch manuell anlegen wenn Sie direkt XML imporieren wollen.

4.1.3 Die ausführbaren Programme von SHORE

Im Verzeichnis `shore\bin` und `shore\client` stehen die ausführbaren Programme von SHORE (`*.exe` und `*.cmd`). Die Programme, die Sie direkt starten können, sind in den folgenden Kapiteln beschrieben.

Die restlichen Programme werden aus den gestarteten Programmen heraus aufgerufen. Details dazu finden sie im Kapitel “laufende Prozesse überwachen” auf Seite 72.

shore.exe

Dieses Programm wird in der Regel ausgeführt, wenn ein SHORE-Server gestartet werden soll. Es befindet sich im Verzeichnis `shore\bin`.

Dieses Programm startet `shore-server.exe` (den eigentlichen SHORE-Server) und wacht darüber, dass dieser verfügbar ist. Falls der Prozess `shore-server.exe` abstürzen sollte, startet `shore.exe` sofort einen neuen `shore-server.exe`-Prozess.

Als Parameter können Sie einen Datenbanknamen übergeben. Falls kein Datenbankname übergeben wird, wird als Standard die Datenbank `shore.db` im Verzeichnis `shore\store` verwendet

So starten Sie SHORE mit einer spezifischen Datenbank als Parameter

1. Öffnen Sie ein DOS-Fenster und gehen Sie ins Verzeichnis `shore\bin`.
2. Starten Sie `shore.exe` mit dem Datenbanknamen als Parameter.

Syntax shore.exe <DB-Name>
Beispiel shore.exe instanz7.db

Damit startet SHORE und benutzt die übergebene Datenbank.

shore-server.exe

Das Programm `shore-server.exe` ist das Herzstück von SHORE. Es befindet sich im Verzeichnis `shore\bin`.
`shore-server.exe` übernimmt die Kommunikation mit den Clients und steuert sämtliche Aktionen wie beispielsweise Im- und Exports oder Anfragen.

In unvorhergesehenen Fehlersituationen kann `shore-server.exe` abstürzen. Für einen Dauerbetrieb sollte daher `shore.exe` ausgeführt werden.

Auch hier können Sie als Parameter (wie bei `shore.exe`) einen Datenbanknamen übergeben.

Kommandozeilen-Client

Mit dem Kommandozeilen-Client `shore.cmd` steuern Sie das laufende SHORE-System. Es befindet sich im Verzeichnis `shore\client`.
Der Kommandozeilen-Client hat viele Parameter, die Sie je nach Aufgabe einsetzen.

So rufen Sie den Kommandozeilen-Client auf

1. Öffnen Sie ein DOS-Fenster und stellen Sie sicher, dass Sie sich nicht im Verzeichnis `shore\bin` befinden.
2. Geben Sie das Kommando `shore` mit den entsprechenden Parametern ein. Eine Übersicht aller Parameter finden Sie im Anhang oder im DOS-Fenster, wenn Sie `shore` ohne Parameter eingeben.

4.1.4 Von SHORE gestartete Prozesse

Im Taskmanager können Sie die laufenden Prozesse eines Rechners ansehen.

SHORE startet folgende Prozesse

Name	Erläuterung	wird gestartet von:
shore.exe	Wächterprozess des SHORE-Servers. Dieser Prozess startet shore-server.exe und wacht darüber, dass dieser verfügbar ist. Falls der Prozess shore-server.exe abstürzen sollte, startet shore.exe sofort einen neuen shore-server.exe-Prozess.	Kommandozeile/ System (Dienst)
shore-server.exe	Der SHORE-Server-Prozess. Dieser Prozess beinhaltet den integrierten Webserver. shore-server.exe beantwortet HTTP-Anfragen und startet je nach HTTP-Request gegebenenfalls weitere Prozesse. Das Ergebnis dieser Prozesse wird von shore-server.exe per HTTP zurückgeliefert.	shore.exe oder Kommandozeile/ System (Dienst)
shore-update.exe	Dokumentim- oder export. Führt alle Aktionen aus, um Dokumente in SHORE zu importieren oder zu exportieren.	shore-server.exe
mkvdk.exe	Dienst der Suchmaschine zur Indexpflege. Wird beim Im- oder Export gestartet oder wenn ein reindex ausgeführt wird. mkvdk.exe pflegt den Suchmaschinenindex.	shore-update.exe
osserver.exe	DB-Manager von ObjectStore.	System (Dienst)
shore-xsb.exe	Prolog-System. Beantwortung von Prolog-Anfragen.	shore-server.exe
s97cgi.exe	Beantwortung von Suchanfragen. Wird bei einer Volltextsuche ausgeführt um Suchanfragen auf den Suchmaschinenindex auszuführen.	shore-server.exe

4.1.5 SHORE konfigurieren

Für alle folgenden Schritte ist es erforderlich, dass Sie als lokaler Systemverwalter (Administrator) angemeldet sind.

Einrichten als Dienst

Wenn Sie SHORE als Dienst einrichten, wird der Server automatisch beim Systemstart hochgefahren und steht Ihnen immer zur Verfügung. Folgende Dinge sind zu tun:

- Legen Sie einen neuen Dienst an und konfigurieren Sie diesen anschließend in der NT Registry.

- Installieren Sie ein "shutdown"-Skript.
  Setzen Sie ein Skript ein, das zuerst SHORE stoppt, dann den Dienst beendet und dann ein shutdown für Windows NT durchführt. Solch ein Skript existiert schon, fragen Sie das Werkzeugteam.
  Hintergrund: Wenn SHORE als Dienst gestartet wird, dürfen Sie Windows NT nicht einfach hinunterfahren, sondern Sie müssen SHORE vorher ordnungsgemäß beenden. Anderenfalls kann die Datenbank in einem inkonsistenten Zustand stehenbleiben. Wenn Sie dann das nächste Mal SHORE starten, wird die Konsistenz wieder hergestellt, was einige Minuten dauern kann. Diese Wartezeit kann lästig werden und wird durch ein geordnetes beenden von SHORE vermieden.

Um SHORE als Dienst einrichten zu können, sind zwei Programme aus dem NT Resource Kit notwendig: `INSTSRV.EXE` und `SRVANY.EXE`. Mit dem Programm `SRVANY.EXE` können beliebige Programme als Dienst ausgeführt werden. Es wird also `SRVANY.EXE` als Dienst angemeldet und `SRVANY.EXE` ruft dann beim Start `shore.exe` auf. Die Installation eines Dienstes wird mit `INSTSRV.EXE` ausgeführt.

Hinweis: Statt des Programmes `INSTSRV.EXE` können Sie alternativ auch das Programm `SERINSTW.EXE` benutzen, das eine grafische Oberfläche bietet. Die Benutzung des Programmes ist hier nicht weiter erklärt und Sie müssen in der uns vorliegenden Version ebenfalls - wie nachfolgend beschrieben - Parameter in der Registry per Hand hinzufügen.

Für SHORE richten Sie den Windows NT Befehltsinterpreter (`CMD.EXE`) mit Hilfe der oben genannten Programme als Dienst ein.
Besorgen Sie sich das neueste verfügbare Windows NT Resource Kit (zum Zeitpunkt wo dies verfasst wird, ist dies das Microsoft® Windows NT® Server 4.0 Resource Kit Supplement Four
Dieses bekommen Sie bei Ihrer Systemverwaltung oder aus dem Internet (`http://www.microsoft.com`)

So richten Sie einen Dienst unter NT ein

1. Kopieren Sie aus dem Ressource Kit die Dateien `instsrv.exe` und `srvany.exe` in das Verzeichnis `...\shore\bin`, oder in Ihr Verzeichnis, in dem die ausführbaren Dateien von SHORE liegen.
2. Öffnen Sie ein DOS-Fenster und wechseln Sie in das Verzeichnis `...\shore\bin` oder in das Verzeichnis in dem die ausführbaren Dateien von SHORE liegen.
   `Beispiel:cd D:\shore\bin`
3. Legen Sie einen neuen Dienst an, indem sie `INSTSRV.EXE` mit seinen entsprechenden Parametern aufrufen.
   Als Beispiel soll der Dienst "SHORE-Instanz 1" eingerichtet werden.

Syntax instsrv "<Name des neuen Dienstes>" <Pfad zur Datei srvany.exe>
Beispiel instsrv "SHORE-Instanz 1" D:\shore\bin\srvany.exe

Tipp: Als Dienstname wählen Sie sinnvollerweise nicht nur den Instanznamen, sondern eine Konkatenation aus SHORE und dem Instanznamen (z.B. "SHORE Instanzname 1", "SHORE Instanzname 2" usw.), so dass in der Anzeige der Dienste die SHORE-Instanzen zusammen aufgelistet werden.

So Stellen Sie in der Windows NT Registry ein, welches Programm von SRVANY.EXE ausgeführt werden soll

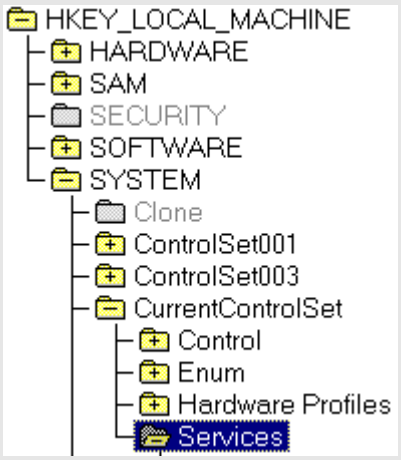
1. Starten Sie den Registry-Editor `regedt32.exe`(z.B. unter Start>Ausführen).

Achtung: Beim Bearbeiten der Windows NT Registry müssen Sie äusserst vorsichtig sein, da falsche Einträge in der Registry den Rechner in einen undefinierten Zustand versetzen können, in dem der Rechner u.U. nicht mehr startet.

2. Aktivieren Sie das Fenster „HKEY_LOCAL_MACHINE auf lokalem Computer“.

3. Öffnen Sie in der linken Fensterhälfte jeweils durch Doppelklick nacheinander die Ordner „SYSTEM“, „CurrentControlSet“, „Services“ und den Namen Ihrer Instanz (z.B. „SHORE-Instanz 1“).

Achtung: In den folgenden Schritten ersetzen Sie den Namen "SHORE-Instanz 1" jeweils mit dem Namen Ihrer Instanz.



4. Markieren Sie den Eintrag mit dem Namen „SHORE-Instanz 1“ in der linken Fensterhälfte.
5. Tragen Sie eine Abhängigkeit ein, damit SHORE erst gestartet wird, wenn der Dienst des ObjectStore-Servers hochgefahren ist.
6. Wählen Sie im Menü „Bearbeiten“ den Eintrag „Wert hinzufügen...“.
7. Tragen Sie im Feld „Wertname“ die Zeichenkette „`DependOnService`“ und wählen Sie als „Datentyp“ „`REG_MULTI_SZ`“ aus.
   Bestätigen Sie mit „OK“.
8. Wählen Sie im Menü „Bearbeiten“ den Eintrag „Schlüssel hinzufügen...“.
9. Tragen Sie im Feld „Schlüsselname“ den Wert „`Parameters`“ ein. Das Feld „Klasse“ bleibt leer.
10. Bestätigen Sie mit „OK“.
    Jetzt müssen im Ordner "SHORE-Instanz 1" zwei Unterordner, nämlich "Parameters" und "Security" vorhanden sein
11. Markieren Sie in der linken Fensterhälfte den Eintrag mit dem Namen „Parameters“ im Ordner "SHORE-Instanz 1".
12. Wählen Sie im Menü „Bearbeiten“ den Eintrag „Wert hinzufügen...“.
13. Tragen Sie im Feld „Wertname“ den Wert „`AppDirectory`“ ein und wählen Sie als „Datentyp“ „`REG_SZ`“ aus und
    Bestätigen Sie mit „OK“.
14. Geben Sie in dem nun aufgeblendeten Zeichenketten-Editor im Feld „Zeichenkette“ das Verzeichnis, in dem Sie SHORE installiert haben, ein (z.B. `D:\Programme\shore`).
    Bestätigen Sie mit „OK“.

15. Wiederholen Sie die Schritte 12. bis 14. für die Einträge der folgenden Liste:
    - „Wertname“: „Application“
      „Datentyp“: „REG_SZ“
      „Zeichenkette“: „D:\Programme\shore\bin\shore.exe“ (bzw. den Pfad ihrer Installation, der auf shore.exe zeigt)
    - „Wertname“: „AppParameters“
      „Datentyp“: „REG_SZ“
      „Zeichenkette“: „dbname.db“

    In der rechten Fensterhälfte stehen jetzt folgende Einträge:

```
AppDirectory :REG_SZ :D:\Programme\shore
Application :REG_SZ :D:\Programme\shore\bin\shore.exe
AppParameters :REG_SZ :dbname.db
```

16. Geben Sie in dem nun aufgeblendeten Fenster „Editor für mehrteilige Zeichenketten“ den Namen des Dienstes „ObjectStore Server R4.0“ ein. Bestätigen Sie mit „OK“.
17. Beenden Sie den Registry Editor.
    Klicken Sie dazu im Menü „Registrierung“ auf „Beenden“.

So stellen Sie in der Systemsteuerung die Startart Ihres Dienstes ein

1. Starten Sie die Systemsteuerung über Start->Einstellungen->Systemsteuerung.
   Das Fenster „Systemsteuerung“ öffnet sich.
2. Starten Sie das Programm `Dienste` durch Doppelklick.
3. Blättern Sie in der Liste, bis Sie den Eintrag Ihres neu eingerichteten Dienstes (in unserem Beispiel "SHORE-Instanz 1") sehen. Die „Startart“ für den neu eingerichteten Dienst soll „Automatisch“ sein.
   Falls das nicht der Fall ist, markieren Sie die Zeile des Dienstes und klicken Sie auf den Button „Startart...“. Wählen Sie „Automatisch“ und bestätigen Sie mit „OK“.

