Endbericht

Konzept eines
FachInformationsSystems (FIS)



Institut für Stadt- und Regionalforschung
TU Wien / 12-2000

Inhalt

ORGANISATORISCHE RAHMENBEDINGUNGEN 4

Beauftragung 4
Durchführung 4

ALLGEMEINE PROJEKTZIELE 5

PROJEKTSCHWERPUNKTE 7

OPERATIONALISIERTE ZIELE 9
Schnelles Publizieren von FIS-Inhalten 9
Breiter Zugriff auf MA22-Daten 9
Flexibilität und Erweiterbarkeit 10
EINGESETZTE TECHNOLOGIEN 11
Extensible Markup Language (XML) 11
Grundprinzipien: 12
XML-Elemente 12
Hierarchische Gliederung durch Verschachtelung 12
XML-Attribute 13
Wohlgeformtheit und Gültigkeit von XML-Dokumenten 13
XML-Namespaces 14
XML-Anwendungsbereiche 15
Cascading Style Sheets (CSS) 16
CSS Defizite im Zusammenhang mit XML 17
Extensible Style Sheet Language (XSL) 18
Active Server Pages (ASP) 19

KONZEPTUELLES DESIGN VON FIS-KOMPONENTEN 20

FIS-KOMPONENTEN 20
Konzeptive Sicht 20
Aufgaben innerhalb der Komponenten 21
Layoutteil 21
Komponentenspezifikationsdatei 21
Darstellungsteil 21
Aufgabenverteilung zwischen Server und Client 21
Administrative Sicht 22
DETAILENTWURF DER FIS-KOMPONENTEN 23
EINLEITUNG 23
Überblick 23
Implementierungsplattformen 24
Datenzugriff 24
Layoutteil 24
Darstellungsteil 24
Ausblick 26
Generelle Codierungsrichtlinien für XML-Spezifiationsdateien 27
TREEVIEW 28
Zweck und Aufbau 28
Rahmenbedingungen und Motivation 28
Operationalisierung 28
Layoutteil der FIS-TreeView-Komponente 30
Entwurfsziel und Funktionalität 30
Implementierung 30
XML-Tagbeschreibung einer FIS-TreeView-Spezifikationsdatei 31
XML-Document-Element (TREEVIEW) 31
Spezifikation des TREEVIEW-Inhaltes über PATTERN-Listen 32
Beispiel 34

Darstellungsteil der FIS-TreeView-Komponente ... *35*
Entwurfsziel ... 35
Funktionalität ... 35
Implementierung ... 35
Integration der FIS-TreeView-Komponente in eine HTML-Seite ... *36*
Möglichkeiten zur Weiterentwicklung ... *37*
TABLEVIEW ... 39
Zweck und Aufbau ... *39*
Layoutteil der FIS-TableView-Komponente ... *40*
Entwurfsziel und Funktionalität ... 40
Implementierung ... 40
XML-Tagbeschreibung einer FIS-TableView-Spezifikationsdatei ... *41*
XML-Document-Element (TABLE, TABLES) ... 41
TABLE-Aufbau (FIELDS- und ROWS-Tags) ... 42
Felddefinition (FIELD-Tag) ... 43
Die eigentlichen Daten (ROW-Elemente) ... 45
Beispiel ... 45
Darstellungsteil der FIS-TableView-Komponente ... *46*
Entwurfsziel ... 46
Funktionalität ... 46
Implementierung ... 48
Integration der FIS-TableView-Komponente in eine HTML-Seite ... *49*
Möglichkeiten zur Weiterentwicklung ... *50*
MAPVIEW ... 51
Zweck und Aufbau ... *51*
Anbindung externer Tabellen ... 52
Layoutteil der FIS-MapView-Komponente (FIS-MapDesigner) ... *54*
Entwurfsziele ... 54
Funktionalität ... 54
Implementierung ... 55
XML-Tagbeschreibung einer FIS-MapView-Spezifikationsdatei ... *56*
Grundsätzliche Struktur ... 56
MAP-Element ... 57
GRAPHIC-Element ... 59
COLOR - Element ... 61
SYMBOL-Element ... 62
TEXTSYMBOL-Element ... 65
LAYER-Element ... 67
RENDERER-Element ... 71
Darstellungsteil der FIS-MapView-Komponente ... *75*
Entwurfsziele ... 75
Funktionalität ... 75
Implementierung ... 77
Integration der FIS-MapView-Komponente in eine HTML-Seite ... *78*
Möglichkeiten zur Weiterentwicklung ... *79*

Organisatorische Rahmenbedingungen

Beauftragung

Auftraggeber:

Magistrat der Stadt Wien, MA 22 – Umweltschutz

Auftragnehmer:

Institut für Stadt- und Regionalforschung der TU Wien

Bearbeitungszeitraum:

Dez. 1999 bis Dez. 2000

Auftragsgegenstand:

"Konzept eines FachInformationsSystems (FIS)"

Durchführung

Die Durchführung des Projektes "Konzept eines Fachinformationssystems“ erfolgte am Institut für Stadt- und Regionalforschung der Technischen Universität Wien im Rahmen der Teilrechtsfähigkeit gemäß §3 Abs. 1 UOG 1993.

Institutsvorstand:

o.Univ.Prof. Dr.-Ing. Dieter BÖKEMANN

Karlsplatz 13
A – 1040 Wien
tel. 58801 / 26601
fax 58801 / 26699

Durchführende Sachbearbeiter:

Dipl.-Ing. Leopold RIEDL
Dipl.-Ing. Robert KALASEK

Während der gesamten Realisierungsphase bestand zwischen Auftragnehmer (Inst. f. Stadt- und Regionalforschung) und Auftraggeber (MA 22) intensive Kooperation. Im Zuge der Arbeitssitzungen/Workshops wurde dem Auftraggeber regelmäßig über den Fortgang der Arbeiten Bericht erstattet. Darüber hinaus ermöglichten die Arbeitssitzungen bzw. Workshops beiden Partnern die in der Beauftragung angesprochene kooperative inhaltliche und organisatorische Konkretisierung des Auftrages. In diesen Prozess waren seitens des Auftraggebers über die gesamte Bearbeitungsphase Mitarbeiter der Magistratsabteilung 14 intensiv eingebunden. Dabei erfolgten inhaltliche Schwerpunktsetzungen im Wesentlichen in Abstimmung mit den Projektverantwortlichen der MA22, technisch-instrumentelle in erster Linie mit Vertretern der MA14.

Allgemeine Projektziele

Ziel des Projektes „Konzept eines Fachinformationssystems (FIS)“ ist die Erstellung eines detaillierten Konzeptes sowie eines experimentellen Prototyps für das geplante inter-/intranetbasierten Fachinformationssystem (FIS) der MA22.

Das FIS soll als Web-basiertes Informations- und Auskunftsinstrument in erster Linie den Mitarbeitern/innen der MA22 – in der aus dem Umgang mit dem Internet vertrauten Arbeitsumgebung (Web-Browser Internet Explorer 5.0) – einfach und schnell Zugang zu fachspezifischen magistratsinternen Daten geben. In weiterer Folge ist grundsätzlich eine Ausweitung der Zugriffsmöglichkeit auf bestimmte Inhalte des FIS auf den gesamten Magistrat der Stadt Wien und letztlich auf eine noch breitere Öffentlichkeit vorgesehen.

Für die Anwender/-innen soll das FIS ein leistungsfähiges Arbeitsmittel im Umgang mit raumbezogenen Sachverhalten sein. In einem grundsätzlich offenen System – bestehend aus statischen Web-Inhalten einerseits und dynamisch erzeugten interaktiven kartographischen und tabellarischen Rauminformationen anderseits – soll dem Benutzer ein alltagstauglicher und themenorientierter Zugang zu Datenbeständen der MA22 zur Verfügung stehen. Dieser interaktive Umgang mit Karten und Tabellen soll die integrative Bewertung komplexer raumbezogener Fragestellungen zusätzlich unterstützen und damit zu einer verstärkten Nutzung des Datenbestandes beitragen.

Die Einbettung sowohl statischer als auch dynamisch erstellter, interaktiver Elemente in ein im Grunde themenorientiertes hierarchisches Navigationsgerüst hält aufgrund der damit verbundenen Orientierungsqualität die Suchaufwände gering. Gleichzeitig garantiert die Vernetzung der Inhalte über die aus dem Internet gewohnten Links quer zur thematisch-inhaltlichen Struktur einen bedarfsorientierten Zugang zu komplexen und vielschichtig verflochtenen Informationen.

Mit Rücksicht auf die geforderte Alltagstauglichkeit ist die Komplexität des Systems problemadäquat zu wählen, d.h. die beiden schwer zu vereinbarenden Vorgaben "große Leistungsfähigkeit und Flexibilität des Werkzeuges“ einerseits und "möglichst intuitiver Benutzerzugang“ andererseits sind gegeneinander abzuwägen. Vor diesem Hintergrund ist daher bewusst in Kauf zu nehmen, dass die Benutzer mit vordefinierten Sichten auf die vorhandenen Datenbestände arbeiten und deren Möglichkeiten Art, Gestaltung und Kontext der präsentierten Inhalte zu beeinflussen grundsätzlich begrenzt sind. Unter vordefinierten Sichten werden hier seitens des Datenanbieters entworfene HTML-Seiten mit Text-, Karten- und Tabelleninformationen verstanden, deren grundsätzliche Darstellungsform durch den Autor weitgehend vorgegeben ist.

Aus dem Blickwinkel des Datenanbieters besteht die Forderung nach möglichst geringem Aufwand für die Bereitstellung dieser Informationen. Das betrifft einerseits die Integration von bestehenden oder laufend anfallenden Informationen in das Informationssystem und anderseits deren Bereitstellung, also die Publikation der Informationen für einen bestimmten engeren oder weiteren Benutzerkreis. In beiden Aufgabenfeldern (Integration und Publikation) gilt es die notwendigen Abläufe derart zu gestalten, dass sie auch durch Nicht-EDV-Spezialisten erfolgen können.

Für die komplexen Aufgaben im Zusammenhang mit der Publikation von Informationen im FIS sind geeignete Layoutkomponenten notwendig. Auch an diese Entwurfs- und Definitionswerkzeuge ist die Forderung nach möglichst großer Praxistauglichkeit bei gleichzeitig bestmöglicher Eingliederung in bestehende EDV-Strukturen auf Seiten der MA22 und der MA14 zu stellen.

Die damit verfolgte Unabhängigkeit der Fachabteilung(-en) von zentralen EDV-Diensten auf der inhaltlichen Ebene des Auskunfts- und Informationssystems bildet die Grundlage eines raschen und nachfrageorientierten Auf- und Ausbaus des FIS und gewährleistet die angestrebte breite Nutzerakzeptanz eines lebendigen und möglichst aktuellen Informationssystems.

Voraussetzung für eine Systemimplementierung mit geringem Schulungsaufwand ist neben der Verwendung von Standard-Web-Browsern für die Navigation zu den Inhalten des FIS und deren Darstellung auch ein klares und einheitliches Konzept bei der Erstellung der Benutzerschnittstelle. Soweit möglich, soll der/die Benutzer/in in erster Linie einen inhaltlichen Zugang zum System haben können. Mit Implementierungsdetails und den zugrunde liegenden Technologien sollen Anwender möglichst nicht konfrontiert werden.