   Hinweis: Bei einigen Instanzen kann es erforderlich sein, dass Sie den Dienst unter einem anderen Konto als dem Systemkonto zu starten. Dies kann z.B. erforderlich sein, wenn Einstellungen pro Benutzer verwaltet werden müssen oder wenn Sie Zugriff auf ein Netzlaufwerk benötigen.
   Das Konto wird ebenfalls über „Startart...“ eingestellt. Dabei muss das Passwort des entsprechenden Benutzers eingegeben werden.
4. Starten Sie den Dienst, indem Sie den Button „Starten“ klicken.
5. Überprüfen Sie, nach dem erfolgreichen Start, mit dem Browser, ob die neu eingerichtete Instanz erreichbar ist und die erwarteten Daten enthält.
6. Verlassen Sie das Programm `Dienste` über „Schliessen“.
7. Entfernen Sie ggf. eine noch im Rechner steckende Diskette aus dem Laufwerk.
8. Starten Sie den Rechner neu, und überprüfen Sie, ob SHORE automatisch startet.

Einstellen der Projektdokumentablage

Sämtliche in SHORE importierten Dokumente sind in der sogenannten Projektdokumentablage gespeichert. Dies sind mehrere Verzeichnisse, die SHORE für seinen Betrieb benötigt. Bei der Installation werden bereits Verzeichnisse angelegt, die Sie individuell ändern können bzw. können Sie zusätzliche Verzeichnisse pro Instanz hinzufügen.

Die physischen Pfade der Projektdokumentablage können Sie individuell ändern, insbesondere dann, wenn Sie mehrere Instanzen auf einem Server betreiben wollen. Hier finden Sie die Standardeinstellungen, dabei liegen alle Verzeichnisse unter dem SHORE-Verzeichnis.

Wurzelverzeichnis der Projektdokumentablage	docstore\projekt\doc\
Wurzelverzeichnis der XML-Dokumentablage	docstore\projekt\xml\

Verzeichnis für den HTML-Export	docstore\projekt\html\
Verzeichnis für den Suchmaschinenindex	docstore\projekt\index\
Verzeichnis der Stylesheet-Ablage	docstore\projekt\css\
Standardverzeichnis für Dokumenttyp-spezifische Parser	parser\bin\

Achtung: wenn Sie diese Pfade ändern, müssen Sie anschließend einen kompletten Neuimport aller Dokumente durchführen, da die Datenbank noch die alten Pfade gespeichert hat. Es empfiehlt sich außerdem, vor dem Neuimport die Dateien und Verzeichnisse innerhalb der Dokumentablage komplett zu löschen, da die alten Dateien beim Ändern der Verzeichnisse nicht gelöscht.

Wenn Sie mehrere Instanzen betreiben wollen, dann bauen Sie sich möglichst eine analoge Struktur auf, wobei sie das Verzeichnis `projekt` durch den Namen ihrer Instanz ersetzen.

So ändern Sie die Einstellungen für die Pfade im Browserfenster

1. Starten Sie in Ihrem Browser SHORE.
2. Wählen Sie im Menü Administration > Ändere Einstellungen ...
   Das Fenster „Servereinstellungen bearbeiten“ öffnet sich.

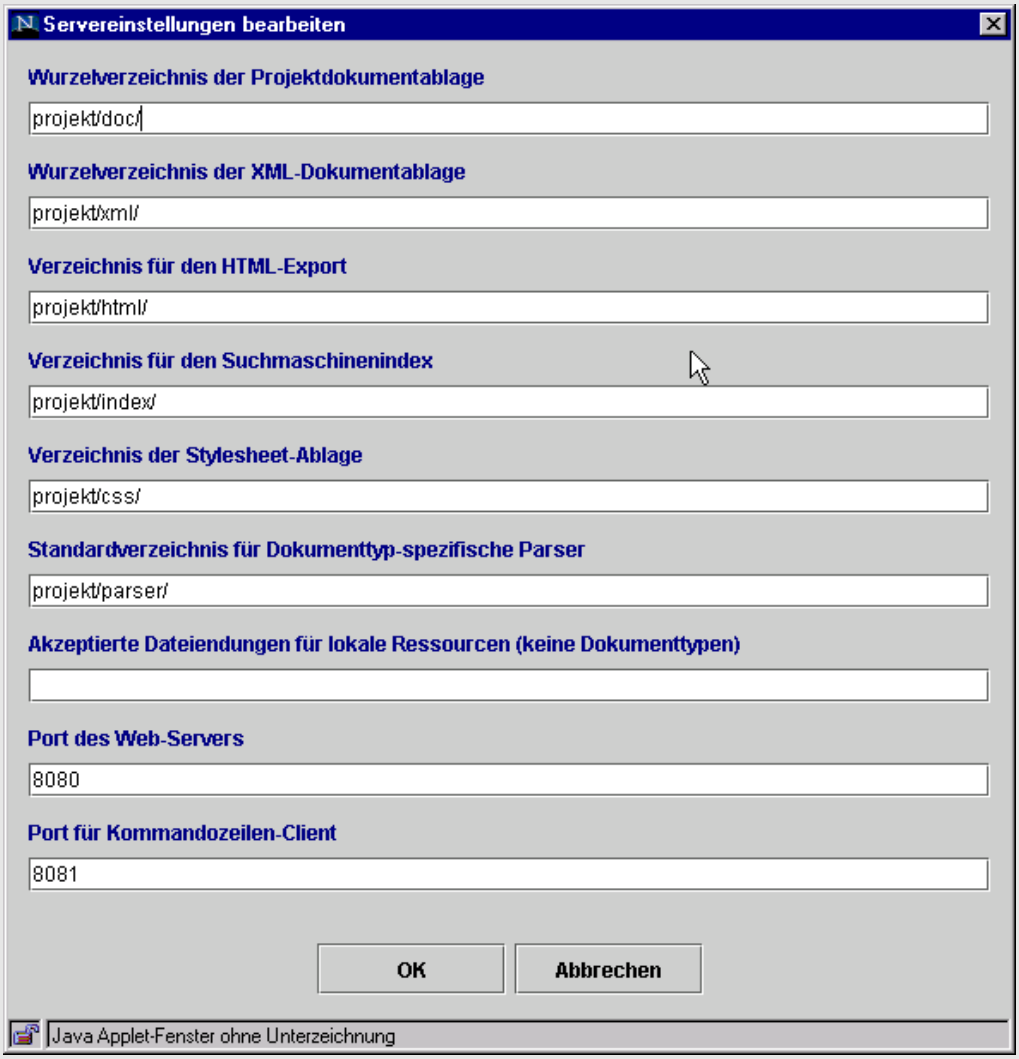


Abbildung 12: Dialog Servereinstellungen bearbeiten

3. Ändern Sie die Pfade wie gewünscht und klicken Sie auf „OK“

Hinweis: Der Eintrag „Port für Komanndozeilen-Client“ wird zur Zeit nicht verwendet. Änderungen an diesem Eintrag haben also keine Auswirkung.

Mehrere Instanzen auf einem Server einrichten

Sie können auf einem NT-Server mehrere Instanzen von SHORE installieren.

Jede Instanz braucht ihren eigenen Port, ihre Verzeichnisse und Datenbank. Diese Einstellung muss auch beim Kommandozeilen-Client gemacht werden (siehe Kapitel “Nutzen mehrerer Instanzen von einem Client aus” auf Seite 87).

So planen Sie die Einrichtung einer neuen Instanz

1. Legen Sie einen Namen für die Datenbank-Datei für die neue Instanz fest. Beispiel: Die neue Instanz soll "rea2" heißen.
2. Legen Sie fest, über welchen Port die neue Instanz erreicht werden soll, da jede Instanz Ihre eigene Portnummer benötigt.
   Der Standardport von SHORE ist 8080.
   Für unser Beispiel wählen Sie die Portnummer 8800.

Tipp: Theoretisch können Sie jeden Port zwischen 0 und 65535 nutzen,
Details dazu stehen in der RFC 1700 (siehe Link im Kapitel “Literatur” auf Seite 123).
Wählen sie am besten Portnummern, die noch nicht reserviert sind (z.B. sind zum Zeitpunkt wo dies verfasst wird die Nummern 8766 bis 8803 nicht zugeordnet).

Tipp: Jede neue Instanz ist zunächst über den Port 8080 zu erreichen. Diesen Port sollten sie möglichst immer freihalten, dann können Sie auch im laufenden Betrieb neue Instanzen einrichten.

3. Legen Sie fest, wo sie die Dokumente in SHORE speichern wollen (siehe dazu auch das Kapitel “Einstellen der Projektdokumentablage” auf Seite 60).
   Eigene Verzeichnisse müssen Sie pro Instanz anlegen für:
   - das Wurzelverzeichnis der Projektdokumentablage (`<projekt>\doc\`)
   - das Wurzelverzeichnis der XML-Dokumentablage (`<projekt>\xml\`)
   - das Verzeichnis für den HTML-Dokumentablage (`<projekt>\html\`)

Hinweis: Das Verzeichnis für den Suchmaschinenindex (`<projekt>\index\`) muss nicht getrennt werden, da automatisch für jede Instanz ein eigener Index erstellt wird. Der Klarheit wegen kann trotzdem ein anderes Verzeichnis eingestellt werden.

4. Die folgenden Verzeichnisse können von mehreren Instanzen genutzt werden, können aber auch getrennt werden:
   - das Verzeichnis der Stylesheet-Ablage (`<projekt>\css\`)
   - Falls Sie pro Instanz unterschiedliche Parser einsetzen wollen (beispielsweise unterschiedliche Word-Patternmatcher), müssen Sie ebenfalls ein eigenes Verzeichnis festlegen. Das Standardverzeichnis für Dokumenttyp-spezifische Parser lautet(`parser\bin\`)

Hinweis: Das Anlegen unterschiedlicher Parserverzeichnisse ist nur nötig, um das Standardverhalten pro Dateiendung festzulegen. Sie können beim Import der Dokumente nach SHORE auch den zu verwendenen Parser als Parameter angeben.

Für unser Beispiel legen Sie alle instanzspezifischen Daten unter `...shore\rea2\` ab.

So richten Sie eine neue Instanz ein:

1. Stoppen Sie alle laufenden SHORE-Server, die unter Port 8080 laufen.
2. Starten Sie in der Eingabeaufforderung einen SHORE-Server mit dem für die neue Instanz festgelegten Datenbanknamen als Parameter.
   Beispiel:`shore rea2.db`
   Diesen Namen müssen Sie sich merken, da Sie ihn bei jedem Start der Instanz als Parameter angeben müssen.

Tipp: Am besten, Sie nutzen ein Skript oder eine Batch-Datei, die den SHORE-Server mit dem Datenbanknamen als Parameter startet.

3. Starten Sie einen Browser und geben sie die URL ihres SHORE-Servers ein.
   Zum Beispiel: unser Rechner heißt "kailua" und befindet sich in der Domäne "sdm.com".

Syntax http://<Server-Adresse>:<Port>/shore/
Beispiel http://kailua.sdm.com:8080/shore
Warten Sie, bis SHORE im Browser geladen ist.

4. Wählen Sie im Menü „Administration > Ändere Einstellungen...“.
   Das Fenster „Servereinstellungen bearbeiten“ öffnet sich.
5. Tragen Sie die für die neue Instanz von Ihnen festgelegten Pfade und die Portnummer des Web-Servers ein.
6. Klicken Sie auf OK.
   Ab sofort ist bei einem neuen Aufruf der neue Port aktiv.

Tipp: Im Browser bekämen Sie jetzt bei einem Reload einen Fehler, da die Portnummer geändert wurde. Die neue URL für unser Beispiel lautet nun `http://kailua.sdm.com:8800/shore/`

Sie haben eine neue Instanz eingerichtet, die sie immer über ihren Datenbanknamen (z.B. `rea2.db`) als Parameter des SHORE-Servers starten.

Tipp: Wenn Sie mehrere Instanzen eingerichtet haben, können Sie sich eine zentrale Seite schreiben, von der Sie die einzelnen SHORE-Instanzen dann aufrufen können. Auf dieser zentralen Seite können z.B. Informationen über den letzten Vollimport stehen.
Benennen Sie die Seite index.html im Verzeichnis `...shore\data` um, und speichern Sie Ihre neue Seite als `index.html`. In die neue Seite setzen sie einen link auf die umbenannte Seite.

Individuelle Startseite pro Instanz einrichten

Sie möchten nach dem Laden von SHORE statt des Bilds, auf dem ein Surfer in der Welle zu sehen ist, pro Instanz eine jeweils andere Seite sehen.

Eine eigene Startseite können sie in der Datei `...shore\data\html\menu.html` definieren. Damit Sie pro Instanz unterschiedliche Seiten anzeigen können, müssen Sie noch etwas mehr tun.

Schreiben Sie sich pro Instanz eine eigene Indexseite, analog zu `...shore\data\index.html`. In jeder Indexseite verweisen sie auf eine eigene Menüseite analog zu `...shore\data\html\menu.html`.

Beispiel:

Ihre Instanzen heissen "Analyse" und "Rea2". Dann kann eine mögliche Struktur so aussehen:

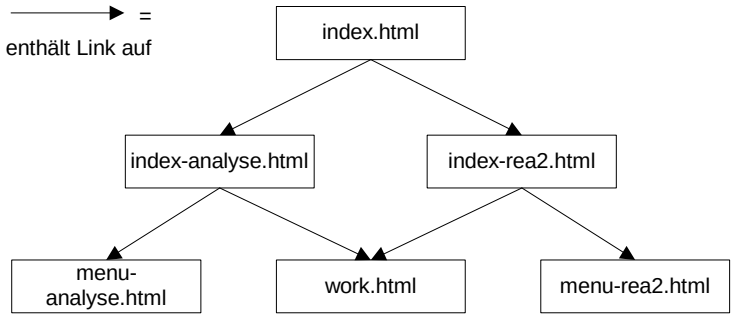


Abbildung 13: Hierarchie von HTML-Einstiegsseiten Seiten bei mehreren Instanzen

Nun können Sie sich in `menu-analyse.html` und `menu-rea2.html` ihre Startseite definieren, indem sie die Zeile

```
<!-- <param name="welcomepage" value="html/document.html"> -->
```

aktivieren (d.h. Kommentarzeichen löschen) und entsprechend ändern. Beispielsweise in:

```
<param name="welcomepage" value="html/rea2-startseite.html">
```

Nun wird nach dem Laden von SHORE zukünftig Ihre eigene Startseite erscheinen.