Projektschwerpunkte

Ausgehend von diesen bereits in der magistratsinternen „Voruntersuchung Fachinformationssystem (FIS)“ in ähnlicher Form angesprochenen allgemeinen Projektzielen wurden im gegenständlichen Forschungsauftrag "Konzept eines Fachinformationssystems (FIS)“ folgende Teilleistungen spezifiziert:

- Demo-Site „BiotopMonitoring-Wien“
- Detailkonzept des FIS der MA22
- Prototyp der FisServerEngine

Es handelt sich beim Projekt FIS laut Vertrag um einen Forschungsauftrag im Sinne eines anwendungsorientierten Konzeptes mit dem Ziel, anhand der im Prototyp und der Demo-Site realisierten Beispiele und Komponenten einerseits technische und inhaltliche Anforderungen an das FIS zu konkretisieren, anderseits das Potential bestehender und absehbarer Internet-Technologien auszuloten und darüber hinaus die Benutzerakzeptanz der implementierten Ansätze zu testen.

Die im Punkt „Detailkonzept“ des Anbotes zusammengefassten Leistungen betreffen zwei zu unterscheidende Ebenen:

- die konzeptive, den methodisch-technischen Rahmen betreffende und damit übergeordnete Ebene. Dazu zählen folgende Punkte des Vertrags:
  - (b) Festlegung des methodisch-technischen Rahmens (server- und clientseitige Konzepte, verwendete Softwarewerkzeuge, div. Konventionen
  - (c) Abgrenzung und modulares Design der Systemkomponenten, Einbettung in die bestehende Daten- und Programmstruktur der MA14
  - (e) Definition der zentralen Schnittstelle zwischen FisServerEngine, LayoutEngine und GEORD-Tools auf Basis der XML-Spezifikationen des W3C.
- die hauptsächlich auf der Voruntersuchung aufbauende und deren Ansätze vertiefende Ebene. Dazu gehören folgende Punkte des Vertrags:
  - (a) Detaillierte Festlegung des inhaltlichen und gestalterischen Konzepts unter Einbeziehung der magistratsinternen Voruntersuchung
  - (d) Detaillierte Definition der Funktionalität der FIS-Komponenten unter Berücksichtigung von Anforderungen der künftig zu implementierenden LayoutEngine
  - (f) Definition von session- und applikationsspezifischen Variablen und Konzepten

Die mit dem Projekt FIS verfolgten Ziele und die bereits in der Voruntersuchung angedachten Realisierungsansätze liegen im Trend aktueller Internetentwicklungen. Die Technologien im unmittelbaren und mittelbaren Umfeld des Web entwickeln sich rasch und mit ihnen die Standards (manchmal im Vorfeld, oft in der Folge neuer Konzepte). Der gesamte Sektor befindet sich in einer Umbruchsphase, die durch die explosionsartige Verbreitung des Internet in den vergangenen 5 Jahren ausgelöst wurde. Projekte, die sich mit Implementierungsmöglichkeiten bestimmter Ansätze auseinandersetzen, laufen in dieser dynamischen Phase generell Gefahr von den aktuellen Entwicklungen überholt zu werden.

So bestanden z.B. mit der Verbreitung des MS Internet Explorer in der Version 5.0 (IE 5) im Jahr 1999 einige wesentliche Voraussetzungen für die stärkere Betonung der umsetzungsorientierten Projektaspekte (gegenüber den rein konzeptuellen).

Dazu gehören:

- volle XML-Unterstützung
- Custom Behaviors (HTML-Objekte mit benutzerdefiniertem Verhalten)
- verbessertes clientseitiges Scripting
- verbesserte Cascading Style Sheets (CSS) und XLS

Die Verfügbarkeit dieser implementierungsfreundlichen SW-Konzepte im IE 5 gestattet die Implementierung einzelner im Vertrag lediglich auf konzeptiver Ebene zur Bearbeitung vorgesehener Elemente des Prototyps.

Die damit prototypisch implementierten Ansätze stießen in einer relativ frühen Phase der Auftragsbearbeitung auf großes Interesse seitens des Auftraggebers, woraus sich in der Folge einvernehmliche Modifikationen in Teilbereichen des Projektes ergaben.

So wurde z.B. der ursprünglich explizit nicht in Anbot und Vertrag enthaltene Layout-Teil der Map-Komponente aufgrund seiner zentralen Bedeutung für Aufbau und Test einer Demo-Site vollständig implementiert. Erst durch dieses Werkzeug konnten die Voraussetzungen geschaffen werden, die es dem Auftraggeber erlauben, weitgehend autonom die Möglichkeiten und Grenzen des Systems unter praxisnahen Bedingungen – d.h. integriert in den alltäglichen Arbeitsprozess – auszuloten.

Die daraus resultierende Entlastung von der Aufgabe „Demo-Site Biotopmonitoring“ wurde vom Auftraggeber in Absprache mit dem Auftrageber genutzt, um die prototypischen Implementierungen der server- und clientseitigen Systembestandteile zum Teil weiter als ursprünglich geplant voranzutreiben. In wesentlichen Punkten des Auftrages konnten daher Ergebnisse erzielt werden, welche über die konzeptive Ebene (d.h. über jene Bearbeitungstiefe, die sich mit dem Machbaren, den möglichen Alternativen und den Vor- und Nachteilen verschiedener Varianten auseinandersetzt) deutlich hinaus gehen.

Im Vorfeld der Implementierung erfolgte jeweils das für ein Konzept charakteristische Abwägen zwischen den möglichen Implementierungsansätzen, im darauf folgenden Umsetzungsschritt die Auswahl und Implementierung eines oder mehrerer Ansätze und im Anschluss daran die reflexive Bewertung.

Der wesentliche Vorteil gegenüber einer in erster Linie konzeptiven Arbeit besteht in der Möglichkeit, die gewählten Ansätze anhand konkreter Applikationen und konkreter Erfahrungen aus dem Umgang mit diesen zu diskutieren und in der Folge zu überarbeiten.

Die eingangs erwartete große Breite des Konzeptes wurde in dessen Verlauf in Richtung stärkerer Schwerpunkte modifiziert, woraus sich folgende **Konsequenzen** ergaben:

- größere Ausgereiftheit der implementierten Komponenten
- geringerer Tiefgang bei den extensiv bearbeiteten Themenfeldern

Die Auswahl der **Schwerpunktsetzungen** erfolgte aufgrund der

- in der Vorstudie angesprochenen Ziele und Prioritäten,
- Bedeutung des implementierten Ansatzes innerhalb des Systems (Schlüsselfunktion),
- Repräsentativität des Ansatzes für andere ähnliche Projektteile,
- im Projektverlauf artikulierten Prioritäten des Auftraggebers.

Operationalisierte Ziele

Schnelles Publizieren von FIS-Inhalten

Dadurch soll seitens des Datenanbieters ein lebendiges – weil nachfrageorientiert entwickeltes und erweiterbares – Werkzeug für den täglichen Gebrauch im Umgang mit raumbezogenen Informationen innerhalb der MA22 (und später darüber hinaus) entstehen.

Geringe Aufwände bei der Bereitstellung von Daten gehen mit rascher Reaktionsfähigkeit auf Benutzerwünsche einher. Ausgehend von einem Basisdatenbestand können oft benötigte Datensichten auch kurzfristig zur Verfügung gestellt, Veränderungen daran kurzerhand durchgeführt werden.

Das FIS-Design unterstützt schnelles Publizieren durch

- eine klare und generische Veröffentlichungs- und Zugriffsstruktur über den FIS-Verzeichnisbaum (Orientierungskomponente) und dem FIS-internen "Explorer", dem sog. TreeViewer.
- Nutzung gewohnter SW-Werkzeuge im Designprozess,
  z.B. ist die MapDesigner-Komponente für die Gestaltung von interaktiven Karten eine ArcView-Extension, mit deren Hilfe das Kartenlayout in der gewohnten Arbeitsumgebung erstellt und beim Publizieren automatisch ins Web übernommen wird
- einfache Integration von Komponenten in "normale" HTML-Seiten mit möglichst minimalem Deklarations- und Spezifikationsbedarf über vorgefertigte CSS-Dateien und darin eingebettete *Custom Behaviors* von HTML-Elementen.

Breiter Zugriff auf MA22-Daten

In der ersten Ausbaustufe soll das FIS innerhalb der MA22 als zentrales Auskunfts- und Informationssystem die Bearbeitung von in der Praxis alltäglich auftretenden Fragestellungen erleichtern.

Hohe Benutzerakzeptanz ist die Voraussetzung für die intensive Nutzung des Werkzeuges und damit die verstärkte Nutzung vorhandener Datenbestände. Wesentlichen Einfluss auf die die Akzeptanz haben u.a. von folgenden Faktoren:

- einfache Bedienbarkeit
  durch die gewohnte Oberfläche im Browser (IE 5) und die darin integrierten Werkzeuge
- geringer Suchaufwand
  d.h. klare Zugriffsstruktur über die TreeView-Komponente auf den inhaltlich strukturierten FIS-Verzeichnisbaum und die gewohnten HTML-Querverweise innerhalb der Texte, Karten und Tabellen
- analytisches Potential und Interaktivität
  - **Kartenausschnitte**
    Auswahl räumlicher Interessensschwerpunkte
  - **Inhalte**
    Auswahl thematischer Interessensschwerpunkte (Auswahl von Karten, An- und Abwahl von Themen im Rahmen vom Kartenautor getroffener Vorgaben) und Auswahl von Objekten (räumliche und thematische Filter).

Flexibilität und Erweiterbarkeit

Viele technische und Inhaltliche Anforderungen an ein komplexes, universell einsetzbares Werkzeug sind in dessen Konzeptphase noch nicht im Detail absehbar. EDV-Werkzeuge mit potentiell breitem Anwendungsspektrum sind daher nicht bereits zum Zeitpunkt ihrer Implementierung aus einem Guss, sondern enthalten – als Teil ihres Werkzeugsatzes – Konzepte zur eigenen Erweiterung und Anpassung an sich verändernde Aufgaben, Problemstellungen und Benutzeranforderungen.

Innerhalb des FIS gehören dazu:

- bedarfsorientierte Erweiterungsmöglichkeit der FIS-Inhalte,
  also der über das Werkzeug zugänglichen Datenbestände. Hier liegt keine Einschränkung von Seiten des Systems FIS vor.
- bedarfsorientierte Erweiterungsmöglichkeit der FIS-Werkzeuge
  also Ausbaufähigkeit der Funktionalität durch Integration neuer Komponenten für neue inhaltliche Fragestellungen über ein modulares Komponenten-Design.
- Integration mit vorhandenen DV-Werkzeugen
  durch einen modularen Aufbau und die Verwendung standardisierter und flexibler Schnittstellen. Neue Komponenten sind dadurch jederzeit nachträglich in das FIS integrierbar. Es müssen lediglich entsprechende Layout- und Darstellungsteile, sowie XML-Spezifikationen entwickelt werden (siehe Seite 20ff).

Eingesetzte Technologien

In diesem Kapitel werden grundlegende und im FIS verwendete Ansätze überblicksartig beschrieben. Dieser Abschnitt kann daher übersprungen werden, wenn die entsprechenden Inhalte bereits geläufig sind (i.e. XML, CSS, XSL und ASP).

Extensible Markup Language (XML)

XML ist eine offene textbasierte Sprache, die sich für die Beschreibung strukturierter semantischer Informationen eignet. Wie HTML ist auch XML eine Markierungs- oder Auszeichnungssprache. Während erstere eine Anwendung der Metasprache SGML ist, bildet letztere eine Teilmenge davon.

XML wurde von einer Arbeitsgruppe des World Wide Web Consortium (**W3C**) entwickelt, die bereits 1996 als Fortführung des SGML Editorial Review Board gegründet wurde. Die wichtigsten Ziele für den Entwurf von XML sind:

- **leichte Einsetzbarkeit im Netz**
  Reduktion der komplexeren Metasprache SGML auf die web-relevanten Bestandteile unter Beibehaltung der notwendigen Flexibilität
- **breites Anwendungsspektrum**
  verallgemeinerte Metasprache, die sich für verschiedenste inhaltliche Anwendungsbereiche über DataTypeDefinitions (DTD) an die spezifischen Bedürfnisse anpassen und aufgrund ihrer klaren Struktur generell gut automatisch verarbeiten lässt.
- **Kompatibilität zu SGML**
  gültige XML-Dokumente sollen auch gültige SGML-Dokumente sein, d.h. XML bleibt eine Teilmenge von SGML und jeder SGML-Prozessor soll grundsätzlich jedes XML-Dokument verarbeiten können.
- **einfach zu erzeugende und zu interpretierende XML-Dokumente**
  XML-Dokumente sind strukturierte Textdateien (ASCII-Text), deren Struktur einfach und durchgängig, und damit auch der menschlichen Interpretation zugänglich ist. Eine derartige Struktur kann selbst mit einfachsten Werkzeugen erzeugt und weiterverarbeitet werden.
- **Unabhängigkeit und Offenheit**
  die Erfahrungen u.a. mit JAVA schlagen sich im Bedürfnis nach einem verbindlichen hersteller- und plattformunabhängigen, offenen und erweiterbaren Standard nieder.

  (vlg. W. J. Pardi (1999): XML in Action, Microsoft Press; Unterschleissheim, 1999)

XML ist eine Metasprache zum Definieren von Dokumenttypen, es liefert also den Regelsatz, der bei der Definition von Dokumenttypen anzuwenden ist. Infolgedessen eignet sich XML für die Erzeugung abgeleiteter Sprachen, die als Sprachanwendungen von XML für bestimmte, meist spezifische Anwendungsbereiche mit ihren jeweils speziellen Bedürfnissen entwickelt werden.

Im Unterschied zu HTML sind in XML nicht die Sprachelemente vordefiniert, sondern lediglich die Regeln für deren Definition. Eine XML-Applikation (i.e. eine Sprachanwendung von XML) wird also entwickelt, indem in einer DataTypeDefinition (oder einem XML-Schema) zulässige Elemente des Dokuments und deren mögliche hierarchische Schachtelung (Inhaltsmodell) determiniert werden. Die DTD schreibt also die formale Grammatik fest, nach der ein XML-Dokument aufgebaut wird.

XML-Dokumente werden daher nicht wie HTML-Dokumente aus einem festen Stamm von Befehlen aufgebaut, sondern aus den in der DTD definierten Elementen und sie enthalten

ebenfalls im Unterschied zu HTML keinerlei Information mit der die Darstellung des Dokuments am Browser beschrieben wird.

Gemeinsam ist beiden Sprachen grundsätzlich die Art der Auszeichnung von Inhalten, also die Markierung von Elementen mittels „Tag“ sowie das Prinzip der Verschachtelung von Elementen.

Grundprinzipien:

- Definition von Elementen (wie bei HTML) über Start- und End-Tags
- hierarchische Gliederung durch Verschachtelung von Elemente
- Formale Grammatik in Form der DTD

XML-Elemente

XML-Elemente werden über semantische Tags aufgebaut, die Informationen über den Inhalt des Elements geben. Im allgemeinen setzt sich ein Element aus einem Start-Tag, einem End-Tag und dem Inhalt des Elements zusammen. Ein Beispiel:

Start-Tag	Inhalt	End-Tag
<AUTOR>	**Hans Mayr**	**</AUTOR>**

Das Start- und End-Tag <AUTOR>...</AUTOR> liefert die Information, dass es sich bei dem Text dazwischen um einen Autor handelt.

Hierarchische Gliederung durch Verschachtelung

Hierarchisch aufgebaute baumartige Strukturen werden in XML durch die Schachtelung von Elementen abgebildet, d.h. innerhalb eines Tag-Paares bestehend aus Start- und End-Tag werden andere vollständig spezifizierte Elemente eingefügt, die ihrerseits wiederum in der selben Form Sub-Elemente enthalten können.

```
<BUCH>
    <TITEL> Der Test</TITEL>
    <AUTOR> Hans Mayr</AUTOR>
</BUCH>
```

Mehrfachverschachtelung – Beispiel (siehe nächste Seite):

```
<BIBLIOTHEK>
   <ZEITSCHRIFTEN>
      <ZEITSCHRIFT>
         <TITEL> Die Zeitschrift 1</TITEL>
         <AUTOR> Peter Müller </AUTOR>
         <JAHRGANG>1999</JAHRGANG>
      </ZEITSCHRIFT>
      <ZEITSCHRIFT>
         <TITEL> Die Zeitschrift 2</TITEL>
         <AUTOR> Franz Bauer</AUTOR>
         <JAHRGANG>1999</JAHRGANG>
      </ZEITSCHRIFT>
   </ZEITSCHRIFTEN>
   <BÜCHER>
      <BUCH>
         <TITEL> Der Test</TITEL>
         <UNTERTITEL> Untertitel 1</ UNTERTITEL>
         <AUTOR> Hans Mayr</AUTOR>
      </BUCH>
      <BUCH>
         <TITEL> ... </TITEL>
         <UNTERTITEL> ...</UNTERTITEL>
         <AUTOR> ... </AUTOR>
      </BUCH>
   </BÜCHER>
</BIBLIOTHEK>
```

XML-Attribute

Attribute werden verwendet, um Elemente genauer zu spezifizieren. Dabei unterscheiden sich XML-Attribute nicht grundsätzlich von den auch in HTML verwendeten Attributen, z.B. HTML-Align-Attribut `<H4 ALIGN="center">`.

Attribute sind grundsätzlich einem Element zugeordnet. Sie sind daher Bestandteil des Element-Start-Tags und haben generell die Form

```
ATTRIBUTNAME = "wert"
```

Je Element sind mehrere Attribute zulässig, deren Werte zwischen Hochkommas stehen müssen (im Ggs. zu HTML). Ihre Reihenfolge innerhalb eines Elements hat i.a. keine Bedeutung.

```
<BUCH ATTRIBUT_1="wert_1", ATTRIBUT_2="wert_2", ... ATTRIBUT_n="wert_n"/>
```

Wohlgeformtheit und Gültigkeit von XML-Dokumenten

Die auf einer DTD beruhenden XML-Dokumente werden Instanzen der DTD genannt. Dabei werden **wohlgeformte** (well formed) und **gültige** (valide) Instanzen unterschieden:

- **wohlgeformte** Instanzen entsprechen den grundlegenden Regeln von XML bezüglich der Tag-Syntax, und der Verschachtelung von Elementen
- **gültige** Instanzen entsprechen exakt der zugrunde liegende DTD (die entweder im Dokument selbst enthalten ist oder auf die ein Verweis besteht)

In Bezug auf die Verarbeitung von XML-Dokumenten differenziert die vom W3C verabschiedete Spezifikation zwischen zwei Instanzen: dem Parser und der Anwendung.

Die Aufgaben des Parsers bei der Verarbeitung eines Dokuments sind:

- Prüfung auf Wohlgeformtheit und ggf. Ausgabe von Fehlermeldungen
- ggf. Zusammenfügen von Bestandteilen des Dokuments aus unterschiedlichen Files
- Erstellen eines Baums, durch den die Inhalte der Elemente für eine bestimmte Anwendung zugreifbar werden

Validierende Parser prüfen XML-Dokumente auf Gültigkeit und lesen dazu die gesamte DTD sowie alle externen im Dokument referenzierten, geparsten Entities ein. Nicht-validierende Parser müssen lediglich das Dokument selbst auf Wohlgeformtheit prüfen.

Die wichtigsten Regel für **Wohlgeformtheit** sind:

- das Dokument sollte mit einer XML-Deklaration beginnen;

  z.B. `<?XML version="1.0" ?>`,

- alle Elemente mit Inhalt müssen sowohl ein Start- als auch ein Ende-Tag haben.
- alle Attribut-Werte müssen in Anführungszeichen stehen,
- jedes leere Element (wie z.B. die HTML-Tags <IMG>, <HR>, und <BR>) muss entweder mit '/>' enden oder es muss ein regulären Ende-Tag hinzugefügt werden.

  Beispiel: <BR> würde entweder zu <BR/> oder zu
</BR>,

- Elemente müssen sauber ineinander eingebettet werden (d.h. keine überlappenden Auszeichnungen bei Verschachtelung),
- auf der obersten Hierarchieebene muss genau ein Element existieren, das als Container für alle anderen Elemente dient (i.e. das sog. *document element*).

  (Quelle: http://members.aol.com/xmldoku/f_und_a.htm)

XML-Namespaces

Das Konzept von XML erlaubt grundsätzlich gleichlautende Element- und Attributnamen in verschiedenen XML-Applikationen. In der Praxis steigt mit zunehmender Verbreitung von XML daher die Wahrscheinlichkeit des Auftretens zweier Elemente aus unterschiedlichen Applikationen mit der selben Bezeichnung. Die potentiellen Namenskonflikte sollen gemäß XML-Spezifikation durch die Einführung von Namensräumen (*namespaces*) vermieden werden.

Namespaces bezeichnen in XML einen Ansatz zur Erzeugung universell eindeutiger Namen in XML-Dokumenten durch Identifikation eines Elementnamens mit einer eindeutigen externen Ressource.

```
<?xml version="1.0"?>
<?xml:namespace ns="http://www.xx.yy/schema/ns" prefix="ns_1"?>
<?xml:namespace ns="http://www.aa.bb/schema/ns" prefix="ns_2"?>
<PRODUKT>
    <P_NAME>
        <ns_1:anzahl> 100 </ns_1:anzahl>
        <ns_2:stueckpreis> 20 </ns_2:stueckpreis>
        <DATUM> 10/10/00</DATUM>
    <P_NAME>
</PRODUKT>
```

Namensraumnamen und Prefix:

- http://www.xx.yy/schema/ns → Prefix: ns_1
- http://www.aa.bb/schema/ns → Prefix: ns_2

Lokaler Teil:

ist der Element- oder Attributname im XML-Dokument, im Beispiel „P_NAME“, „anzahl“, „stückpreis“, „DATUM“

Durch Kombination von Namensraumnamen und Namensraumpräfix entstehen universell eindeutige Elementnamen

- <ns_1:anzahl> ....</ns_1:anzahl>
- <ns_2:stueckpreis> ... </ns_2:stueckpreis>

XML-Anwendungsbereiche

- **Layoutproblem**

  Teil des Konzeptes von XML ist die konsequente Trennung von Layout und Inhalt. Die Art der Darstellung von XML-Dokumenten wird über Fromatvorlagen (*style sheets*) festgelegt. Die Art der Visualisierung kann z.B. auf Anforderung durch den Benutzer oder in Abhängigkeit von den Darstellungsmöglichkeiten des Zielsystems (z.B. WAP, HTML-Browser, Drucker) (auch online) modifiziert werden.