Menüs strukturieren

Sie können in SHORE die Menü-Einträge unter „Dokumente“ und „Objekte“ manuell ändern.
Überlegen Sie sich zunächst, wie die Menüs strukturiert sein sollen. Dabei können Sie auch im Metamodell definierte Dokumenttypen weglassen.
Beispiel:

Folgende Dokumenttypen sind in einem Metamodell definiert:

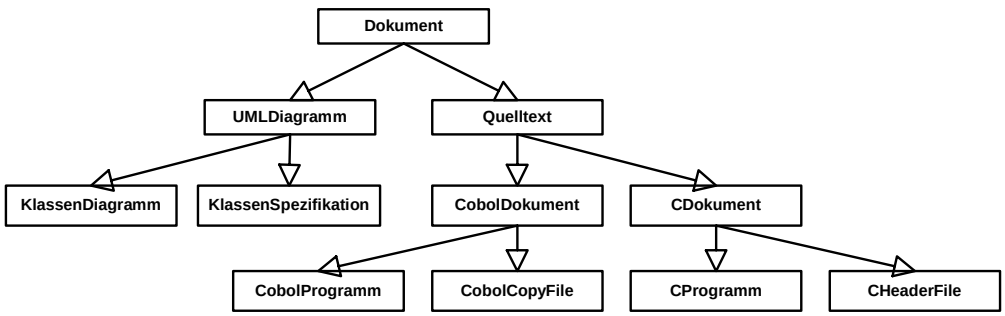


Abbildung 14: Metamodell-Beispiel: Hierarchie von Dokumenttypen

In Metamodellnotation formuliert sieht dieses Modell wie folgt aus:

```
Dokumenttyp Dokument
```

```
Dokumenttyp UMLDiagramm
ist ein Dokumenttyp
Dokumenttyp KlassenDiagramm
ist ein UMLDiagramm
Dokumenttyp KlassenSpezifikation
ist ein UMLDiagramm
Dokumenttyp Quelltext
ist ein Dokument
Dokumenttyp CobolDokument
ist ein Quelltext
Dokumenttyp CobolProgramm
ist ein CobolDokument
Dokumenttyp CobolCopyFile
ist ein CobolDokument
Dokumenttyp CDokument
ist ein Quelltext
Dokumenttyp CProgramm
ist ein CDokument
Dokumenttyp CHeaderFile
ist ein CDokument
```

Das Menü für unser Beispiel soll so aussehen:

```
   Dokumente
      Diagramm
         Klassendiagramm
         Klassenspezifikation
      Quelltext
         CobolProgramm
         CobolCopyFile
         CProgramm
         CHeaderFile
```

Wenn Sie sich die Struktur überlegt haben, können Sie sie in SHORE definieren.

So definieren Sie Ihre Menüstruktur

1. Öffnen Sie mit einem Texteditor im Verzeichnis `shore\data\html` die Datei `menu.html`
2. Finden Sie den folgenden Abschnitt:
```
<!-- <param name="docmenu" value="
   Titel1 {
      Dokumenttyp1;
      Dokumenttyp2;
   }
   Titel2 {
      Untertitel1 {
         Dokumenttyp3;
         Dokumenttyp4;
         Unteruntertitel1 {
            Dokumenttyp5;
         }
      }
   }
   Dokumenttyp5;
"> -->
```
3. Löschen Sie die Kommentarzeichen ("`<!--`" und "`-->`") und ändern Sie diesen Abschnitt entsprechend Ihren Wünschen. Beispiel:
```
<param name="docmenu" value="
   Diagramm {
      KlassenDiagramm;
      KlassenSpezifikation;
   }
   Quelltext {
         CobolProgramm;
         CobolCopyFile;
         CProgramm;
         CheaderFile;
   } ">
```
4. Speichern Sie die Datei und laden Sie SHORE im Browser neu (`<Shift-Reload>` oder `<Ctrl-Reload>`, je nach Browser).
   Die neue Menüstruktur sollte nun angezeigt werden.

PlugIns einbinden

Sie können PlugIns einginden, die Sie dann über das Menü im Browser starten können. Existierende PlugIns sind

- das Graphtoolzeigt graphisch die Struktur eines Objektes
- der VisClientzeigt die Metamodellstruktur grafisch und zeigt die besuchten Seiten als "anklickbaren" Graf

Zu Details fragen Sie bitte das Werkzeugteam.

Verteilung auf unterschiedliche physische Platten

Aus Geschwindigkeitsgründen speichern Sie möglichst die Logdateien der Objektstore-Datenbank auf einer eigenen (physische) Platte. Falls Sie das bei der Installation von SHORE nicht getan haben, so können Sie dies nachholen.

So ändern Sie das Verzeichnis, wo die ObjectStore-Logdateien gespeichert werden

1. Beenden Sie den SHORE-Serverprozess mit dem Kommando `shore exit`. Falls Sie mehrere SHORE-Instanzen laufen haben, so beenden Sie alle weiteren Instanzen auf die gleiche weise.
2. Melden Sie sich am Server als Administrator an
   Damit besitzen Sie die notwendigen Rechte, um Einstellungen an ObjectStore vornehmen zu können.
3. Starten Sie Start > Programme > ObjectStore Win32 > ObjectStore Setup
   Der ObjectStore-Setup wird gestartet, es erscheint das Dialogfenster „ObjectStore Setup“.
4. Falls noch der ObjectStore Server läuft, erscheint die Frage, ob der Server beendet werden soll ("... Do you want to shut down services and proceed with setup?").
   Bestätigen Sie diese Frage mit „Ja“ oder „Yes“.
5. Wählen Sie die Option „Set Up ObjectStore Server“ aus und klicken sie „Weiter“
6. Wählen Sie die Option „Yes - change ObjectStore Server“.
   Das Fenster „Customize Server Startup Parameters“ wird geöffnet.
7. Wählen Sie unter „Parameter Name“ den Parameter „Log File“ aus und klicken Sie auf „Edit Value“.
   Das Fenster „Log File“ öffnet sich.
8. Geben Sie hier den neuen Pfad und den Dateinamen für das Logfile ein.
   Beispiel: `E:\Logfiles\OSSVXNT.LOG`

Tipp: Merken Sie sich diesen Wert, sie müssen ihn etwas später noch einmal eingeben.

9. Klicken Sie „OK“ und danach im Fenster „Customize Server Startup Parameters“ nochmals „OK“.

Hinweis: Sollte das Verzeichnis nicht existieren, so wird es automatisch angelegt.

10. Antworten Sie auf die Frage "Would you like to create RAWFS partitions?" mit „No - ...“ und klicken Sie auf „Next >“
11. Auf die Frage "Reinitialize the ObjectStore log file" mit „Yes - ...“ und klicken Sie auf „Next >“.

Hinweis: Da das Logfile neu erstellt wird, muss es auf jeden Fall initialisiert werden.

12. Geben Sie im Fenster „Server Initialization“ noch einmal den Namen des Logfiles ein und klicken auf „OK“.
    `Beispiel: E:\Logfiles\OSSVXNT.LOG`

Tipp: Dies ist der gleiche Wert, wie oben.

Hinweis: Falls das Logfile vorher schon existierte, wird noch einmal nachgefragt, ob es wirklich initialisiert werden soll.

Das Logfile wird nun ggf. initialisiert. Anschließend erscheint eine Bestätigung, dass die Initialisierung fertig ist.

13. Wählen Sie „Yes - start the ObjectStore Server automatically“ und klicken Sie auf „Next >“.
14. Wenn Sie den ObjectStore-Server sofort starten wollen, dann antworten Sie auf die Frage "Do you wish to start these services right now?" mit „Ja“ oder „Yes“.
15. Klicken Sie auf „Finish“.

Das Setup ist nun abgeschlossen.

4.2 Betrieb eines SHORE-Servers

Wenn SHORE auf Ihrem Server installiert ist, gibt es einige regelmäßige Tätigkeiten, die Sie als Administrator während des Betriebs von SHORE durchführen.

4.2.1 SHORE starten und stoppen

SHORE starten

Wechseln sie in das Verzeichnis `shore\bin`.
Starten Sie den SHORE-Server durch Eingabe des Befehls `shore.exe`.

Tipp: Wenn Sie nur eine Instanz betreiben und der Datenbankname dem Standard entspricht, dann können Sie auch im Verzeichnis `shore\bin` des Explorers, auf die Datei `shore.exe` doppelklicken.

SHORE stoppen

Tipp: Wenn Sie SHORE als Dienst gestartet haben, dann dürfen Sie nicht einfach den Dienst stoppen, da dadurch SHORE nicht ordnungsgemäß heruntergefahren wird. Dies liegt daran, dass NT nur eine begrenzte Zeit zur Verfügung stellt, in der ein Dienst beendet werden muss. Braucht der Dienst länger, so wird er hart abgebrochen. SHORE benötigt in der Regel länger, als die von NT vorgegebene Zeit. Daher muss vor dem Beenden des Dienstes SHORE per Kommandozeile beendet werden.
Anschließend können Sie den Dienst stoppen.

Stellen Sie sicher, dass Sie im Kommandozeilen-Client den richtigen Port eingestellt haben und sich nicht im `shore\bin`-Verzeichnis befinden (Aufruf von `shore` ohne Parameter).
Geben Sie das Kommando `shore exit` im Kommandozeilen-Client ein.
Damit wird dem SHORE-Server der Befehl gesendet sich zu beenden. Wenn der SHORE-Server gestoppt ist, erscheint die Meldung:

```
<html>
<h1> und tsch³ssss.... </h1>
</html>
```

4.2.2 Systemdiagnose

SHORE ist als sehr robustes System ausgelegt. Um den Zustand Ihres Systems beurteilen zu können, gibt es einige Logdateien bzw. Prozesse, die Ihnen Hinweise über mögliche Systemstörungen liefern können.

Logdateien

Wenn Sie `shore.exe` gestartet haben, wacht es darüber, dass immer ein Prozess `shore-server.exe` aktiv ist.

Server-Log

Jeder von shore.exe gestartete shore-server.exe schreibt seine Ausgaben in eine eigene Logdatei, die im Verzeichnis ...shore\log gespeichert wird. Der Name der Logdatei ist nach folgendem Muster aufgebaut: server-<PID>.log wobei PID für die Prozess-ID des jeweiligen shore-server.exe steht. Die PID können Sie im Taskmanager sehen.

Falls ein shore-server.exe unerwartet abstürzt, sehen Sie im entsprechenden log-File am Ende eine entsprechende Fehlermeldung.

Hinweis: Die Logdateien können überschrieben werden, wenn zufällig ein shore-server.exe mit der selben Prozess-ID gestartet wird. Dies geschieht aber sehr selten und meist nur nachdem NT neu gestartet wurde oder nach sehr langer Laufzeit.

Tipp: Sortieren Sie die Logdateien nach Datum, um schnell die neuesten Dateien zu finden. Löschen Sie nicht mehr benötigte Logdateien, um die Übersicht zu behalten.

Parser-Log

Die Parser schreiben Ihre Ausgaben ebenfalls in Logdateien. Zu jeder Eingabedatei des Parsers existiert im Dokumentenverzeichnis eine gleichnamige Logdatei mit der Endung .log. Es kann sein, dass vom selben Projektdokument mehrere Versionen existieren, die sich nur durch eine angehängte Nummer unterscheiden. In dem Fall ist die Version mit der höchsten Nummer das neueste Dokument.

Sollte der Parser selbst fehlerhaft arbeiten, so kann es sein, dass im Verzeichnis ...shore\log eine Fehlermeldung in der Datei parser.log steht. Hier stehen Meldungen, die die Parser auf stdout oder stderr schreiben und Fehlermeldungen, die die Skriptumgebung meldet.

Datenbank-Log

Die ObjectStore-Datenbank schreibt Meldungen in zwei Dateien im Verzeichnis <Installationslaufwerk>:\Programme\ostore\log (der Pfad des Installationsverzeichnisses steht als Wert in der Umgebungsvariablen OS_TMPDIR). Der Prozess osserver.exe schreibt Meldungen in die Datei osserver.txt und der Prozess oscmgr4.exe schreibt Meldungen in die Datei oscmgr4.txt (oscmgr steht für den cache manager).

laufende Prozesse überwachen

Die auf dem SHORE-Server laufenden Prozesse (Siehe “Von SHORE gestartete Prozesse” auf Seite 54.) sollten von Ihnen als SHORE-Administrator regelmäßig überwacht werden.

hängende Prolog-Anfragen beenden

Falls Sie oder ein Surfer eine Prolog-Anfrage formuliert haben, die nicht beendet wird, können Sie ohne Gefahr den Task shore-xsb.exe im Taskmanager beenden.

4.2.3 Administration der Benutzer

Sie können für einen SHORE-Server den Zugriff per Browser auf bestimmte Benutzer einschränken. Dann lädt SHORE im Browser eines Surfers nur nach Eingabe einer gültigen UserID und passendem Passwort. Diese Beschränkung können Sie nur für einen Server festlegen, nicht für einzelne Instanzen. UserID und Passwort werden vom SHORE-Administrator festgelegt und können vom Benutzer nicht geändert werden.

So schränken Sie den Zugriff auf SHORE auf bestimmte Benutzer ein

1. Öffnen Sie in einem Texteditor im Verzeichnis `...shore\conf` die Datei `account.conf`
2. Fügen Sie am Ende der Datei Benutzer hinzu:

Syntax `<Name>:<Passwort>:<Kommentar>`
Beispiel `aladin:sesamopen:Benutzer Aladin aus dem Morgenland`

3. Die Benutzer können sich nach dem nächsten Start von `shore-server.exe` bzw. `shore.exe` anmelden.

Sobald Sie in der Datei `account.conf` Benutzer definiert haben und SHORE neu gestartet haben, erscheint beim Laden von SHORE im Browser ein Anmeldefenster. Das Anmeldefenster kann je nach Browser unterschiedlich aussehen.

Abbildung 15: Dialog zum Anmelden in SHORE

Eine Sitzung wird ungültig, wenn das Browserprogramm beendet wird.

So reaktivieren Sie einen gesperrten Benutzer

Wenn für einen Benutzer das Kennwort dreimal falsch eingeben wurde, wird der Benutzer gesperrt. Im Browser des Benutzers erscheint folgende Meldung:

Abbildung 16: Meldung im Browser nach dreimal falsch eingegebenem Kennwort

Erst nach einem Neustart des SHORE-Servers wird der Benutzer wieder freigeschaltet.