  Das Konzept der Style Sheets erlaubt es, die spezifischen Möglichkeiten der Ausgabemedien optimal zu nutzen ohne am XML-Dokument selbst irgendwelche Veränderungen vornehmen zu müssen. Voraussetzung dafür ist allerdings die vollständige und standardkonforme Implementierung des StyleSheet-Konzepts, die beispielsweise bei den Cascading Style Sheets (CSS) für die verschiedenen Browser nicht gegeben ist.

- **Suche und Agenten**

  XML-Ausschreibungssprachen für spezialisierte Anwendungsbereiche existieren bereits und dienen dort als Quasi-Standards. Darauf aufgebaute Dokumente folgen einer einheitlichen Struktur mit Informationen über die einzelnen Elemente. Dieser Informationen erlauben den etablierten Suchmaschinen eine gezieltere Suche als über die bisher üblichen Schlüsselworte in HTML-Dokumenten. In Zukunft kann auf der Grundlage von XML-Dokumenten die Aufgabe Informationen im Web zu finden, zusammenzutragen und zu vergleichen auch von autonom agierenden Programmen wahrgenommen werden.

- **Datenaustausch zwischen Systemen**

  DataTypeDefinitions (DTD) und Schemata sind die Grundlage von XML-Applikationen. Die vollständige Beschreibung der Struktur eines Dokuments vereinfacht den Austausch und die unmittelbare Weiterverarbeitung strukturierter Daten erheblich. Infolge der rasch voranschreitenden Ausarbeitung von anwenderspezifischen XML-Aplikationen entwickeln sich die entsprechenden DTDs/Schemata zu standardisierten Datenaustauschformaten. Aufgrund der gemeinsamen Basis – XML – kommt es trotz der damit einhergehenden Ausdifferenzierung und Spezialisierung kaum zu Inkompatibilitäten.

Cascading Style Sheets (CSS)

Dem Begriff Style Sheet entspricht im Deutschen jener der Formatvorlage oder Layoutvorlage. Dabei handelt es sich um eine Vorlage zur Umwandlung der logischen Auszeichnungen innerhalb eines Markup-Dokuments in physische Auszeichnungen.

Cascading Style Sheets (CSS) wurde vom World Wide Web Consortium (W3C) für die Verwendung in HTML-Dokumenten standardisiert (aktuelle Version CSS 2, CSS 3 ist in Vorbereitung)

Mit Hilfe von CSS wurde HTML-Funktionalität wesentlich erweitert. Formatvorlagen, wie sie im Textverarbeitungsbereich üblich sind, können damit auf ganz ähnliche Weise in Webdokumenten eingesetzt werden. Wo vor Einführung der CSS Formatierungsanweisungen im gesamten Dokument verändert werden mussten, genügt aufgrund des Style Sheet-Ansatzes eine einzige Änderung. Art und Umfang der Implementierung des CSS-Konzeptes unterscheiden sich jedoch bei den aktuellen Browserversionen teilweise erheblich.

Festlegungen bezüglich der Formatierung gemäß CSS-Spezifikation können Bestandteil eines HTML-Dokuments sein oder in einer externen Datei zusammengefasst werden. In jedem Fall unterstützten sie durch die Verringerung (bzw. im Idealfall sogar Vermeidung) von Formatierungsanweisungen innerhalb von HTML-Tags die Trennung von Inhalten und Darstellungs-/Formatierungsvorschriften.

In der CSS-Definition des HTML-Dokuments werden Element-Eigenschaften mit Gültigkeit für das gesamte Dokument bzw. auch mehrere Dokumente festgelegt. So wird im nachfolgenden Beispiel angegeben, dass eine Überschrift erster Ordnung in der Schrift »Arial« Punktgröße 28 in Fettschrift vor blauem Hintergrund ausgegeben wird. Sofern der Browser CSS unterstützt, werden anstelle der voreingestellten Formatierung Elemente in der definierten Form angezeigt.

```
<STYLE TYPE="text/css">
H1 {
  background-color: #000080;
  font-size: 28pt;
  font-family: Arial;
}
</STYLE>
```

Cascading Stile Sheets können direkt in den HTML-Sourcecode integriert werden bzw. wie im folgenden Beispiel als externe Datei in Form eines Links einem Dokument zugeordnet werden:

```
<LINK REL="stylesheet" TYPE="text/css" HREF="Projekte.css">
```

Neben der Definition von neuen Formaten für bereits bestehende HTML-Tags können mit Hilfe der CSS auch sogenannte Klassen gebildet werden.

```
<STYLE TYPE="text/css">
  .wichtig
   {background-color : yellow; color : black }
  .unwichtig
   {background-color : white;  color : grey  }
</STYLE>
```

Um im HTML-Quellcode diese Formatvorlagen (Klasse "wichtig" und "unwichtig") anwenden zu können, muss dem Tag zusätzlich über das Attribut CLASS übergeben werden, welche definierte Klasse verwendet werden soll:

```
 <BODY>
 <H1>Normale Überschrift 1. Ordnung</H1>
 <H1 CLASS="wichtig">Wichtige Überschrift</H1>
```

```
      <H1 CLASS="unwichtig">Nicht so wichtige Überschrift</H1>
      </BODY>
```

vgl.

```
                          Praxis und Referenz
                          Autoren: Oliver Pott / Gunter Wielage
                          Preis: 79,95 DM      Seiten: 384
                          ISBN (Best.Nr.): 3-8272-5737-9
                          Markt & Technik
http://www.mut.de/media/buecher/xmlnt/start.htm (last visited 07.12.00)
```

Die Verwendung von Cascading Style Sheets im Zusammenhang mit XML war seitens W3C von Beginn der XML-Spezifikation an vorgesehen. XML-Dokumente enthalten selbst keine Formatierungsanweisungen. Die Information über die Art der Darstellung einzelner Elemente kann über CSS an das Element „gelinkt“ werden.
Über verschiedene CSS-Dateien ist es folglich möglich, ein und den selben Inhalt eines XML-Dokuments unterschiedlich darzustellen.

CSS Defizite im Zusammenhang mit XML

- CSS ist nicht imstande, eine Einheit (etwa eine Kapitel-Überschrift) zu nehmen und sie an einem anderen Platz in dem Dokument erneut anzuzeigen (etwa in der Kopfzeile).
- CSS kann keine Zwillings-Beziehungen schaffen. Zum Beispiel ist es nicht möglich, eine CSS-Formatvorlage zu erstellen, die jeden zweiten Absatz im Fettdruck erscheinen läßt.
- CSS ist keine Programmiersprache. Es werden keine IF-Abfragen unterstützt; Erweiterungen durch den Ersteller der Formatvorlagen sind nicht möglich.
- CSS verfügt nicht über die Fähigkeit, Einheiten durchzuzählen und Werte in Variablen zu speichern. Das bedeutet, daß es nicht einmal möglich ist, gebräuchliche Parameter an einem Platz zu versammeln, an dem sie leicht verändert werden können.
- CSS kann nicht eigenständig Text hervorbringen (etwa Seitenzahlen und dergleichen).
- CSS liefert ein simples Kasten-orientiertes Formatierungs-Modell, das für die bisherigen Web-Browser gut geeignet ist, sich aber nicht für die fortgeschritteneren Auszeichnungsmöglichkeiten ausweiten läßt.
- CSS ist auf die Sprachen der westlichen Welt ausgerichtet und setzt voraus, daß die Schriftzeichen in einer horizontalen Abfolge gesetzt werden.
  (Quelle: Das Web automatisieren mit XML; http://members.aol.com/xmldoku/index.htm
  Copyright 1998, Kuno Dünhölter / last visited 07.12.00)

Siehe auch:

Hakon Wium LIE, Bert BOS (1999): Cascading Style Sheets – Designing for the Web, 2nd Edition, Addison-Wesley.

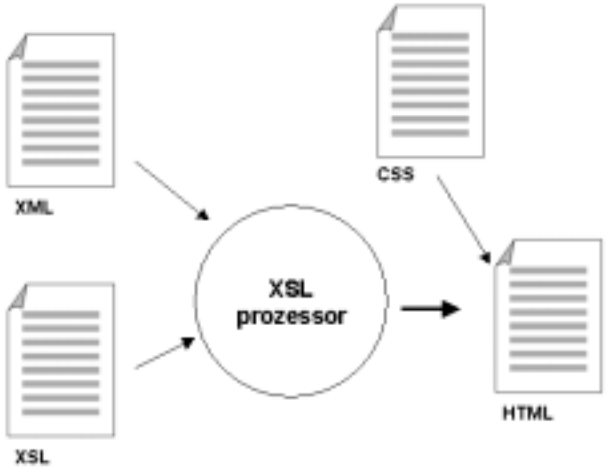
Extensible Style Sheet Language (XSL)

Die o.g. Einschränkungen von CSS in Bezug auf XML führten seitens des W3C zur Entwicklung eines Style-Sheet-Mechanismus mit der Bezeichnung **eXtensible Style Sheet Language** (XSL). XSL ist von der Document Style Semantics and Specification Language (DSSSL) abgeleitet, einer Formatvorlagen-Sprache, die ihre Ursprünge in der SGML-Gemeinde hat.

Ergänzend zu den Cascading Style Sheets übernimmt XSL die Funktion von „aktiven" XML-spezifischen Formatvorlagen. XSL bestimmt – gemeinsam mit CSS – das Seitenlayout oder wandelt Dokumente beispielsweise in HTML-fähige Konstrukte um.

Wesentliche Grundzüge von CSS bzw. der neueren Version CSS 2.0 fanden in XSL Eingang. Als XML-Anwendung ist XSL im Prinzip nach den auch für XML gültigen Grundsätzen aufgebaut. Die bereits bei XML angeführten Vorteile wie klare Struktur, einfache Interpretierbarkeit gelten daher auch für XSL sinngemäß.

Im Gegensatz zu CSS verfügt XSL selbst über typische Elemente einer Programmiersprache, so ist es z.B. möglich bedingte Abfragen zu definieren oder Variablen zu verwenden. Das Dokumentmodell gestattet es (wie bei CSS) die Struktur des XSL-Dokuments zur Laufzeit zu modifizieren und die Veränderungen auf XML-Daten anzuwenden. Diese Transformation kann mit Hilfe des selben (!) XSL-Stylesheets sowohl server- als auch clientseitig erfolgen.