1. Stoppen Sie SHORE (siehe Kapitel “SHORE stoppen” auf Seite 71).
2. Starten Sie SHORE (siehe “SHORE starten” auf Seite 71).

Nun kann sich der Benutzer mit dem in `account.conf` eingetragenen Kennwort wieder anmelden.

4.2.4 weitere Systemeinstellungen

Objectstore-Datenbank

Die Objectstore-Datenbank enthält die Informationen über die Objekte und Beziehungen im SHORE-Metamodell. Neben der instanzspezifischen Datenbank (im Verzeichnis `...shore\store`) gibt es allgemeine Verwaltungsinformationen zur Datenbank in drei Dateien. Der Ort dieser Dateien wird bei der Installation festgelegt (siehe auch Kapitel “Verteilung auf unterschiedliche physische Platten” auf Seite 68):

Datei	Erläuterung
`oscmgr4.txt`	Logdatei des Objectstore Cache Managers. Zu beachten sind hier Meldungen über einen Abbruch des Cache Managers - wenn dieser nicht automatisch neu anläuft, dann bedeutet ein Abbruch des Cache Managers, dass SHORE nicht zur Verfügung steht. Der SHORE-Server scheint in dieserm Fall aber zu hängen, er liefert keine Fehlermeldung.
`osserver.txt`	Logdatei der Objectstore Datenbank. Hier kann es beim Start von Objectstore vorkommen, dass das Transaktionslog nicht leer ist, es wird dann nachgefahren. Das ist prinzipiell ok.

Datei	Erläuterung
`ossvxnt.log`	Das Datenbank-Log der Objectstore Datenbank. Hier werden alle offenen Transaktionen mitgeschrieben.

Wenn nach dem Systemstart die SHORE-Instanzen nicht zur Verfügung stehen, die entsprechenden Prozesse (`shore-server.exe`) aber im Taskmanager zu sehen sind, dann kann das u.a. an der Objectstore-Datenbank liegen.

Möglicherweise ist das Nachfahren der Transaktionen (Transaction log processing) noch nicht abgeschlossen, das geht aus einer entsprechenden Meldung in der Datei `osserver.txt` hervor.

In diesem Fall muss das System nochmal gestartet werden, dabei sollten die SHORE-Server nicht automatisch hochgefahren werden. Nachdem die Verarbeitung der Transaktionen abgeschlossen ist (Transaction log processing complete), können die SHORE-Server manuell gestartet werden.

verity-Suchmaschine

Bei der Suchmaschine sind nachträglich keine Änderungen zu machen.

Die Suchmaschine wird von SHORE aus nicht wie üblich genutzt. D.h. es läuft kein "spider", der sich zu indizierende Dokumente zusammensucht, sondern es werden die zu indizierenden Dokumente über ein Programm namens `mkvdk.exe` einzeln verarbeitet.

Der Aufruf von Suchanfragen erfolgt über die CGI-Schnittstelle von verity per Aufruf des Programms `s97cgi.exe`

verity-Support

Bei Schwierigkeiten mit der Suchmaschine wenden Sie sich bitte zuerst an das SHORE-Team. Falls das nicht möglich sein sollte, so müssen Sie beim verity-Support anmerken, dass es sich bei Ihrem Problem um eine verity-Installation in SHORE von sd&m handelt (wahrscheinlich kommt dann als Antwort, dass der Fehler aufgrund der SHORE-Installation geschieht, das ist aber nicht zwingend richtig ;-).

4.2.5 Projektdokumente

So löschen Sie die Daten einer Instanz:

Vor einem vollständigen Import kann es aus Geschwindigkeitsgründen vorteilhaft sein, vorher alle Daten einer Instanz zu löschen.

Tipp: Um nicht immer wieder alle Pfade und Parsereinstellungen neu eintragen zu müssen, sichern Sie sich einen leeren Stand der Datenbank, d.h. in der noch keine Dokumente importiert sind, aber schon die Pfadangaben und Parserzuordnungen gemacht wurden.
Dazu sichern Sie aus dem Verzeichnis "`...\shore\store`" die entsprechende Datei mit der Endung "db". Vor dem nächsten Start des SHORE-Servers kopieren Sie die Datei in das Verzeichnis zurück.
Bei einem Versionswechsel von SHORE kann sich allerdings das Datenbank-Schema ändern, dann können Sie die gesicherte Datenbank nicht weiter verwenden.

1. Stoppen Sie den SHORE-Server der Instanz mit den Kommando `shore exit` des Kommandozeilen-Clients.
2. Löschen Sie aus dem Verzeichnis `...\shore\store` die entsprechende Datei mit der Endung `db`.
   Damit ist die Datenbank gelöscht.
3. Löschen Sie aus dem Projektverzeichnis (Standard: `...\shore\projects`) die Dateien aus folgenden Verzeichnissen: `doc`, `html`, `xml` und `index`

   Hinweis: Unterhalb des Verzeichnisses `index` müssen Sie auch die entsprechenden Unterverzeichnisse löschen.
4. Starten Sie den SHORE-Server neu.

Projektdokumente nach SHORE importieren

Sie möchten Dokumente aus einem Projekt in SHORE einspielen.

So importieren Sie Dokumente mit dem Kommandozeilen-Client

1. Öffnen Sie ein DOS-Eingabefenster und wechseln Sie in das Verzeichnis, in dem die zu importierenden Dokumente liegen.

Hinweis: Dieses Verzeichnis muss in oder unter dem Arbeitsverzeichnis liegen.

2. Importieren Sie die Dateien mit dem Kommando `shore` und seinen Parametern.
   Den erfolgreichen Import zeigt der Kommandozeilen-Client für jede Datei mit einem ok an.

Syntax <Pfad des Arbeitsverzeichnisses>shore import <Datei>
Beispiel D:\Programme\shore\xml>shore import .\java*.java
Bestätigen Sie mit „`Enter`“

Hinweis: Der Dateiname kann Wildcards und den relativen Pfad zum Working Folder enthalten.

Mit dem Beispiel importieren Sie alle Dateien aus dem Verzeichnis `java`, die auf `.java` enden.

Als Ergebnis eines erfolgreichen Imports befinden sich die importierten Dateien auf dem Server im Projektdokumentverzeichnis (Standard: `...\shore\projekt\doc`), die vom Parser erzeugten XML-Dateien in `...\shore\projekt\xml` und sämtliche Dokument-, Objekt- und Beziehungsinformationen sind in die Datenbank eingetragen. Wenn eine Suchmaschine installiert ist, existieren außerdem HTML-Dateien im Verzeichnis `...\shore\projekt\html` und der Suchmaschinenindex ist aktualisiert.
Wenn beim Import kein ok zurückgegeben wird, deutet dies auf einen Fehler hin. Falls der Fehler vom Parser gemeldet wird, steht in der Regel ein Logfile des Parsers im XML-Verzeichnis.Falls der Fehler von der Skriptumgebung gemeldet wird, steht dies in der Datei `...\shore\log\parser.log`

Der Import kann während und nach seiner Verarbeitung Fehler melden. Bei einigen Fehlern erfolgt trotzdem ein import, bei einigen bricht der import ab. Hier nun einige Meldungen mit einem Hinweis, wie man den Fehler vermeiden kann:

deadlock

Normalerweise kann es keinen deadlock in SHORE geben abgesehen von einer Ausnahme, die sehr, sehr selten vorkommt. Wenn während eines Imports im Browser "Mengengerüst anzeigen" ausgeführt wird und diese Funktion für diese Instanz das allererste mal aufgerufen wird, kann es einen deadlock geben. Die Meldung sieht im Eingabeaufforderungsfenster so oder so ähnlich aus:`Fehler in SHORE ('F:\Programme\shore/bin/shore-update')`

```
  Klassifikation:  ObjectStore-Fehler <- Fehler im Paket store
  Fehlerparameter: <maint-0025-0692>Deadlock in upgrade_lock
(err_deadlock)
```

Abhilfe: einmal die Funktion "Mengengerüst anzeigen" ausführen und dann den Import erneut starten.

Automatisierung des Imports

Um den Import von Projektdokumenten zu automatisieren, haben Sie zwei Möglichkeiten (siehe auch “Anbindung von SHORE an ein KM-System” auf Seite 36 und “Triggerung über at-Kommandos oder cron-jobs” auf Seite 36):

- Zeitgesteuert (z.B. per `AT`-Befehl) und
- durch Aktionen gesteuert, wie z.B. durch einen check-in eines Konfigurationsmanagement-Systems.

Den Aufruf eines Imports können Sie über eine Batch-Datei, per Skript (wie z.B. Perl) oder auch aus Programmen (wie Java) heraus absetzen.

Bevor Sie das Rad neu erfinden, setzen Sie sich mit Projekten in Verbindung, die bereits SHORE einsetzen. Hinsichtlich der Automatisierung des Imports ist schon einiges entwickelt worden.

Projektdokumente aus SHORE exportieren

Auf die in SHORE enthaltenen Dokumente können Sie jederzeit zugreifen. Durch den Export können Sie sich Ihre Projektdokumente sowohl im Originalformat als auch in XML ausgeben lassen.

So exportieren Sie mit dem Kommandozeilen-Client ein Projektdokument im Originalformat aus SHORE

Originalformat bedeutet für SHORE, das Format, dass der Parser vorgefunden hat. Der Export liefert ein Projektdokument aus dem `doc`-Verzeichnis oder einem darunterliegenden Verzeichnis aus.

1. Öffnen Sie ein DOS-Eingabefenster.

Hinweis: Im Gegensatz zum `import`-Befehl können Sie sich in einem beliebigen Verzeichnis befinden.

2. Exportieren Sie eine Datei mit dem Kommando `shore` und seinen Parametern.
   Bei einem erfolgreichen Export wird die Datei im DOS-Fenster in der Standardausgabe (`stdout`) ausgegeben.

Syntax shore export <Pfad/Projektdatei>
Beispiel shore export uc/Taschenrechner.uc
Bestätigen Sie mit „`Enter`“

Hinweis: Der Dateiname muss mit dem relativen Pfad zum Working Folder zum Zeitpunkt des Imports eingegeben werden.

Mit dem Beispiel exportieren Sie die Datei `Taschenrechner.uc` aus dem Verzeichnis `uc`.

Tipp: Die Ausgabe können Sie wie folgt in eine Datei umleiten: `shore export > eineDatei.txt`

4.2.6 Anfragen und Auswertungen

Anfragen

Anfragen sind Prolog-Regeln, die bestimmte Fragestellungen beantworten und als Ergebnis eine Menge von Dokumenten oder Objekten zurückgeben. Wie Sie Anfragen in Prolog formulieren, finden Sie in einem eigenen Handbuch "Anfragen in Prolog".

Prolog-Regeln

Hinter jeder Anfrage steckt eine oder mehrere Prolog-Regeln. Diese Regeln können über den Browser formuliert und abgespeichert werden.

Die formulierten Regeln müssen einen instanzübergreifenden eindeutigen Namen besitzen und werden im Verzeichnis `shore\xsb\rules` als `<Name der Regel>.P` gespeichert.

Dabei sollten im Namen möglichst keine Sonderzeichen enthalten sein, auch Umlaute können bei der Formulierung von Standardanfragen Probleme verursachen.

So legen Sie eine Prolog-Regel an

1. Öffnen Sie im Browser unter dem Menüpunkt Administration > Prolog-Regel das Fenster „Prolog-Regeln bearbeiten“.

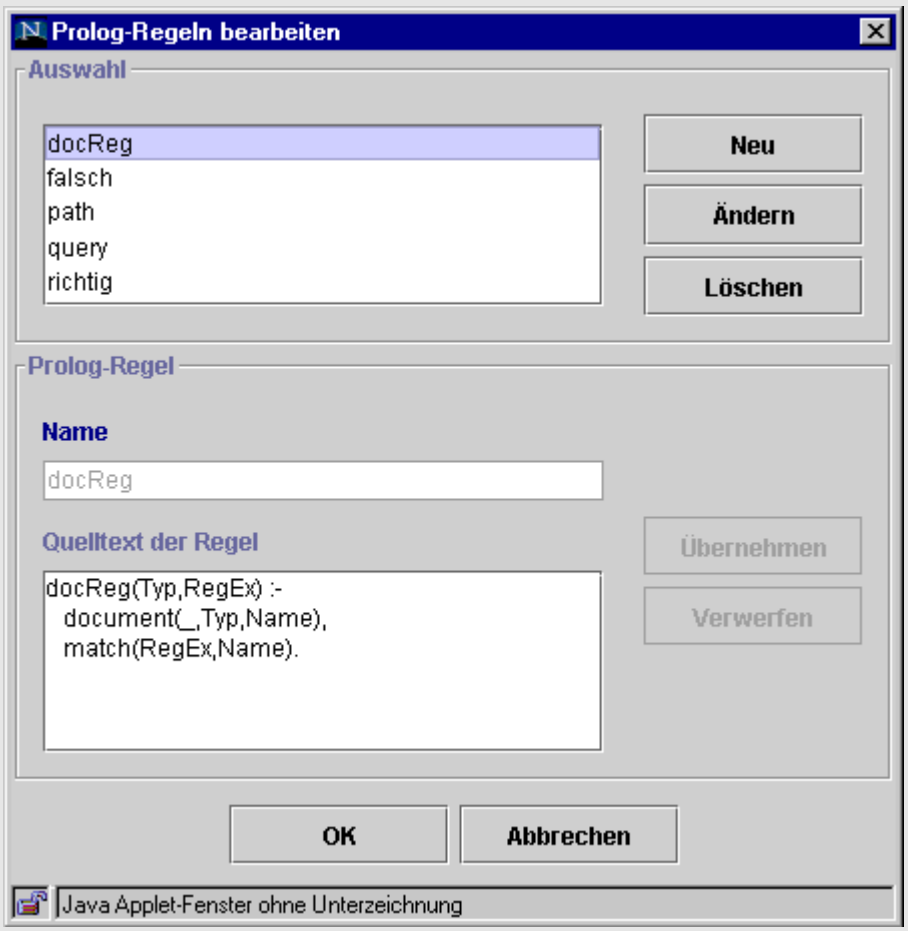


Abbildung 17: Dialog: Prolog-Regeln bearbeiten

2. Klicken Sie auf „Neu“.
   Der Bereich „Prolog-Regel“ wird zur Eingabe freigegeben.
3. Geben Sie im Feld „Name“ einen Namen ein, der noch nicht in der Liste unter „Auswahl“ existiert.
4. Geben Sie unter „Quelltext der Regel“ eine Prolog-Regel ein.
5. Klicken Sie auf „Übernehmen“.
   Die Regel wird in der Liste unter „Auswahl“ aufgenommen.
6. Schließen Sie das Fenster mit „OK“.