Die Verwendung von XSL unterstützt die bei der Entwicklung von XML intendierte klare Trennung der drei Ebenen:

- Daten → XML
- Struktur → XSL
- Layout → CSS

Die Aufgaben von XSL liegen vor allem in zwei Bereichen:

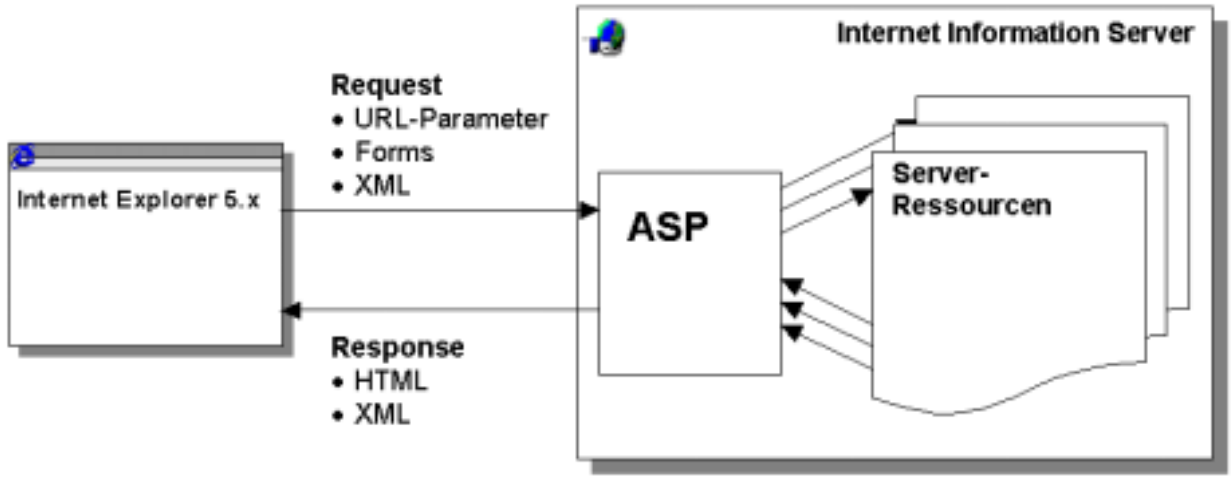
- Die Bildung einer Sprache, mit der sich XML-Dokumente in andere Formate konvertieren lassen (Beispielsweise in HTML oder "anderes" XML, aber eigentlich in beliebige Zielsprachen).
- Zur Verfügung stellen eines Vokabulars, das den semantischen Tags bestimmte Formatierungen zuweist (ähnlich CSS).

(vgl. Praxis und Referenz
Autoren: Oliver Pott / Gunter Wielage
Preis: 79,95 DM Seiten: 384
ISBN (Best.Nr.): 3-8272-5737-9
Markt & Technik
http://www.mut.de/media/buecher/xmlnt/start.htm (last visited 07.12.00)

Active Server Pages (ASP)

Active Server Pages (ASP) ist Microsofts Web-Server-Technologie für komplexe Web-Applikationen. Dieser Ansatz gestattet die Entwicklung von skriptbasierten serverseitigen Anwendungen. Standardmäßig unterstützt ASP lediglich VBScript und JScript/JavaScript.

ASP ist integraler Bestandteil der Active-Plattform - Microsofts Internet-Strategie. Die Active-Plattform basiert auf einer Anzahl von Sprachen, Standards und Services, die dazu benutzt werden können, um entweder Active-Desktop-(Client-Seite) oder Active-Server-(Server-Seite) Applikationen zu erstellen.



Vor der Einführung von ASP (und verwandten Technologien auf anderen Plattformen, wie z.B. PRP) war es notwendig, typische interaktive Web-Applikationen zu kompilieren, um diese ausführbar zu machen. Nach der Kompilation des entsprechenden Programms, musste es in das CGI-Verzeichnis des Web-Servers kopiert werden. Jede Modifikation am Code machte eine neue Kompilierung der aktuellen Applikation notwendig.

Im Unterschied dazu ist aufgrund des skriptsprachen-basierten Ansatzes bei ASP keine Kompilierung notwendig. Sobald eine Datei mit der Endung .asp auf einem entsprechend konfigurierten Internet Information Server in einem IIS-Ordner vorhanden ist, wird die ASP-Applikation serverseitig verarbeitet und das Resultat (das ist in der Regel HTML- oder XML-Code) wird anschließend zum Client übermittelt, der die entsprechende ASP-Datei angefordert hat.

Die Vorteile der Benutzung von ASP

- ASP ist einfach zu erlernen. →
  VB-Script und JScript sind weit verbreitet und haben hohen Bekanntheitsgrad
- ASP-Umgebungen sind erweiterbar →
  ASP ist in seinem Funktionsumfang erweiterbar und wird mit verschiedenen Build-in-Komponenten ausgeliefert, z.B. ADO - Datenbankanbindung
- ASP schützt Algorithmen und Informationen →
  die Inhalte der ASP-Dokumente bleiben gegenüber dem Browser immer verborgen und damit vor unbefugtem Zugriff sicher

Konzeptuelles Design von FIS-Komponenten

FIS-Komponenten

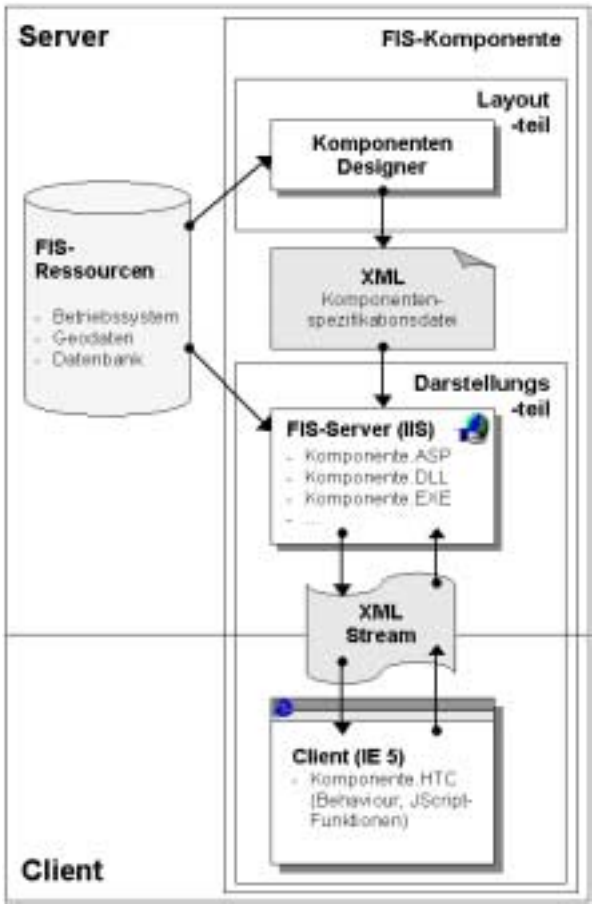
Konzeptive Sicht

Aufgabe des FIS ist es letztlich, den Benutzern jene Strukturen und Ressourcen zur Verfügung zu stellen, die ihn in die Lage versetzen mit Hilfe eines konventionellen Browsers (aktuell IE 5) den Datenbestand der MA22 intensiv und im Rahmen analytischer Fragestellungen explorativ zu nutzen.

In einer umsetzungsorientierten Betrachtungsweise sind damit Teilaufgaben verbunden, wie z.B. die interaktive Darstellung von Karten und Tabellen, die Abbildung von hierarchischen Strukturen oder auch die Suche nach Schlagworten und den damit verbundenen Informationen. Diese Teilaufgaben werden seitens des Systems in Form von **Komponenten** gebündelt, die somit als modulare funktionale Einheiten betrachtet werden können.

Generell sind in jedem Web-basierten Informationssystem sämtliche Inhalte im Entwurfs- und Definitionsschritt dergestalt aufzubereiten, dass sie auf dem Web-Server in abrufbarer Form zur Verfügung gestellt werden können. In einem dynamischen Konzept werden darüber hinaus die an den Browser zu übergebenden Inhalte zumindest teilweise erst im Anschluss an die Benutzeranforderung generiert.

Innerhalb des FIS sollen statische und dynamische Elemente zu einem integrierten Informationssystem verbunden werden. Die FIS-Komponenten müssen daher in der Lage sein, Funktionalität für folgende Aufgaben zur Verfügung zu stellen:



- Spezifikation des Inhalts und Layouts im Entwurfsprozess:

  ← **Layoutteil** der Komponente

- Bereitstellung von Ressourcen und deren browserkonforme Aufbereitung auf Verlangen des Endbenutzers:

  ← **Darstellungsteil** der Komponente

  Der Darstellungsteil einer FIS-Komponente gliedert sich wiederum in eine

  ← **serverseitige**

  und in eine

  ← **clientseitige**

  Teilkomponente.

Aufgaben innerhalb der Komponenten

Layoutteil

Zentrales Element des Layoutteils der Komponente ist der Komponenten-Designer. Innerhalb jeder Komponente existiert ein adäquater Werkzeugsatz, mit dessen Hilfe die Layoutdefinition erfolgt.

Unter Layoutdefinition ist die Auswahl, Strukturierung und Gestaltung der über die Komponente prinzipiell zu spezifizierenden Inhalte im jeweils bestehenden Rahmen der Gegebenheiten und Möglichkeiten zu verstehen.

Komponentenspezifikationsdatei

Die im Layoutprozess getroffenen Festlegungen werden in der Komponentenspezifikationsdatei abgelegt. Sie enthält eine formale Beschreibung der im Browser darzustellenden Inhalte in XML-Syntax.

Darstellungsteil

Aufgabe des Darstellungsteils ist es, dem Benutzer die abgerufenen FIS-Dokumentinhalte zur Verfügung zu stellen. Dazu ist es notwendig,

- die Inhalte der im Layoutteil festgelegten XML-Spezifikationsdatei zu interpretieren,
- die darin definierten Ressourcen bereitzustellen und
- diese mit Hilfe server- und clientseitiger Technologien in vom Browser interpretierbaren HTML-Code umzuwandeln,

und weiters

- vom Benutzer interaktiv ausgelöste Veränderungen der Spezifikationen zu übernehmen, zu verarbeiten und deren Ergebnisse wiederum browserkonform umzusetzen, und
- dabei die XML-basierte Kommunikation zwischen Client und Server vom Benutzer unbemerkt zu administrieren.