In diesem Fenster können Sie auch bestehende Prolog-Regeln bearbeiten und löschen.

Standard-Anfragen

Anfragen, die über eine Sitzung hinaus gespeichert werden sollen, können Sie als Standardanfrage abspeichern.

Standardanfragen erscheinen im Browser im Menü „Anfragen“.

Sie haben die Möglichkeit, Standardanfragen durch die Angabe von Kategorien zu gruppieren. Dies können Sie schon festlegen, wenn Sie eine neue Anfrage anlegen, aber auch später.

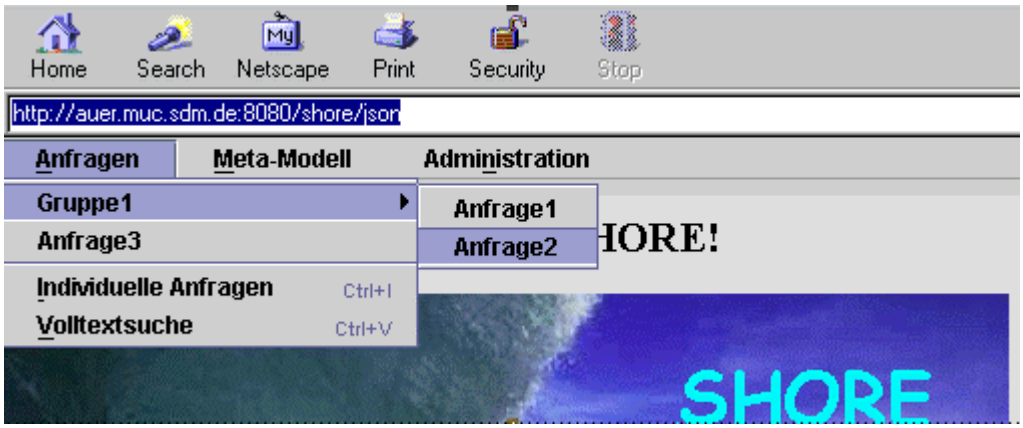


Abbildung 18: Menü ‘Anfragen’ mit bereits angelegten Standardanfragen

So legen Sie eine neue Standardanfrage an

1. Klicken Sie im Menü „Administration“ auf „Standardanfragen...“
   Das Fenster „Standardanfragen bearbeiten“ öffnet sich.

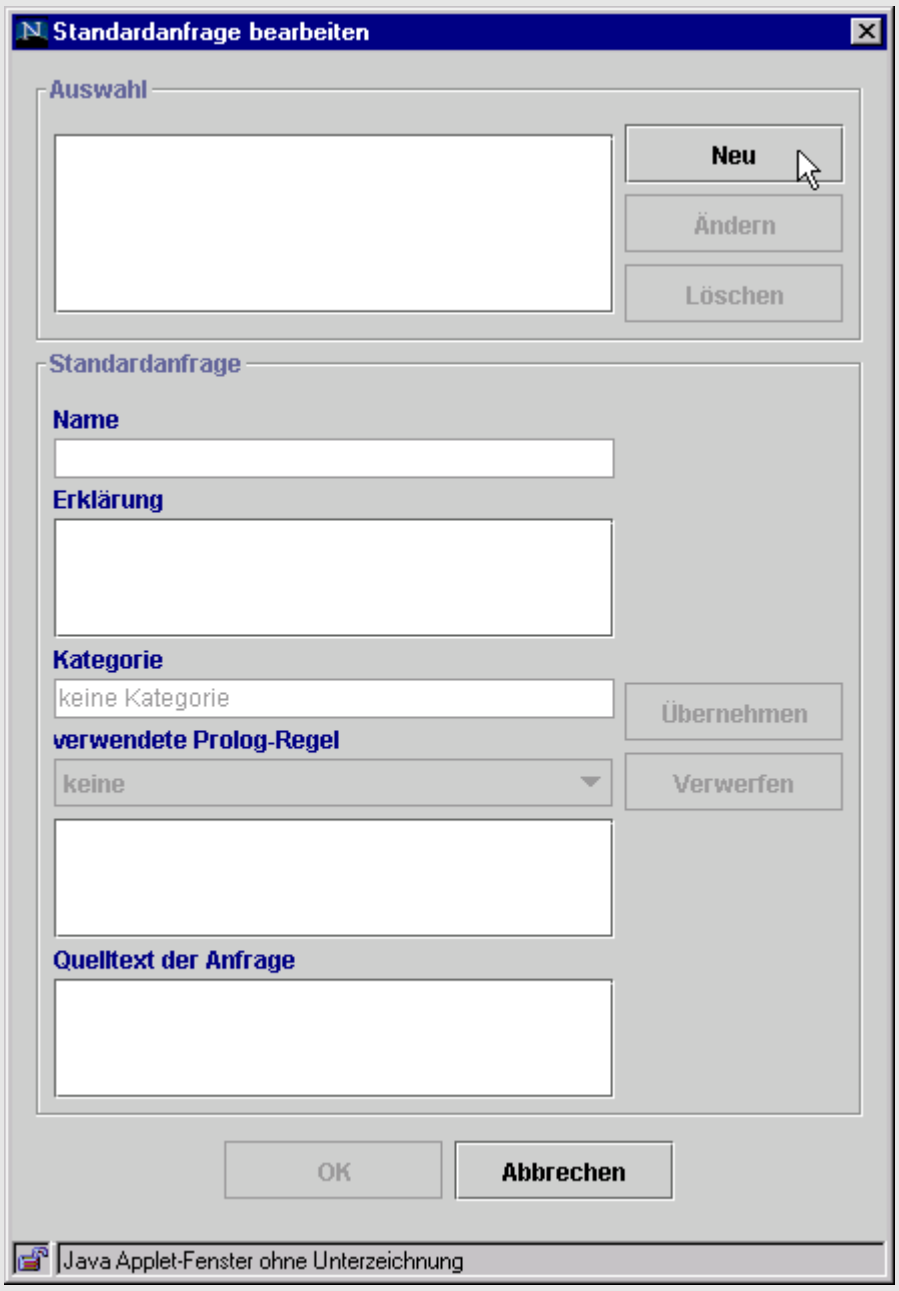


Abbildung 19: Dialog: Standardanfrage bearbeiten

2. Klicken Sie auf „Neu“.
   Der Fenster-Bereich „Standardanfrage“ öffnet sich für Eingaben.
3. Geben Sie im Feld „Name“ einen Namen ein.
   Beispiel:`Alle Dokumente`
4. Wechseln Sie ins Feld „Erklärung“ und geben Sie eine passende Beschreibung ein.
   Beispiel:`Zeigt alle Dokumente an.`

5. Wechseln Sie ins Feld „Quelltext der Anfrage“ und geben Sie eine Prolog-Anfrage ein.
   Beispiel: `document(X).`

Hinweis: Bei einer Anfrage können Sie angeben, dass beim Start der Anfrage Parameter abgefragt werden, die dann von der Anfrage verwendet werden. Dazu geben Sie einen Platzhalter an.

Syntax %(<Bezeichnung des Parameters>, <MetamodellTyp>,<Ausprägung des Typs>)
mit:
<Bezeichnung des Parameters>::=beliebiger String
<MetamodellTyp>::=Objekt | Dokument | String
<Ausprägung des Typs>::=<konkreter Typ wie in Metamodell definiert>
Beispiel %(Quelle, Objekt, JavaMethode)
Beispiel %(Wurzeldokument, Dokument, Quelltext)

6. Klicken Sie auf „Übernehmen“.
   Der Name der Anfrage erscheint im Bereich „Auswahl“
7. Schließen Sie das Fenster mit „OK“.

So ändern Sie eine bestehende Anfrage

1. Klicken Sie im Menü „Administration“ auf „Standardanfragen...“
   Das Fenster „Standardanfragen bearbeiten“ öffnet sich.
2. Markieren Sie in der Liste „Auswahl“ mit der Maus eine Anfrage.
3. Klicken Sie auf „Ändern“.
   Der Fenster-Bereich „Standardanfrage“ öffnet sich für Eingaben.
4. Bearbeiten Sie die Query und klicken Sie auf „Übernehmen“.
5. Bestätigen Sie die Frage „Standardanfrage überschreiben?“ mit „Ja“, wenn Sie die Änderungen speichern wollen.
6. Schließen Sie das Fenster mit „OK“.

Volltextsuche mit der verity-Suchmaschine

Eine Volltextsuche können Sie in SHORE ausführen, sobald auf dem Server (nach Anleitung) die verity-Suchmaschine installiert ist.

Nach der Installation stehen alle notwendigen Programme und Dateien unter der Verzeichnisstruktur `shore\verity`.

Damit eine Volltextsuche durchgeführt werden kann, muss ein Index erzeugt werden, in dem sämtliche im Wort vorkommenden Zeichenketten enthalten sind. In SHORE wird bei jedem Import standardmäßig der Suchmaschinenindex automatisch aktualisiert. Betreiben Sie mehrere Instanzen, so gibt es für jede Instanz einen eigenen Index (collection).

Die Indizierung erfolgt auf Basis von SHORE-intern angelegten HTML-Dokumenten. Diese werden aus den in SHORE enthaltenen XML-Dokumenten erzeugt und liegen mit in der Projektdokumentablage (standard: `shore\projekt\html`).

Wenn Sie die Suchmaschine nachträglich installieren oder aus irgendeinem Grund der Volltextindex nicht vollständig oder zerstört ist, so kann der Index per Kommandozeilen-Client mit dem Befehl `shore reindex` komplett neu aufgebaut werden.

Wie Sie eine Volltextsuche ausführen, ist im Anwenderhandbuch beschrieben.

Als Ergebnis einer Volltextsuche wird eine Liste der Dokumente angezeigt, die dem Suchbegriff entsprechen.

Daten als Liste ausgeben - dump

Sie können die in SHORE enthaltenen Daten als Liste aufbereitet ausgeben. Dafür nutzen Sie den Kommandozeilen-Client mit dem Aufruf `shore dump`.

Als Ergebnis erhalten sie auf der Standardausgabe (stdout) eine Liste nach folgendem Muster:

```
Überschrift[<Tab>;<Tab>Überschrift]*<CRLF>
Data[<Tab>;<Tab>Überschrift]*<CRLF> *
[<CRLF>
Überschrift[<Tab>;<Tab>Überschrift]*<CRLF>
Data[<Tab>;<Tab>Überschrift]*<CRLF> *
]*
```

Konkret könnte eine Ausgabe für einen Aufruf
`shore dump -modelType m`
so aussehen (stark gekürzt), wie auf der folgenden Seite.

```
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.

D:\>shore dump -modelType m
DocumentType;DocumentName
Dokument;Aggregation
Dokument;Auftragsschnittstellen
Dokument;Basisklassen
(...)
SequenzDiagramm;bucheUmsatz
SequenzDiagramm;pruefeDeckungSollumsatz
SequenzDiagramm;pruefeZulaessigkeitUmsatzartBeiDispositionskon-
toart

ObjectType;ObjectName;DocumentType;DocumentName;Offset
Attribut;RegelAuswertungDispositionsentscheid.bearbeitungsart;
KlassenSpezifikation;RegelAuswertungDispositionsentscheid;4
```

```
Attribut;RegelAuswertungDispositionsentscheid.dispositionsent-
scheid;KlassenSpezifikation;RegelAuswertungDispositionsentscheid;5
Attribut;RegelAuswertungDispositionsentscheid.lieferungstyp;Klas-
senSpezifikation;RegelAuswertungDispositionsentscheid;3
(...)
Variable;uebersetzung-kundennr.mandant-kunde.mdt-nr;KlassenSpezi-
fikation;uebersetzung-kundennr;16
Variable;uebersetzung-kundennr.mandant-kunde.returncode;Klassen-
Spezifikation;uebersetzung-kundennr;21
Variable;uebersetzung-kundennr.mandant-kunde.stamm-nr;KlassenSpe-
zifikation;uebersetzung-kundennr;13

RelType;DocType;DocName;Offset;SrcType;SrcName;TgtType;TgtName
Element_in_Namensraum;KlassenSpezifikation;regelwerk-admin;84;Ro-
seMethode;regelwerk-admin.op-update-daten-007;RoseKlasse;regel-
werk-admin
Element_in_Namensraum;KlassenSpezifikation;Umsatz;13;RoseMethode;
Umsatz.faelleDispositionsentscheid;RoseKlasse;Umsatz
Namensraum_ruft_Operation;SequenzDiagramm;DispositiveBearbeitung-
Umsatz;8;RoseKlasse;Umsatz;RoseMethode;Kon-
to.pruefeZulaessigkeitUmsatzartBeiEinschraenkungsgrund_S2
(...)
Operation_hat_Parameter;KlassenSpezifikation;regelwerk-admin;460;
RoseMethode;regelwerk-admin.op-insert-daten-013;RoseParameter;re-
gelwerk-admin.op-insert-daten-013.returncode
Operation_hat_Parameter;KlassenSpezifikation;regelwerk-admin;177;
RoseMethode;regelwerk-admin.zsys-struktur-typ-zuord;RoseParameter;
regelwerk-admin.zsys-struktur-typ-zuord.operation-kz
Operation_hat_Parameter;KlassenSpezifikation;regelwerk-admin;434;
RoseMethode;regelwerk-admin.op-insert-daten-035;RoseParameter;re-
gelwerk-admin.op-insert-daten-035.returncode


D:\>
```

Welche Möglichkeiten Sie mit `shore dump` haben, ist in der Befehlsreferenz des Kommandozeilen-Clients beschrieben (siehe Anhang).

4.3 Datenauswertung

Eine Stärke von SHORE ist es, schnell gewünschte Informationen über die enthaltenen Daten zu liefern. Sei es durch komplexe Auswertungen, die wertvolle Erkenntnisse über Systemzusammenhänge liefern oder durch das schnelle Auffinden von Dokumenten, die bestimmte Wörter oder Phrasen enthalten.