Aufgabenverteilung zwischen Server und Client

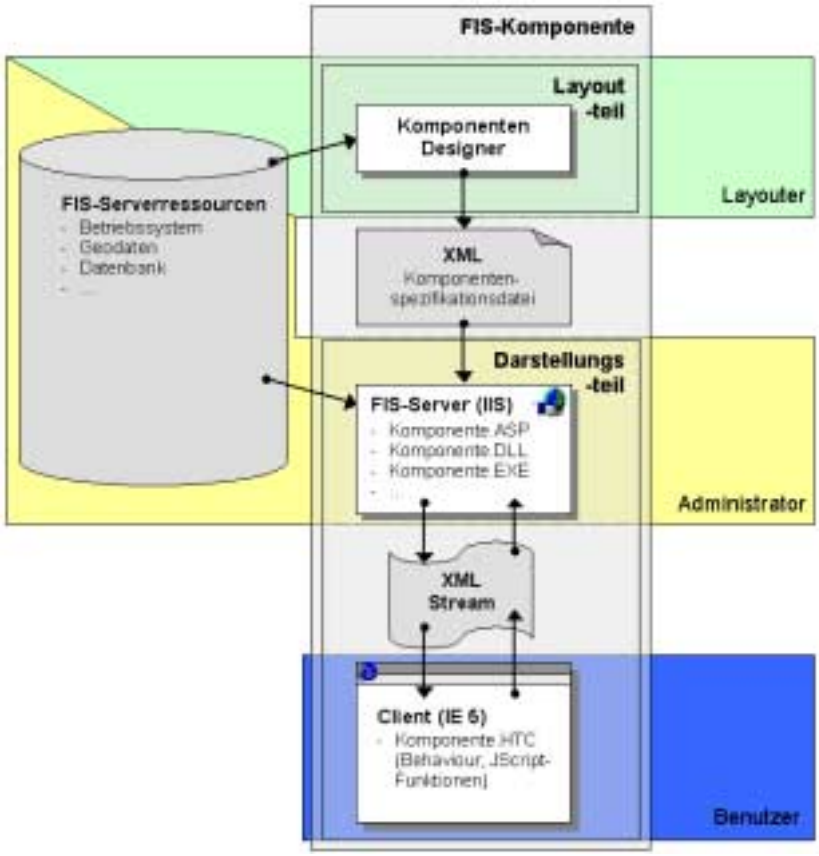
An den mit diesen Aufgaben verbundenen Prozessen können im FIS aufgrund der verwendeten Technologien sowohl Server als auch Client im unterschiedlichen Ausmaß beteiligt sein. Wo genau die Grenze zwischen Client und Server ggf. gezogen wird, ist abhängig von verschiedenen Rahmenbedingungen, wie z.B.

- der eingesetzten Hardware (v.a. Server und Netzwerkverbindung),
- der eingesetzten Software
- der Programmierkapazität im Betrieb,
- der Häufigkeit des Einsatzes, sowie
- sonstiger im speziellen Fall relevante Rahmenbedingungen.

Diese für die erfolgreiche Einführung und den zufriedenstellenden Alltagsbetrieb eines Informationssystems immens wichtigen Aspekte sind jedoch in erster Linie eine Frage der Implementierung und weniger des Konzeptes. Sie werden daher ab Seite 24 im Kapitel "Implementierungsplattformen" näher beleuchtet.

Administrative Sicht

Bei dieser Betrachtungsweise geht es vorrangig um die Zuteilung von administrativen Aufgaben an handelnde Personen. Im Rahmen des FIS umfasst dies die Gruppe der Layouter, den FIS-Systemadministrator und – wenn man will auch – die (End-)Benutzer des Systems.



Die Aufgabenzuteilung zu den drei Gruppen stellt sich im Großen und Ganzen wie folgt dar:

Layouter:

- erzeugt Web-Dokumente (darunter auch welche mit eingebetteten FIS-Komponenten)
- erstellt und modifiziert FIS-Systemressourcen (z.B. definiert eine View in WUIS/Oracle)
- erstellt XML-Komponenten-Spezifikationsdateien (mit den Layoutwerkzeugen)
- publiziert diese Dokumente (über die FIS-TreeView-Komponente, siehe Seite 28ff)

Administrator:

- vergibt Rechte im Betriebssystem (u.a. an die Layouter und die Benutzer)
- stellt alle Ressourcen am Server (z.B. Geodaten) über die Dateiverwaltungsroutinen des Betriebssystems zur Verfügung.
- verwaltet systemweite ODBC-Verbindungen zu Datenbanken
- stellt die FIS-Komponenten über den Internet-Information-Server (IIS) und den Microsoft-Transaction-Server (MTS) zur Verfügung

Benutzer:

- hat eigentlich keine administrativen Aufgaben im engeren Sinn. Ihm bleibt die zweck-gerichtete Auswahl und Analyse von Inhalten und Daten über einen Internet-Browser

Detailentwurf der FIS-Komponenten

Einleitung

Überblick

Das im vorigen Abschnitt entwickelte allgemeine Konzept wird nun anhand spezifischer FIS-Komponenten konkretisiert. Diese Konkretisierung erfolgt für Komponenten, welche aus der magistratsinternen Voruntersuchung unter Berücksichtigung der eingangs genannten allgemeinen und operationalisierten Ziele abgeleitet wurden.

In Zusammenarbeit und Übereinstimmung mit dem Auftraggeber haben sich aufgrund ihrer zentralen Aufgaben und Bedeutung letztlich drei im FIS vorrangig zu implementierende Komponenten herauskristallisiert – hier ein Überblick:

		Detailinformation siehe ab Seite		**Implementierung im Prototyp**	
Komponente	**Aufgabe**	Detail-entwurf	XML-Spezifikation	Layout-teil	Darstellungs-teil
TreeView	Direkter, generisch erzeugter und hierarchisch strukturierter Zugriff auf Inhalte (Dokumente) des FIS	27	31	-	+
TableView	Publikation, Visualisierung und Abfrage alphanumerischer, in tabellarischer Form abgelegter Daten (Datenbanken)	39	41	-	+
MapView	Publikation, Visualisierung und Abfrage von Daten mit konkretem Raumbezug in Form von interaktiven Web-Karten.	51	56	+	+

Die anschließende Konkretisierung der oben drei genannten Komponenten erfolgt jeweils in derselben systematischen Gliederung:

- Zweck und Aufbau
- Layoutteil
  - Entwurfsziel
  - Funktionalität
  - Implementierung
- XML-Syntax der Spezifikationsdatei
- Darstellungsteil
  - Entwurfsziel
  - Funktionalität
  - Implementierung
- Integration in eine HTML-Seite
- Möglichkeiten zur Weiterentwicklung

Zuvor führen noch generelle Betrachtungen zu verschiedenen Implementierungsoptionen und den damit verbundenen Werkzeugen und Aufwänden in die Thematik des konkreten Komponentenentwurfs ein, gefolgt von einigen allgemeinen Festlegungen zur XML-Syntax der FIS-Komponenten-Spezifikationsdateien.

Implementierungsplattformen

Datenzugriff

Der Datenbankzugriff erfolgt im FIS generell via ODBC (*Open DataBase Connectivity*). Die Vorteile von ODBC stellen sich wie folgt dar:

- OBDC bietet einen einheitlichen Zugriffsmechanismus zu verschiedenen Datenbanktechnologien. Die Spannweite reicht von einfachen Textdateien über dBase- und Excel-Files bis hin zu Datenbanksystemen wie MS-Access und im professionellen Bereich Oracle.
- Diese verschiedenen Datenbankarchitekturen sind mit den selben Softwarewerkzeugen (z.B. ADO – *Active-Data-Objects* in Zusammenwirken mit Visual Basic) verarbeitbar.

An sich ergeben sich beim ODBC-Zugriff via ADO keinerlei Probleme. Im Rahmen der FIS-TabeView-Komponente zeitigt ODBC zudem ein akzeptables Laufzeitverhalten. Die Ausnahme davon bildet jedoch der ODBC-Zugriff von MapObjects2. Für die ausführliche Behandlung der diesbezüglichen Probleme siehe das Kapitel "Anbindung externer Tabellen" auf Seite 52.

Layoutteil

Der Layoutteil von Komponenten wird i.a. an zentraler Stelle ausgeführt. Meist geschieht dies direkt am Server, jedenfalls aber in einer relativ homogenen Softwareumgebung. Es kann also bei der Wahl des Entwicklungswerkzeugs entsprechend pragmatisch vorgegangen werden.

Oberstes Grundprinzip ist die möglichst friktionsfreie Einbettung in die gewohnte Arbeitsumgebung des jeweiligen Autors (Sachbearbeiter, Web-Designer). Deshalb wurde z.B. das Produkt ArcView-GIS-3.1 als Plattform für den Layoutteil der FIS-MapView-Komponente eingesetzt, um dem Sachbearbeiter die Publikation seiner kartographischen Arbeitsergebnisse ins FIS mit nur wenigen Mausklicks aus (s)einem Standardprogramm heraus zu erlauben.

Layoutteile weiterer Komponenten (deren Implementierung nicht im gegenständlichen Auftrag inkludiert ist) sollten ebenfalls möglichst gut in die Arbeitsumgebung potentieller Autoren integriert werden. Als Werkzeuge hierfür bieten sich u.a. Visual Basic 6.0 und VBA (*Visual-Basic-for-Applications*, z.B. für Frontpage- oder Access-Erweiterungen) an.

Darstellungsteil

Serverseitige Softwarewerkzeuge

Die zentrale serverseitige Funktion jeder Komponente nimmt ein ASP-(*Active-Server-Page*)-Script wahr, welches im Wesentlichen zwei Aufgaben erfüllt:

- **Den Zugriff auf FIS-Resourcen am Server und deren Bereitstellung**

  Diese Funktion kann entweder direkt von der ASP-Datei erfüllt werden (wie im Fall der TableView-Komponente) oder mit Hilfe einer ActiveX-DLL-Datei (wie im Fall der TreeView-Komponente) oder einer ActiveX-EXE-Datei (wie im Fall der MapView-Komponente).

  Der Vorteil der Codierung mit VB-Script direkt in der ASP-Datei ist die relativ kurze Entwicklungszeit (*rapid prototyping*), der Vorteil bei der Einbindung von ActiveX-Objekten liegt im besseren Laufzeitverhalten (Script-Code in einer ASP-Seite wird interpretiert und ist daher eher langsam, ActiveX-Objekte sind kompiliert und sind vergleichsweise schneller).

- **Die Kommunikation mit dem Client**

  Entgegennehmen von Client-Requests, deren Verarbeitung und Zurücksenden des Ergebnisses an den Client.

Zwischenablage für Image-Dateien

Normalerweise schickt das ASP-Script sein Ergebnis (i.e. entweder HTML- oder XML-Code) direkt an den Client zurück (d.h. ohne Zwischenspeichern in einer temporären Server-Datei).

Im Falle der MapView-Komponente kann diese Vorgehensweise nur begrenzt eingesetzt werden, weil dabei auch Ergebnisse in Form von Bilddaten anfallen (JPG- und PNG-Dateien), welche dem Client nicht direkt, sondern nur über die Angabe einer URL in einem entsprechenden IMG-Tag übermittelt werden können (<IMG src="...url..."">). Das zieht die Notwendigkeit, diese Bilddateien am Server zwischenzuspeichern, zwingend nach sich.

Jedem Client, der eine FIS-Arbeitssitzung beginnt, wird in diesem Moment automatisch vom Internet Information Server (IIS) eine SessionID zugeordnet, über die er während seiner gesamten Arbeitssitzung eindeutig identifiziert werden kann. Es wird im FIS für jede Session im virtuellen Verzeichnis **/fis_temp** automatisch ein **Unterordner mit der SessionID als Namen** angelegt, welcher als temporärer Zwischenspeicher dient, und der beim Beenden der Session automatisch wieder gelöscht wird (siehe dazu auch die Installationsdokumentation).

Clientseitige Softwarewerkzeuge

Die Auswahl von clientseitigen Softwarewerkzeugen ist ungleich diffiziler als auf dem Server, da hier grundsätzlich von einer extrem heterogenen Softwareumgebung auszugehen ist. Die Bandbreite reicht von schlimmstenfalls total divergierenden und inkompatiblen Softwarearchitekturen (z.B. Apple, Windows, UNIX) bis bestenfalls "nur" unterschiedlichen HTML- und JavaScript-Dialekten (z.B. Netscape vs. InternetExplorer).