Komplexe Auswertungen können durch das integrierte Prolog-System in Form von Anfragen formuliert werden und Wörter und Phrasen können durch die Anbindung einer externen Volltextsuchmaschine gefunden werden.

Weiterhin kann der Inhalt von SHORE formatiert ausgegeben werden: als mit Trennzeichen formatierte Liste (csv) oder als formatiertes Worddokument.

SHORE bietet je nach Bedarf unterschiedliche Möglichkeiten.

Sie wollen...	Dann nutzen Sie:
...die Namen von Dokumenten oder Objekten herausfinden, die untereinander in bestimmten Beziehungen stehen.	Prolog-Anfragen
...die in SHORE enthaltenen Daten mit einem anderen Werkzeug weiterbearbeiten.	shore dump
...aus SHORE heraus ein Dokument erzeugen (z.B. Fachkonzept).	Reportgenerator
...die Namen von Dokumenten oder Objekten herausfinden, in denen bestimmte Wörter vorkommen.	Volltextsuche

Wie Sie diese Anfragen und Auswertungen nutzen, steht im Anwenderhandbuch. Wie Sie Prolog-Anfragen formulieren, steht in einem eigenen Handbuch. Für die Administration notwendige Informationen zu Anfragen und Auswertungen sind an anderer Stelle gesammelt. Siehe Kapitel “Anfragen und Auswertungen” auf Seite 79

4.4 SHORE auf dem Client

SHORE ist als Client-/Server-System ausgelegt. Während auf dem Server die von SHORE verwalteten Daten liegen, hat der Client zwei mögliche Aufgaben: die Darstellung der Daten im Browser und die Steuerung von SHORE über den Kommandozeilen-Client.

Die Installation des Kommandozeilen-Clients und eine einfache Einführung ist im Anwenderhandbuch beschrieben.

4.4.1 Nutzen mehrerer Instanzen von einem Client aus

unterschiedliche Ports

Sie können vom Browser auf unterschiedliche SHORE-Instanzen eines Servers zugreifen. Bei nur einer Instanz folgt der Zugriff über die URL `http://<serveradresse>:8080/shore/`. Wenn auf dem Server mehrere Instanzen laufen, so erfolgt der Zugriff über unterschiedliche Ports. Das heißt, dass jede Instanz seinen eigenen Port besitzt, die restliche URL aber gleichbleibt.

Im Browser können Sie dann einfach die jeweils eingerichtete Portnummer einsetzen.

Syntax http://<serveradresse>:<Port>/shore

Beispiel http://kailua.sdm.com:8081/shore

unterschiedliche Working-Directories

Für den Import existieren in der Regel pro Instanz auf einem Client unterschiedliche Working-Directories. Wenn dies der Fall ist und Sie einen Import starten wollen, dann prüfen Sie im Kommandozeilen-Client

- ob Sie den richtigen Port eingestellt haben und
- ob Sie das richtige Working-Directory eingestellt haben.

Diese Einstellungen können Sie sehen, wenn Sie den Kommandozeilen-Client ohne Befehl ausführen, also einfach mit dem Befehl `shore`. Dann wird die Hilfe zum Kommandozeilen-Client ausgegeben, an deren Ende die aktuellen Einstellungen stehen.

FAQ

Allgemeine Fragen zum Einsatz von SHORE

Was macht SHORE?

SHORE kombiniert die Eigenschaften eines Repositories mit denen eines Hypertextsystems.
Es kann alle Grundlagen und Arbeitsergebnisse eines Softwareentwicklungsprojekts (Pattern, Konzepte, Anforderungen, Schnittstellen, Entwürfe, Quelltexte, Testfälle, Testprotokolle, Fehlerbeschreibungen, Änderungsprotokolle usw.) aufnehmen und ermöglicht darüber eine komfortable Navigation inklusive komplexer Abfragemöglichkeiten.

Arbeitsergebnisse gelangen in Form von Dokumenten in das SHORE-System. Dabei können unterschiedliche Dokumenttypen existieren, zwischen denen Beziehungen definiert werden. Über die Beziehungen kann wie in einem Browser "gesurft" werden und es können komplexe Auswertungen über den Inhalt und die Beziehungen der Dokumente durchgeführt werden.

Tiefergehende Beschreibungen finden sich unter der Rubrik Dokumentation im sww.

übrigens: SHORE steht für sd & m Hypertext Object Repository

Wann lohnt es sich, SHORE einzusetzen, wann nicht?

Wenn mit strukturierten Dokumenten gearbeitet wird, lohnt es sich eigentlich immer, SHORE einzusetzen.

Der Aufwand zum Aufsetzen von SHORE erscheint oftmals als Hürde, obwohl dies eine Investition darstellt, damit im Laufe des Projektes massiv Aufwände gespart werden können. Wir haben einmal versucht, ein paar Punkte zu sammeln, die aus unserer Sicht eine erhebliche Zeitersparnis bringen können:

folgende Funktion...	führt zu Zeitersparnis bei...
schnelle ad hoc-Auswertungen	Analyse, Reengineering, Wartung
undefinierte Dokumente/ Objekte finden	Analyse, Design, Reengineering,
systemübergreifende Sicht	Analyse, Konzeption, Reengineering, Einarbeitung neuer Mitarbeiter
Report Generator	Erzeugung von Dokumenten für unterschiedliche Zielgruppen auf Knopfdruck

folgende Funktion...	führt zu Zeitersparnis bei...
Nutzung mit normalem Browser	keine Installation von Clientsoftware bei Versionswechsel, kein Einarbeitungsaufwand in unterschiedliche Oberflächen
ständig aktuelle Sicht durch Kopplung mit Konfigurationsmanagement-System	keine vergessene Aktualisierung, vermeiden von Missverständnissen aufgrund unterschiedlicher Versionsstände

Was unterscheidet SHORE von anderen ähnlichen Werkzeugen?

SHORE unterscheidet sich von bisher am Markt erhältlichen Analysewerkzeugen und Dokumentmanagementsystemen in folgenden Punkten:

- SHORE ist durch die Möglichkeit, das Metamodell frei definieren zu können, für alle Arten von strukturierten Dokumenten einsetzbar (nicht nur für eine Programmiersprache oder nur bestimmte Dialekte von Programmiersprachen)
- SHORE erlaubt Vererbungsbeziehungen zwischen Dokumenttypen, über die navigiert werden kann und über die Auswertungen gemacht werden können.
- es sind nicht definierte bzw. abstrakte Ressourcen (Dokumente, Objekte und Beziehungen) erlaubt. D.h. Beziehungen können erst einmal ins "Leere" zeigen. Wenn das entsprechende Zieldokument dann importiert wird, werden die Beziehungen automatisch aktualisiert (konkretisiert). Nicht definierte Ressoucen können auf Knopfdruck angezeigt werden.
- SHORE kann von jedem Projekt einfach und flexibel erweitert werden (Metamodelle und Parser).

Wie integriert sich SHORE in Entwicklungsumgebungen?

Sie können SHORE recht einfach zusammen mit Entwicklungsumgebungen nutzen, die Ihre Daten in Form von Dokumenten ablegen. Wenn dies nicht der Fall ist, so gibt es noch die Möglichkeit, über ein Konfigurationsmanagement bei definierten Aktionen (z.B. check-in) oder zu bestimmten Zeitpunkten einen Import nach SHORE zu starten.

Der Import von Dokumenten kann durch einen Aufruf per Kommandozeile erfolgen und ist dadurch sehr flexibel.

Zwischen SHORE und Rational Rose ist es außerdem möglich, Objektnetze zu transferieren. Trägermedium ist dabei XML. Die Modell- und Metamodell-Inhalte werden im XMI-Standard ausgetauscht. Dies funktioniert, ist allerdings noch langsam!

So funktioniert es:

Austausch des Modells: SHORE-PML => XMI-PML => XMI-UML
Austausch des Metamodells: SHORE-MM <=> XMI-UML + XSLT-Generato-

ren

PML: ein Standard-MM für Programmiersprachen

Wieviel kostet SHORE?

SHORE ist für sd&m-Projekte kostenlos.

Für eine darüber hinaus gehende Nutzung existiert ein gestuftes Preismodell, das Sie im sww finden können.

Welche Projekte setzen SHORE ein?

Es gibt einige Projekte, die SHORE schon seit längerer Zeit nutzen. Dazu gibt es eine eigene Übersicht im sww.

Warum ist SHORE noch kein Selbstläufer geworden?

Seinerzeit war HORA - die Vorgängerin von SHORE - ein revolutionäres Werkzeug, das innerhalb von sd&m recht häufig eingesetzt wurde.

In der Hoffnung, dass SHORE mindestens genauso häufig genutzt wird, wie HORA, hatte man darauf gesetzt, dass jedes SHORE nutzende Projekt sich an den Pflege- und Weiterentwicklungskosten beteiligen.

Wie die Erfahrung aber gezeigt hat, findet eine deartige Regelung wenig Akzeptanz bei den Bereichs- und Projektleitern. Denn sie sehen zunächst nur die (kurzfristige) Kostenseite und nicht die Chance, das Projekt in kürzerer Zeit und/oder besserer Qualität auszuliefern. Wie so oft - bei vom Projekt selber zu finanzierenden Werkzeugen (sinnvolle Testwerkzeuge, vernünftige Dokumenationstswerkzeuge usw.) - unterbleibt dann der Einsatz ganz.

Gibt es Argumentationshilfen oder fertige Texte für Akquisitionen und Angebote?

Ja, wir sammeln Texte, die für das Schreiben für Angebote genutzt werden können (siehe in der Rubrik Dokumentation im sww).

Argumentationshilfe bietet weiterhin diese FAQ ;-)

Ist SHORE nicht ein Thema für sd&m Research?

Nein. sd&m Research fertigt keine Werkzeuge für den Projekteinsatz an. Zukünftig wäre ein Szenario denkbar, in dem sd&m Research aufgrund von Forschungsergebnissen und Projekterfordernissen zusammen mit sTm den Bau eines Werkzeuges anstößt. Die Realisierung wird dann vom Werkzeugteam durchgeführt.

Fragen zur Einführung von SHORE

Mit welchem Aufwand muss ich rechnen, wenn ich SHORE einsetzen möchte?

Zur besseren Abschätzung des Aufwandes lesen Sie das Administratorhandbuch durch oder lesen Sie die Antwort auf die nächste Frage.

Ich möchte SHORE einsetzen! Was muss ich tun?

Hier in Stichworten die Aktivitäten, die notwendig sind:

- Nutzungskonzept überlegen
- SHORE-Master benennen (eine Person aus dem Projekt, die sich um SHORE kümmert)
- Hardware besorgen
- SHORE-Version besorgen
- SHORE installieren
- Metamodell formulieren
- ggf. Parser erstellen
- Parser installieren

Zusätzlich gibt es noch einige Aktivitäten, die in einem Projekt häufig zusätzlich gemacht werden:

- Import automatisieren / an Konfigurationsmanagement koppeln
- Reporting anpassen

Das Nutzungskonzept sollte enthalten:

- die Ziele, die mit SHORE verfolgt werden,
- welche Dokumenttypen sollen in SHORE verwaltet werden
- die Einbindung in die Projektorganisation und Toollandschaft beschreiben,
- Vorgehen beim Import nach SHORE beschreiben
- SHORE-Betrieb
- etc.

Für eine reibungslose Einführung von SHORE hat es sich bewährt, eine Person zu benennen, die SHORE bei der Einführung und späteren Betrieb betreut (der SHORE-Administrator).

Die Installation von SHORE geht sehr schnell (ca. 30 Minuten).

Es muss für SHORE ein Metamodell definiert werden und es müssen passende Scanner oder Parser existieren. Im günstigsten Fall kann ein existierendes Metamodell mit den dazu passende Parsern aus einem anderen Projekt oder vom SHORE-Team genommen werden.
Im schlimmsten Fall muss man sich seine Parser selbst schreiben aber kann immer noch von einem vorhandenen Metamodell abkupfern.

Bekomme ich Unterstützung bei der Einführung von SHORE?

Das SHORE-Werkzeugteam kann Unterstützung bei der Einführung von SHORE leisten, dies muss jedoch möglichst langfristig abgestimmt werden.

Wir können bei folgenden Dingen unterstützen:

- Beratung beim Vorgehen
- Metamodell Design
- Parser Entwicklung

Der Betrieb von SHORE muss vom Projekt selbst erfolgen.

Welche Parser gibt es schon?

Über die folgenden Parser können Sie verfügen und sofort einsetzen:

- Java
- Cobol
- XCobol

Folgende Parser sind verfügbar, erfordern aber eine eigene Anpassung/Installation:

- C
- Framework für Office-Dokumente

Zum jetzigen Zeitpunkt in der Entwicklung befinden sich:

- allgemeines, programmiersprachenunabhängiges Scanner-/Parserframework
- allgemeiner Patternmatcher (ein erster Wurf ist bereits fertig)
- C++

Es kann auch ohne Parser gearbeitet werden. Dazu müssen die Dokumente in XML vorliegen und entsprechend dem Metamodell die passenden XML-Auszeichnungen enthalten (beispielsweise können so theoretisch Dokumente mit Framemaker erstellt werden, die ohne parsen nach SHORE importiert werden können).

Sind die Parser der Engpaß?

Jedes Projekt hat seine eigenen Vorstellungen und Wünsche. Zunehmend steigt der Aufwand bei nachfolgenden Fällen:

1. Es kann ein bestehender Parser oder Scanner genutzt werden.
2. Ein existierender Scanner wird angepasst
3. Ein existierender Parser wird angepasst.
4. Ein Scanner (Patternmatcher) wird neu erstellt.
5. Ein Parser wird neu erstellt

Zukünftig wird es aber vom SHORE-Team Frameworks für Patternmatcher und Parser geben, die den Aufwand erheblich vereinfachen.

Fragen zu SHORE in der Praxis

Wie koordiniert das Werkzeugteam die Entwicklung der Parser?

Mit Ihrer Hilfe. Wenn Sie einen Parser entwickeln wollen, wenden Sie sich an das Werkzeugteam. Vielleicht wissen wir von einem Projekt, die etwas ähnliches schon gemacht haben oder gerade realisieren.

Ich will Änderungen am Parser machen. Was muss ich tun?

Sprechen Sie mit uns (dem Werkzeugteam). Wir helfen Ihnen gerne mit Rat und Tat weiter.

Gibt es sowas wie eine SHORE-Community?

Ja, sowas ähnliches gibt es. Es gibt das sd&m-übergreifende "SHORE User Relation Forum" - kurz SURF - dass sich unregelmäßig zusammenfindet und wo ein Gedankenaustausch über SHORE stattfindet. Dafür gibt es den sd&m-Mailverteiler V_P_SURF@muc.sdm.de.

Für den internen Austausch gibt es den Mailverteiler V_P_USER

Wie beschleunige ich meinen Import?

Hier gibt es einige Dinge, die man ausprobieren kann:

- Trennung der Datenhaltung und des Logfiles auf unterschiedliche Platten.
- Importieren mit `-grouped n`
  Hier muss etwas experimentiert werden, wie groß das `n` sein soll. Dies hängt von der Hardware (CPU, Swapgröße, Hauptspeicher) und der Dokumentgröße ab. Es hat sich bewährt den Import über Skripten zu steuern und sich die jeweiligen Importzeiten zu merken.
- Importieren mit `-noindex` danach muss ein "`shore reindex`" gestartet werden, da sonst der Stand der Datenbank nicht mit dem Volltextindex zusammenpasst.
- Wenn bei einem inkrementellen Import sehr viele Dokumente importiert werden, lohnt es sich ggf. die Datenbank und Dokumentablagen per Hand zu löschen und den Index neu aufzubauen.

Welche Möglichkeiten gibt es, um Auswertungen auf meine Dokumente zu erzeugen?

Es gibt ein Framework (Reportgenerator) und die Möglichkeit, mit Prolog Auswertungen zu machen.

Welche Möglichkeiten bietet mir das integrierte Prolog und wie nutze ich es?

Prolog bietet umfangreiche Auswertungen auf die in SHORE gespeicherten Daten an. Dabei existierten automatisch eine Reihe von Standardabfragen.

Hier einige Beispiele:

`object(_).`	gibt alle in SHORE enthaltenen Objekte zurück
`document (_,'Quellcode',_).`	gibt alle Dokumente des Typs Quellcode zurück.

Diese Standardabfragen können zu eigenen Abfragen kombiniert und abgespeichert werden (unter dem Menüpunkt Administration).

Beispiel:

`document(_,Cobol-Source,Name), match('DBZGS.*',Name).`	gibt alle in SHORE enthaltenen Dokumente vom Dokumenttyp 'Cobol-Source' zurück, deren Name mit 'DBZGS' anfängt.

Fragen zu Konzepten in SHORE

Wo finde ich einen Überblick über die SHORE-Architektur?

Eine grobe Übersicht findet sich im Artikel des Objekt-Spektrums oder auf der SHORE-Homepage im sww (unter "Dokumente"). Detailliert beschrieben ist die Architektur im DV-Konzept ().

Was soll das mit Meta- und Meta-Metamodell?

Hier sei auf das Kapitel "2.2 Modelle, Metamodelle und das Meta-Metamodell" aus der Diplomarbeit von Harald Graich verwiesen, wo diese Frage anschaulich beantwortet wird. Die Diplomarbeit ist über das sww zugänglich.

Wie kommen meine Dokumente in SHORE hinein und wo speichert SHORE die Dokumente?

Es geschieht folgender Ablauf beim Import eines Dokuments in einer Transaktion. Die verschiedenen Verzeichnisse der Dokumentenablage können in der SHORE-Oberfläche unter Administration > Ändere Einstellungen... festgelegt werden.

1. Das Dokument wird vom "Kommandozeilen-Client" per HTTP an den SHORE-Server übertragen.
2. Dort wird das Dokument zunächst im Originalformat in der "Dokumentenablage" gespeichert.
3. Dann wird das Dokument vom Parser gelesen und in XML im entsprechenden Verzeichnis der Dokumentenablage gespeichert.
4. Wenn eine Suchmaschine installiert ist, wird das Dokument in HTML abgelegt und der Suchmaschinenindex aktualisiert.
5. Das XML wird vom SHORE-Server geparst und im Dokument gefundene Objekte und Beziehungen werden in der Datenbank gespeichert.
6. Wenn alles fehlerfrei lief, wird ein "ok" an den Client zurückgeliefert, ansonsten eine Fehlermeldung.

Warum wird eine OO-Datenbank eingesetzt?

Hierfür gibt es drei Gründe, die hier aus der im sww zugänglichen Diplomarbeit von Harald Graich zitiert sind:

- "Die Objekte und Beziehungen der Datenbasis bilden einen im Allgemeinen stark vernetzten Graphen. Ein Graph lässt sich zunächst in jedem der Datenbankmodelle einfach ausdrücken: als Menge von Knoten und Kanten in einer relationalen oder deduktiven Datenbank oder als ein durch Objektreferenzen verbundenes Netz von Objekten in einer objektorientierten Datenbank. Wichtige Zugriffe auf die Datenbank beim Verfolgen von Hyperlinks und bei der Berechnung von Navigationsanfragen (siehe 3.4.1) bestehen darin, Wege durch diesen Graphen zu verfolgen. Solche Anfragen werden in relationalen Datenbanken durch eine Folge von Joins berechnet. Ein Join zwischen zwei Relationen ist eine nach bestimmten Kriterien, wie zum Beispiel der Übereinstimmung gemeinsamer Attribute, selektierte Teilmenge des kartesischen Produkts der beiden Relationen und damit eine zumindest potentiell teure Operation. In der objektorientierten Darstellung des Graphens dagegen werden die Pfade durch ein direktes, schnelles Verfolgen von Objektreferenzen bestimmt. Diese Effizienzbetrachtung liefert das wesentliche Argument für die Verwendung einer objektorientierten Datenbank.
- Relationale Datenbanken sind aufgrund der historischen Entwicklung die technisch ausgereiftesten Systeme. Bereits seit Anfang der 80er Jahre existieren kommerzielle relationale Datenbanksysteme, sie sind heute die gebräuchlichste Form der Datenhaltung. Dagegen sind kommerzielle objekt-orientierte Datenbanksysteme erst seit Ende der 80er Jahre auf dem Markt (laut [Heuer]: GemStone 1987, ONTOS 1989, O2 1990, ObjectStore 1990). In dieser jetzt immerhin elf Jahre dauernden Präsenz am Markt konnte zwar nicht annähernd der Erfolg relationaler Datenbanken wiederholt werden, doch lässt der erreichte Verbreitungsgrad auf eine akzeptable Reife dieser Systeme schließen. Deduktive Datenbanksysteme fanden trotz einiger kommerzieller Ansätze und einer starken theoretischen Untermauerung keine weitere Verbreitung, die Datenbank-Fähigkeiten der verfügbaren Prototypen lassen einen Einsatz in einem industriellen Softwareprojekt nicht ratsam erscheinen.
- Sowohl der Entwurf als auch die Implementierung von SHORE geschehen in objektorientierter Form. Die Wahl eines geeigneten objektorientierten Datenbanksystems erlaubt eine direkte Übernahme der im Entwurf identifizierten persistenten Klassen als Schema für die Datenbank und eine nahtlose Integration von Datenbank und Implementierungssprache. Dies geht so weit, dass die Verwendung der Datenbank in weiten Teilen des Programms völlig unsichtbar ist, da persistente und transiente Objekte fast austauschbar verwendet werden können. Beim Einsatz eines relationalen Datenbanksystems dagegen wäre ein zweifacher Bruch zu überwinden gewesen: Zum einen hätte das objektorientierte Datenmodell auf "flache" Relationen in der Datenbank abgebildet werden müssen. Zum anderen wäre bei der Implementierung der übliche "impedance mismatch", also die Kluft zwischen der mengenorientierten, deklarativen Datenbanksprache SQL und der imperativen, eher satzorientierten Hostsprache C++ zu überbrücken gewesen."

Warum kein SQL sondern Prolog?

Die Anfragesprache von SHORE soll intuitiv und leicht erlernbar sein. Wie in der vorhergehenden Antwort ausgeführt, sollen komplexe Anfragen inklusive Rekursion und Pfadverfolgungen möglich sein. Außerdem muss auf einer objektorientierten Datenbank aufgesetzt werden.

Da SHORE ein Online-System ist, müssen Anfragen spontan zur Laufzeit formuliert und interpretiert werden können. Ein bestehender Anfrageauswerter und -optimierer wurde einer Eigenentwicklung vorgezogen.

Keine der zum Entscheidungszeitpunkt bekannten Standardkombinationen von Anfragesprache und Datenbank erfüllten diese Anforderungen, so dass für SHORE die Kopplung einer Objekt-Datenbank mit einem Logikprogrammiersystem als Anfragekomponente gewählt wurde.

Fragen zum SHORE-Projekt

Was macht das Werkzeugteam?

Das Werkzeugteam entwickelt Werkzeuge für sd&m-Projekte. Weitere Details sind auf der Seite des Werkzeugteams im sww zu finden.

Woran arbeitet das Werkzeugteam?

Das SHORE-Team ist ständig in Kontakt mit Projekten und Kunden, die SHORE einsetzt und bekommt daher viele Anforderungen direkt mit.

Zusätzlich haben wir immer neue Ideen und Konzepte, wie wir SHORE weiterbringen können. Was wir konkret machen, ist leider nicht aktuell im sww zu finden.

Welche Anforderungen an SHORE bestehen, ist im sww in unserem Problem Tracking-System webgnats im sww zu finden.

Wie kam es zu dem Projekt SHORE?

SHORE wurde aus vielerlei Gründen ins Leben gerufen:

- Probleme mit der Implementierung von HORA
- Akzeptanzprobleme von HORA in Projekten und bei Kunden
- hoher Bedarf an einem frei konfigurierbaren Werkzeug zur Verwaltung und Auswertung von Dokumenten in großen Projekten
- Bedarf an einem frei anpaßbaren Repository, das nicht gleich ein CASE-Tool ist/bedingt (z.B. Integration von vorhandenem Host-Repository, Reports eines vom Kunden festgelegtem CASE-Tool und eigenen Dokumenten)
- Bedarf an einem frei anpaßbaren Reengineering-Werkzeug
- kein Produkt am Markt erfüllt unsere Anforderungen

Was sind die Projektziele von SHORE?

Die Ziele des SHORE-Projektes sind:

- Implementierung eines neuen Hypertext-Repositories, das den Anforderungen von sd&m-Projekten entspricht
- Nutzung von Internet-Technologie (offene Architektur)
- Datenhaltung in einer Datenbank
- Eignung für große Projekte
- umfassende Abfragemöglichkeiten

Anhang

A Befehlsreferenz Kommandozeilen-Client

A.1 admin

Ändert Servereinstellungen für Aufrufe des Kommandozeilen-Clients.

Parameter

`-server <adresse>`	optional	Ändert die Serveradresse
`-port <port>`	optional	Ändert die Portnummer, an die der Kommandozeilenclient seine Befehle sendet.
`-wf <pfad>`	optional	Stellt das Arbeitsverzeichnis ein.

A.2 backup

Startet eine Sicherung der Daten auf dem Server.

Parameter

`<zielverzeichnis>`	muss	Das Verzeichnis, wohin die Daten geschrieben werden..

siehe auch `restore`

A.3 delete

Löscht Projektdokumente auf dem Server.

Parameter

`-noindex`	optional	Der Suchmaschinenindex wird nicht aktualisiert. Anschließend muss ein shore -reindex ausgeführt werden, um den Suchmaschinenindex zu aktualisieren.
`<projektdoks>`	muss	Die Namen der zu löschenden Dokumente.

siehe auch

`del_pdok`
`import`

A.4 del_pdok

Löscht alle Projektdokumente eines Typs auf dem Server.

Parameter

<dokumenttyp>	muss	Die Namen der zu löschenden Dokumente.

siehe auch

`delete`
`import`

A.5 del_ss

Löscht die Style-Sheet-Zuweisung für einen Dokumenttypen.

Parameter

<dokumenttyp>	muss	Der Name des Dokumenttyps.

siehe auch imp_ss

A.6 dump

Gibt Inhalte von SHORE formatiert auf stdout aus. Es können je nach Aufruf unterschiedliche Teilmengen der in SHORE enthaltenen Informationen ausgegeben werden.

siehe auch `query`

A.6.1 dump des Metamodells

Es werden alle Definitionen des Metamodells ausgegeben.

Parameter

`-format csv`	optional	Ausgabe als Komma-separierte Liste. default, da keine andere Ausgabe implementiert ist.
`-modelType mm`	muss	Steuert, dass das Metamodell ausgegeben werden soll.

A.6.2 dump des gesamten Modells

Es werden alle Objekte, Dokumente und Beziehungen ausgegeben.

Parameter

`-format csv`	optional	Ausgabe als Komma-separierte Liste. default, da keine andere Ausgabe implementiert ist.
`-modelType m`	muss	Steuert, dass das Modell ausgegeben werden soll.

A.6.3 dump von Objekten

Es werden nur Objekte ausgegeben.

Parameter

`-format csv`	optional	Ausgabe als Komma-separierte Liste. default, da keine andere Ausgabe implementiert ist.
`-modelType m`	muss	Steuert, dass Daten des Modells ausgegeben werden sollen.
`-section objs`	muss	Es sollen nur Objekte ausgegeben werden.
`-subsection <ObjectType>`	optional	Es sollen nur Objekte des angegebenen Typs ausgegeben werden

A.6.4 dump von Dokumenten

Es werden nur Dokumente ausgegeben.

Parameter

`-format csv`	optional	Ausgabe als Komma-separierte Liste. default, da keine andere Ausgabe implementiert ist.
`-modelType m`	muss	Steuert, dass Daten des Modells ausgegeben werden sollen.
`-section docs`	muss	Es sollen nur Dokumente ausgegeben werden.
`-subsection <DocType>`	optional	Es sollen nur Dokumente des angegebenen Typs ausgegeben werden

A.6.5 dump eines bestimmten Dokumentes

Es wird genau ein Dokument ausgegeben.

Parameter

`-format csv`	optional	Ausgabe als Komma-separierte Liste. default, da keine andere Ausgabe implementiert ist.
`-modelType m`	muss	Steuert, dass Daten des Modells ausgegeben werden sollen.
`-section docs`	muss	Einschränkung auf Dokumente.
`-subsection <DocType>"<DocName>"`	muss	Der Typ des Dokumentes und der Dokumentname.