Die Grenze zwischen client- und serverseitiger Verarbeitung von Benutzerrequest ist prinzipiell verschiebbar. Auswirkungen ergeben sich dabei vor allem in Hinsicht auf

- das Laufzeitverhalten der Komponenten,
- die Belastung des Servers und
- den Programmieraufwand

Es folgt eine kurze Zusammenstellung der jeweiligen Vor- und Nachteile von zwei einander diametral gegenüberstehenden Konstellationen:

Konstellation	**Vorteile**	**Nachteile**
Thick Server Thin Client	Verschiedene Browser sind rel. leicht bedienbar, weil viele Requests zentral am Server verarbeitet werden. Einfache Software-Wartung.	Hohe Server- und viel Netzwerkbelastung, was zu sehr schlechtem Antwortzeitverhalten des Gesamtsystems führt. Dadurch negativer Einfluss auf die Benutzerakzeptanz
Thin Server Thick Client	Hohe Interaktivität durch browserseitige Scripts Geringe Server- und Netzwerkbelastung In Summe schnelles Laufzeitverhalten und daher eine hohe zu erwartende Benutzerakzeptanz	Hoher Programmieraufwand, aufgrund der Notwendigkeit clientseitigen Code an viele Browser anzupassen, entweder durch (a) eine "Ver-n-fachung" des Codes (bei n verschiedenen Browsern), oder durch (b) den Aufbau von abstrakten (i.e. browserunabhängigen) Datenstrukturen und Methoden, wobei manche browserseitige "Goodies" verloren gehen könnten.

Implementierung im Prototyp

Die Implementierung erfolgte vereinbarungsgemäß auf Basis des Microsoft Internet Explorer Version 5.0 (IE5) oder höher (empfohlen 5.5), wobei bei der Verarbeitung von Benutzerrequests der Schwerpunkt im Prototyp eher auf der Clientseite liegt.

Hier die Vorteile des gewählten Ansatzes:

- Wenig Netzwerkverkehr zwischen Client und Server.

  Daten werden in der Regel nur einmal an den Client geschickt, dort dann lokal verarbeitet und in entsprechenden HTML-Code übersetzt.
- Daraus resultierten spürbare Geschwindigkeitsvorteile bei der interaktiven Bearbeitung der Requests aufgrund der Verringerung notwendiger Serveranforderungen.
- Die Serverbelastung hält sich infolgedessen ebenfalls in Grenzen, was ein Mehr an parallelen Zugriffen (Sessions) ohne spürbare Zeitverzögerung gestattet
- Clientseitig eingesetzte JavaScript-Funktion müssen nur auf den Internet Explorer 5.x abgestimmt sein.
- Voller XML-Support im IE5
- Verfügbarkeit spezieller auf XML aufbauender Werkzeuge, wie VML (*Vector Markup Language*, siehe http://www.w3.org/TR/NOTE-VML) für interaktive Vektorgraphiken und *Custom Behaviors* zur objektorientierten Spezifikation von benutzerdefinierten HTML- (bzw. XML-) Tags.

Ausblick

Falls künftig, aus welchen Gründen auch immer, der Zugang zum FIS auf andere Browsersoftware als den Internet-Explorer von Microsoft ausgedehnt werden sollte (z.B. Netscape Navigator, Opera-Browser), ist das zwar prinzipiell denkbar und möglich, jedenfalls aber mit beträchtlichen und womöglich überproportionalen Aufwänden verbunden.

Es sollten daher auch alternative Konzepte erwogen werden, welche von der Auftraggeberseite im gegenständlichen Auftrag apriori ausgeschlossen wurden, wie z.B. die Implementierung von Komponenten mit Hilfe von in Java geschriebenen **Applets**. Applets haben im vorliegenden Zusammenhang den Vorteil, dass sie auf allen Java-fähigen Browsern einheitlich ablaufen und der Programmieraufwand somit in Grenzen gehalten werden kann.

Weiters sollte auch das seit Mitte 2000 kommerziell verfügbare Produkt **ArcIMS 3.0** der Firma ESRI eingehend auf Funktionalität und Einsatztauglichkeit im Rahmen der Fachabteilungen der MA22 geprüft werden. ArcIMS ist dem Grundgedanken nach ähnlich aufgebaut wie FIS und verwendet ähnliche Technologien wie z.B. XML-Spezifikationsdateien für Internet-Karten. Da ArcIMS vom selben Softwarehersteller wie ArcView, ArcINFO und MapObjects stammt, ist eine volle Kompatibilität bezüglich der geographischen Datenbestände gewährleistet.

Für weitere Informationen zu ArcIMS siehe http://www.esri.com/software/arcims/index.html.

Generelle Codierungsrichtlinien für XML-Spezifiationsdateien

In weiterer Folge werden in entsprechenden Unterkapiteln die jeweils notwendigen und optionalen Angaben in den XML-Spezifikationsdateien für die im Prototyp implementierten FIS-Komponenten im Detail erläutert.

Generelle Codierungsrichtlinien und Namenskonventionen, welche für alle Komponenten gleichermaßen gelten, werden an dieser Stelle vorab zusammengefasst:

- Sämtliche FIS-XML-Dateien müssen wohlgeformten (*well-formed*) XML-Code enthalten.
- Zu Beginn jeder FIS-Komponenten-Spezifikationsdatei ist eine *XML-Declaration* erforderlich, welche neben der zugrundeliegenden XML-Version (version="1.0") auch den im Dokument verwendeten Zeichensatz festlegt (encoding="ISO-8859-1").

  ```
  <?xml version="1.0" encoding="ISO-8859-1"?>
  ```

  Der "ISO-8859-1"-Zeichensatz erlaubt die Verwendung von Spezifika der deutschen Sprache wie Umlaute und das Scharfe S.
- Namen von XML-Elementen sind generell mit Großbuchstaben anzugeben. Bsp.:

  ```
  <TABLE> .... </TABLE>
  ```
- Attribute von XML-Elementen beginnen grundsätzlich mit einem Großbuchbuchstaben, die weiteren Buchstaben werden in der Regel klein geschrieben. Ausnahmen bestehen in wenigen Fällen zu Verbesserung der Lesbarkeit von langen Attributname zu Beginn eines "Teilwortes". Bsp.:

  ```
  <TABLE Title="..."> ... </TABLE>

  <TREEVIEW OpenLevel="..."> ... </TREEVIEW>
  ```
- Attributwerte werden, so sie FIS-Standard-Codewörter enthalten, vollständig in Kleinbuchstaben geschrieben. Bsp.:

  ```
  <PATTERN Case="sensitive"> ... </PATTERN>

  <LAYER Visible="true" ...> ... </LAYER>
  ```
- Die Reihenfolge der Attribute innerhalb von Elementen ist beliebig. Bsp.:

  Die XML-Struktur

  ```
  <TREEVIEW Style="list"
            Root="/fis/content" ...>
   ...
  </TREEVIEW>
  ```

  ist gleichbedeutend mit

  ```
  <TREEVIEW Root="/fis/content"
            Style="list" ...>
   ...
  </TREEVIEW>
  ```
- Die Reihenfolge von geschachtelten Sub-Elementen innerhalb ihrer direkt übergeordneten Elemente ist grundsätzlich beliebig, außer wenn im Dokumentationskapitel explizit Gegenteiliges angesprochen wird.

  Die XML-Struktur

  ```
  <EXTENT>
     <TOP>2000</TOP>
     <BOTTOM>1000</BOTTOM>
     <LEFT>2000</LEFT>
     <RIGHT>4000</RIGHT>
  </EXTENT>
  ```

  ist gleichbedeutend mit

  ```
  <EXTENT>
     <RIGHT>4000</RIGHT>
     <BOTTOM>1000</BOTTOM>
     <LEFT>2000</LEFT>
     <TOP>2000</TOP>
  </EXTENT>
  ```
- Dezimalzahlen müssen generell mit einem Dezimalpunkt anstelle des hierzulande üblichen Dezimalkommas angegeben werden.

  ```
  <POINT X="12.3" Y="4.56" />
  ```

TreeView

Zweck und Aufbau

Der Hintergrund, vor dem die FIS-TreeView-Komponente entworfen wurde, ist das Bestreben, die Veröffentlichung von Dokumenten im FIS grundsätzlich mit minimalem Aufwand zu verbinden und somit eine möglichst rasche Publikation von Inhalten zu gewährleisten.

Rahmenbedingungen und Motivation

Als Rahmenbedingungen sollen vor allem zwei Aspekte in den Vordergrund gestellt werden:

- Dokumente bzw. Dokumentgruppen stehen ihrer inhaltlichen Logik nach oft in einem hierarchischen Zusammenhang.
- Web-Designer erstellen Web-Dokumente, welche in Form von speziellen Dateien (z.B. *.html) an dafür vorgesehene, klar bestimmte Stellen im Betriebssystem abgelegt werden.

Ein Zugriff auf Web-Dokumente erfolgt – technisch betrachtet – entweder über

(1) die Eingabe der entsprechenden URL in das Adressfeld des Browsers oder
(2) einen Verweis aus einem anderen Web-Dokument.

Die Möglichkeit (1) der händischen Eingabe der URL fällt für eine wirkungsvolle und geordnete Art des Zugriffs in einem nach sachlich-inhaltlichen Gesichtspunkten organisierten und auf Erweiterung ausgelegten Informationssystem aus naheliegenden Gründen von vorne herein aus.

Bei der Zugriffsart (2) sind bezüglich der Zugriffsart folgende zwei Fälle zu unterscheiden:

(2a) Der Zugriff erfolgt über einen statischen Verweis.
(2b) Der Zugriff erfolgt über einen dynamisch generierten Verweis.

Im Fall (2a) stößt der Autor bei der Veröffentlichung auf das Problem, dass apriori keine Verweise auf seine neuen Dokumente existieren, und diese somit gleichsam "in der Luft hängen". Derartige Verweise müssen aufgrund ihrer statischen Natur von Hand aus und in der Regel einzeln in bereits bestehende Dokumente eingearbeitet werden.

Der Fall (2b) hingegen – wie er z.B bei Web-Suchmaschinen vorliegt – bietet die geforderte generische Flexibilität, um allein aus der Existenz eines Dokuments die notwendigen Verweise zur Laufzeit ("*on the fly*") zu generieren.