A.6.6 dump von Beziehungen

Es werden nur Beziehungen ausgegeben.

Parameter

`-format csv`	optional	Ausgabe als Komma-separierte Liste. default, da keine andere Ausgabe implementiert ist.
`-modelType m`	muss	Steuert, dass Daten des Modells ausgegeben werden sollen.
`-section rels`	muss	Es sollen nur Beziehungen ausgegeben werden.

`-subsection <RelType>`
 	
`<srcType> <tgtType>`
	optional
	
	optional
		Es sollen nur Beziehungen des angegebenen Typs ausgegeben werden.
		Es sollen nur Beziehungen ausgegeben werden, die den angegebenen Quell- bzw. Zieltyp enthalten.

A.7 exit

	Beendet den SHORE-Server.
Parameter	keine

A.8 export

Gibt ein Projektdokument im Originalformat auf stdout aus.

Parameter

`<projektdokument>`	muss	Der Name des Dokuments, dass ausgegeben werden soll.

siehe auch

`exp_xmldok`
`import`

A.9 exp_ss

Gibt den Inhalt des Style-Sheets für einen Dokumenttypen auf stdout aus.

Parameter

`<dokumenttyp>`	muss	Der Name des Dokumenttyps, für den das Style-Sheet ausgegeben werden soll.

siehe auch `imp_ss`

A.10 exp_xmldok

Gibt ein Projektdokument im XML-Format auf stdout aus.

Parameter

`<projektdokument>`	muss	Der Name des Dokuments, dass ausgegeben werden soll.

`export`

siehe auch

A.11 import

Importiert eine oder mehrere Dateien.

Dabei können die angegebenen Dokumente einzeln oder gruppiert importiert werden.

A.11.1 import - Einzelverarbeitung

Jede Datei wird einzeln importiert.

Parameter

`<projektdoks>`	muss	Der Name eines Dokuments, dass importiert werden soll. Es sind hier Wildcards zulässig. Dann werden alle passenden Dokumente importiert.
`-parser <name>`	optional	Der Name des Parsers, der für den Import genommen werden soll.

A.11.2 import - Gruppenverarbeitung

Mehrere Dateien werden innerhalb einer Transaktion importiert.

```
import          | [-binary|-text] <projektdoks>
                | [-parser <parser>]
                | [-grouped [n] ] [-noindex]
```

Parameter

`-binary`	optional	Die nachfolgenden <projektdoks> sind keine Textdateien und sollen fuer die Uebertragung (base64-) codiert werden. Der Parameter -binary gilt fuer alle nachfolgende <projektdoks>. Wird nur beachtet, wenn auch -grouped angegeben wird.
`-text`	optional	Die nachfolgenden <projektdoks> werden uncodiert uebertragen. Wird nur beachtet, wenn auch -grouped angegeben wird.
`<projektdoks>`	muss	Der Name eines Dokuments, dass importiert werden soll. Es sind hier Wildcards zulässig. Dann werden alle passenden Dokumente importiert.
`-parser <name>`	optional	Der Name des Parsers, der für den Import genommen werden soll.

`-grouped`
 	
`<n>`
	muss
	
	optional
		Steuert das Transaktionsverhalten.
		Wenn dieser Parameter nicht angegeben wird, wird jede Datei in einer eigenen Transaktion importiert.
		-grouped fuehrt den Import aller Dokumente in einer Transaktion durch.
		Wenn nach -grouped eine Nummer > 0 eingegeben wird, dann wird die entsprechende Anzahl von Dokumenten in einer Transaktion importiert.

`-noindex`	optional	Der Volltextindex soll nicht aktualisiert werden. Um den Index zu aktualisieren muss dann noch ein shore -reindex ausgeführt werden.

A.11.3 import - Beispiele

`shore import foo.cbl`
foo.cbl wird als Textdatei uebertragen.

`shore import -binary bar.xls -text foo.pl -grouped`
bar.xls wird codiert uebertragen,

foo.pl als Textdatei innerhalb einer Transaktion.

`shore import foo.pl -binary *.dbf bar.pl`
foo.pl wird als Textdatei uebertragen, die restlichen

Dateien werden codiert.

A.12 imp_mm

Importiert eine Metamodelldatei.

Dazu darf kein Dokument in SHORE existieren.

Parameter

`<metamodelldatei>`	muss	Der Name der Metamodelldatei, die importiert werden soll.
`<zielverzeichnis>`	optional	Ein Verzeichnis, in dem die Metamodelldatei auf dem Server gespeichert werden soll.

siehe auch `imp_mm_rescan`

A.13 imp_mm_rescan

Importiert eine Metamodelldatei und sämtliche bereits enthaltene Dokumente werden neu importiert.

Parameter

`<metamodelldatei>`	muss	Der Name der Metamodelldatei, die importiert werden soll.
`<zielverzeichnis>`	optional	Ein Verzeichnis, in dem die Metamodell-datei auf dem Server gespeichert werden soll.

siehe auch `imp_mm`

A.14 imp_parser

Installiert einen Parser in SHORE.

Parameter

<parserdatei>	muss	Die ausführbare Parserdatei, die importiert werden soll.
<name>	muss	Der Name, unter dem der Parser beim Import angesprochen werden soll.
<endungen>	muss	Dateiendungen, die dem Parser zugeordnet werden sollen.

A.15 imp_ss

Importiert ein Style-Sheet für einen Dokumenttypen in SHORE.

Parameter

`<style-sheet>`	muss	Der Name der Style-Sheet-Datei, die importiert werden soll.
`<dokumenttyp>`	muss	Der Dokumenttyp, für den das Style-Sheet importiert werden soll.

siehe auch

`del_ss`
`exp_ss`

A.16 move

Benennt ein Projektdokument um.

Parameter

<name_alt>	muss	Der Name des Projektdokuments, dass umbenannt werden soll. Es sind keine Wildcards erlaubt.
<name_neu>	muss	Der neue Name des Projektdokuments. Es sind keine Wildcards erlaubt.

siehe auch `import`

A.17 query

Gibt das Ergebnis einer Query auf stdout aus.

Parameter

`<queryfile>`	muss	Der Name einer Text-Datei, in der eine Query formuliert ist.
`<prologregel>`	optio-nalmuss	Name der verwendeten Prolog-Regel

siehe auch

`dump`
`stdquery`

A.18 reindex

	Erzeugt den Volltextindex komplett neu.
Parameter	keine
siehe auch	`import` (mit Parameter `-noindex`)

A.19 restore

Stellt eine mit backup erzeugte Sicherung wieder her.

Parameter

`<quellverzeichnis>`	muss	Der Verzeichnisname, aus dem die Daten wieder hergestellt werden sollen.

siehe auch

`backup`

A.20 stdquery

Gibt das Ergebnis einer Standard-Query auf stdout aus.

Parameter

`<stdquery>`	muss	Der Name der Standard-Query, die ausgeführt werden soll.
`<param-name>=<wert>`	optional, mehrfach	Wenn die Standard-Query Parameter verlangt, dann werden diese hintereinander als Parameter mit angegeben.

siehe auch

`dump`

`query`

B Glossar

Arbeitsverzeichnis	Das Verzeichnis des Kommandozeilen-Clients, aus dem Projektdokumente importiert werden können.
Beziehung	Eine Beziehung im SHORE Sinne bildet ein Zusammenhang zwischen zwei durch SHORE Objekte oder Dokumente beschriebene Ausschnitte eines realen Systems ab. SHORE Beziehungen sind getypt, gerichtet und besitzen ein eindeutiges Quell- und Zielobjekt/-dokument.
CSS	Cascading Style Sheet(s) Vom W3C definierte Sprache zur Verknüpfung einer Darstellungsart mit den verschiedenen Strukturelementen eines Dokuments. Einzelne Layouts können einander teilweise überdecken (daher der Name).
Dokument	Abbildung von Wissen zu einem bestimmten Zeitpunkt, das von einem oder mehreren Autoren zusammengetragen wurde und nach Möglichkeit von den adressierten Lesern ohne zusätzliche Hilfen verstanden werden kann. Der Dokumentbegriff umfaßt daher - insbesondere bei SHORE - auch nichttextuelle Formen wie Grafiken und Diagramme mit einer vereinbarten Semantik.

DTD	Document Type Defintion (Dokumenttypdefiniton) Verweis in einem XML-Dokument auf seinen Dokumenttyp und damit auf eine Grammatik, die festlegt welche Auszeichnungen in Form von sog. Tags erlaubt sind, welche Parameter diese Tragen können und wie die Auszeichnungen geschachtelt werden können. Entspricht ein XML-Dokument dieser Grammatik, so handelt es sich um ein gültiges (valid) XML-Dokument. Das XML-Dokument kann entweder selber die Grammatik in einer "document type declaration" enthalten oder auf ein anderes Dokument (DTD Dokument) verweisen, die diese enthält.
Instanz	siehe SHORE-Instanz
Kommandozeilen-Client	Ein Java-Programm, mit dem vom Client aus Funktionen auf dem Server ausgeführt werden können, wie z.B. der Im- und Export von Dokumenten.
Markup	In Markupsprachen enthaltene Tags, die Abschnitte einschliessen. Beispiel: <kursiv>hervorgehobener Text</kursiv> Die Tags werden dann als Markup bezeichnet.
Markupsprachen	Markupsprachen fügen einem Text Elemente hinzu (sogenannte Tags), die für Formatierung oder Bedeutung sorgen. Markupsprachen sind beispielsweise HTML, XML, SGML.
Meta-Metamodell	Beschreibung der Typen und ihrer Beziehungen untereinander, die im Metamodell vorkommen.
Metamodell	Beschreibung der Typen und ihrer Beziehungen untereinander, die im Modell vorkommen. Konkret werden im Metamodell Objekttypen, Beziehungstypen, Dokumenttypen und ihre Vererbungsbeziehungen untereinander definiert.
Modell	Die Menge aller SHORE Objekte, Beziehungen und Dokumente bilden ein Modell eines realen Systems (z.B. eines Software-Systems).
Objekt	Bei einem Objekt im SHORE Sinne handelt es sich um einen Abschnitt in einem Dokument, der einen Ausschnitt eines Systems beschreibt bzw. definiert. Objekte sind von einem im Metamodell definierten Typ und tragen einen (mit Ausnahmen) typweit eindeutigen Namen.

Parser	Allgemein: Ein Software-Modul, das Dokumente oder Quelltexte syntaktisch analysiert und für die Weiterverarbeitung aufbereitet. Für SHORE wandelt ein Parser Eingabedateien in XML um. Dabei enthält die XML-Datei ein dem Metamodell entsprechendes Markup.
Projektdokumentablage	Dateiverzeichnishirachie auf der Server-Maschine, die alle Projektdokumente enthält.
Repository	(engl.: Lager, Archiv) System zur Aufbewahrung von Daten eines Projektes (typischerweise Quellen und Dokumente).
Ressource	Der Begriff Ressource im SHORE Sinne bildet einen Oberbegriff für Objekte und Dokumente. Jedes Objekt und jedes Dokument ist also eine Ressource. Ressourcen stellen Quellen und Ziele für Beziehungen dar.
SGML	Standard Generalized Markup Language (allgemeine Standardauszeichnungssprache) Umfangreicher ISO Standard zur Auszeichnung, Behandlung und Verarbeitung von Dokumenten.
SHORE-Instanz	Auf einem Server, auf dem SHORE installiert ist, können mehrere SHORE-Repositories eingerichtet werden. Jedes eingerichtete Repository wird synonym mit SHORE-Instanz bezeichnet. Mit diesen Instanzen kann parallel gearbeitet werden, indem für jede ein eigener SHORE-Server gestartet wird.
Tag	Ein in Markupsprachen verwendetes Element, das in spitzen Klammern eingeschlossen ist und eine bestimmte Bedeutung hat. Beispiel: Das Tag <html> leitet einen Text ein, der als HTML interpretiert werden soll. Mit dem Tag </html> endet dieser Text. Davor und danach kann allerdings noch Text stehen, der dann aber nicht zu dem eingeschlossenen Text hinzugezählt wird.
W3C	W3 Consortium Internationales Industriekonsortium zur Entwicklung einheitlicher Protokolle und Sicherung der Interoperabilität im World Wide Web.

XML	Extensible Markup Language (Erweiterbare Auszeichnungssprache) Vom W3C definierte textuelle Syntax zur Trennung von Struktur, Inhalt und Layout/Verhalten eines Dokuments. Strukturinformationen stehen bei XML-Dokumenten in sog. Auszeichnungen (Markup), die den zu beschreibenden Inhalt immer durch öffnende und schließende Tags klammern. Auszeichnungen können beliebige Attribute besitzen. Ein XML-Dokument heißt wohlgeformt, wenn der Syntax für die Tags eingehalten wurde und zu jedem öffnenden Tag ein schließendes existiert. Über eine DTD läßt sich die Struktur (Typ) eines XML-Dokuments beschreiben und prüfen. Das Layout bzw. Verhalten eines XML-Dokuments beim Drucken oder Anzeigen legt die Empfehlung nicht fest. Statt dessen sollen Stylesheets (entweder als CSS oder XSL) die Struktur mit einem für das jeweilige Ausgabegerät angemessenem Layout bzw. Verhalten verknüpfen. XML ist eine Untermenge vom SGML.
XML-Dokumentenablage	Dateiverzeichnishierarchie auf der Server-Maschine, die ein Abbild der Projektdokumentenablage ist aber die durch die Parser generierten XML-Dokumente enthält (und evt. entstandene Entitäten-Datei).
XSL	XML Style Langauage Vom W3C definierte Sprache zum Layout und Umformung von XML-Dokumenten zum Zwecke der Ausgabe oder sonstiger Darstellung. Insbesondere umfaßt XSL eine Teilsprache zum Zusammenfassen und Umformen von Dokumente, um z.B. Inhaltsverzeichnisse, Zusammenfassungen und verschiedene Sichten generieren zu können.

C Literatur

[1] Port numbers / Well known port numbers:
http://www.isi.edu/in-notes/iana/assignments/port-numbers

D Weitere SHORE Dokumente

Release-Notes
Anwenderhandbuch

Index