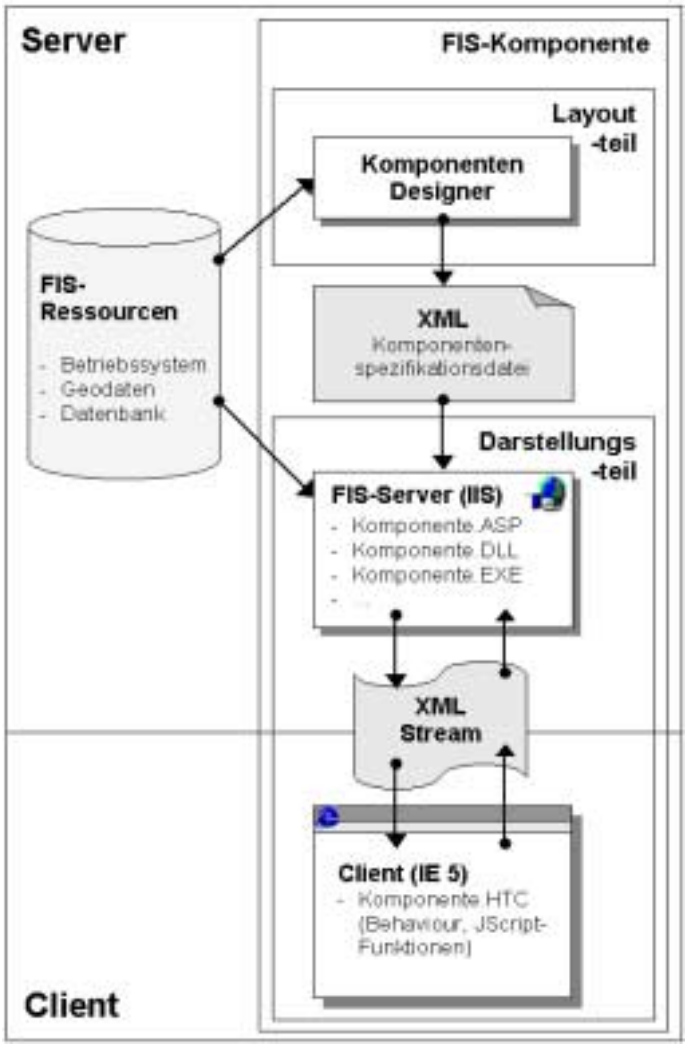
Operationalisierung

Eine einfache und wirkungsvolle Lösung für dieses Zugriffsproblem besteht aus einer vordefinierten hierarchischen Struktur, welche einerseits als Publikationsplattform für den Produzenten der Information dient (i.e. für den Autor, Web-Designer) und welche gleichzeitig als automatisierter Zugriffspfad für den Konsumenten der Information herangezogen werden kann (i.e. für den Leser, Web-Surfer).

Als eine solche Plattform bietet sich die hierarchische Dateistruktur des Betriebssystems in Kombination mit einem Werkzeug an, welches automatisch Verweise auf darin enthaltene Dokumente erzeugt, und diese dem Benutzer dynamisch (d.h. erst zum Zeitpunkt des Zugriffs berechnet) und hierarchisch gegliedert im Browser anbietet. Dokumente innerhalb dieser Struktur sind also implizit und ad hoc veröffentlicht.

Im FIS wird diese Komponente bereits in der Voruntersuchung FIS-TreeView genannt.

Die FIS-Komponente TreeView besteht aus drei Teilen, von denen zwei prototypisch im Rahmen dieses Forschungsauftrags implementiert wurden:



- Layoutteil
  (nicht implementiert,
  die XML-Spezifikationsdatei kann jedoch mit geringem Aufwand mittels eines Texteditors generiert und gewartet werden, zur Syntax siehe Seite 31ff)
- serverseitiger Darstellungsteil
  (*TreeView.asp, DirTree.dll*)
- clientseitiger Darstellungsteil
  (*TreeView.htc, TreeView_List.xsl, TreeView_Block.xsl TreeView.css*)

Die serverseitig über das ASP-Script *TreeView.asp* der (mit Visual Basic entwickelten) Klasse *TreeView.dll* übergebene XML-Komponenten-Spezifikation wird von dieser in eine interne XML-Struktur des gewünschten Teils des FIS-Dateibaumes übersetzt, welche anschließend dem Client übergeben wird.

Der clientseitige Darstellungsteil wurde in Form eines *Custom Behaviors* implementiert (*Tree-View.htc*), welches über die CSS-Vorlage *TreeView.css* in ein beliebiges HTML-Dokument integriert werden kann. Der vom Server übernommene XML-Baum wird mit Hilfe eines der beiden XSL-Stylesheet (*TreeView_List.xsl* bzw. *TreeView_Block.xsl*) in eine hierarchische Liste transformiert und zuletzt im Browserfenster angezeigt.

Layoutteil der FIS-TreeView-Komponente

Entwurfsziel und Funktionalität

Zur Entwurfszeit muss der Layouter prinzipiell angeben können,

(1) welcher Ordner im Betriebssystem (eigentlich: im virtuellen Verzeichnisbaum des FIS) die Wurzel des Baums bildet,

(2) welche Dokumente bzw. Dokumenttypen in den Baum aufgenommen werden sollen und welche nicht,

(3) welche Teilbäume ("Äste", untergeordnete Ordner) in den Baum aufgenommen werden sollen und welche nicht,

(4) auf welche Art und Weise der Baum optisch im Browserfenster umgesetzt werden soll,

(5) in welcher Reihenfolge Dokumente und untergeordnete Teilbäume (Ordner) im Browser angezeigt werden sollen, wenn sie sich im selben übergeordneten Ordner befinden,

(6) bis zu welcher Hierarchieebene sich der Baum bei der ersten Anzeige im Browser automatisch entfalten (öffnen) soll.

Vor allem die Punkte (2) und (3) sind von besonderer Bedeutung. Viele Web-Publishing-Programme (z.B. MS-Frontpage, aber auch MS-Word) legen Hilfsdateien – z.B. Bilder oder HTML-Dokumente, welche in eigenen Subframes angezeigt werden – in speziell benannten Unterordnern ab. Diese Dokumente sollen für sich alleine nicht publiziert werden, sondern sind nur als Teil des Ganzen zu betrachten. Das gezielte Ausblenden dieser Hilfsstrukturen aus dem Direktzugriff ist somit ein unverzichtbares Mittel zur sauberen Veröffentlichung von Dokumenten in einem in sich konsistenten und stimmigen Informationssystem.

Außerdem bietet ein solcher Inkludierungs-Exkludierungs-Mechanismus zumindest grundsätzlich die Möglichkeit zur benutzergruppenspezifischen Publikation von Dokumenten, ohne dass diese physisch dupliziert werden müssen (z.B. über eine geeignete Namenskonvention).

Für die im Punkt (4) angesprochene Art und Weise der visuellen Umsetzung des Baumes im Browser sind grundsätzlich verschiedene Formen vorstellbar, z.B. als in sich geschachtelte Listen, aber auch in Form von menü-ähnliche Strukturen. Jedenfalls besteht hier die Forderung nach einer prinzipiellen Erweiterbarkeit (siehe dazu auch den Abschnitt "Darstellungsteil der FIS-TreeView-Komponente" ab Seite 35 und die "Möglichkeiten zur Weiterentwicklung" ab Seite 37).

Implementierung

Der Layoutteil der FIS-TreeView-Komponente wurde vertragsgemäß im Rahmen dieses Forschungsauftrags nicht implementiert.

Für die Implementierung bei einer allfälligen Weiterentwicklung von FIS bieten sich sämtliche Standardwerkzeuge zur Softwareentwicklung an, welche über Klassen verfügen, die effizienten Zugang zum Windows-Dateisystem gewährleisten (wie z.B. Visual Basic).

XML-Tagbeschreibung einer FIS-TreeView-Spezifikationsdatei

XML-Document-Element (TREEVIEW)

In jeder XML-Datei muss genau ein sog. Document-Element existieren (i.e. das XML-Wurzelelement). Das Document-Element der FIS-TreeView-Komponente lautet TREEVIEW und enthält weitere Angaben in Form von Attributen:

```
<TREEVIEW  Type="folder/file"  Root="..."    Title="..."
           Style="..."         Target="..." OpenLevel="...">
...
</TREEVIEW>
```

Attribut-name	Angabe notwendig?	Beschreibung
Type	NEIN	"file/folder" Wird im aktuellen Prototyp ignoriert und ist reserviert für künftige Erweiterungen der Funktionalität. Für diesbezügliche Vorüberlegungen siehe das entsprechende Kapitel ab Seite 37.
Root	JA	absoluter Pfad ausgehend vom virtuellen Wurzelverzeichnis des IIS am Server zur Wurzel des darzustellenden Verzeichnisbaums in URL-Syntax (i.e. mit "/" anstelle von "\"). Bsp.: Root="/fis/content"
Title	JA	Text, welcher auf der obersten Ebene des Baumes (i.e. die Ebene 0, die Wurzel des Baumes) anstelle des Root-Verzeichnisnamens angezeigt wird.
Style	NEIN	"list" / "block". Defaultwert ist "list". Gibt an, in welcher Form der Baum vom clientseitigen Darstellungsteil visualisiert werden soll. "list": Anzeige als hierarchische Bullet-Liste aus geschachtelten UL- und LI-Elementen "block": Anzeige in einer hierarchisch eingerückten Blockstruktur mit "Schaltflächen-Optik" (die Elemente des Baums ähneln Befehlsschaltflächen) Hinweis für Erweiterungen: Es muss im FIS-System-Ordner */fis/engine/TreeView* eine XSL-Stylesheet-Datei namens "Treeview_***style***.xsl" existieren, welche für die Umsetzung der Baumstruktur aus einer internen XML-Repräsentation in HTML-Verweisanker-Tags verantwortlich ist.
Target	NEIN	Name des Zielfensters oder –frames für die automatisch generierten Verweisanker (<A HREF="..." TARGET="...">...</A>). Der Defaultwert für Target ist "_NEW", d.h. es wird beim Anklicken eines Elementes (eines "Blattes" im Baum) ein neues Fenster am Browserbildschirm geöffnet.
OpenLevel	NEIN	Eine ganze Zahl größergleich 0. Der Defaultwert ist "1" Gibt an, bis zu welcher Hierarchieebene sich der Baum bei seiner ersten Anzeige im Browser entfalten (öffnen) soll.

Spezifikation des TREEVIEW-Inhaltes über PATTERN-Listen

Das TREEVIEW-Element kann (muss aber nicht !) ein FILE- und/oder ein FOLDER-Element enthalten, welche ihrerseits wieder aus IN- und OUT-Elementen mit darin enthaltenen PATTERN-Elementen bestehen können:

```
<FILE>
     <IN>
          <PATTERN> ... </PATTERN>
          ...
          <PATTERN> ... </PATTERN>
     </IN>
     <OUT>
          <PATTERN> ... </PATTERN>
          ...
          <PATTERN> ... </PATTERN>
     </OUT>
</FILE>
```

Die IN- und OUT-PATTERN innerhalb des FILE-Elementes enthalten Suchmuster und steuern, welche Dokumente in die resultierende Baumstruktur aufgenommen werden sollen (IN) und welche nicht (OUT).

```
<FOLDER>
     <IN>
          <PATTERN> ... </PATTERN>
          ...
          <PATTERN> ... </PATTERN>
     </IN>
     <OUT>
          <PATTERN> ... </PATTERN>
          ...
          <PATTERN> ... </PATTERN>
     </OUT>
</FOLDER>
```

Die IN- und OUT-PATTERN innerhalb des FOLDER-Elementes enthalten Suchmuster und steuern, welche untergeordneten Ordner ("Äste", Teilbäume) in die resultierende Baumstruktur aufgenommen werden sollen (IN) und welche nicht (OUT).

FILE-Element

```
<FILE Sort="..."> ... </FILE>
```

Das optionale Attribut *Sort* kann die Werte "*alphabetic*" oder "*pattern*" annehmen. Der Defaultwert für *Sort* ist "*alphabetic*".

Sort="alphabetic" Die Reihenfolge der Einträge in der im Browser angezeigten Liste von Verweisen auf FIS-Dokumente ist alphabetisch aufsteigend.

Sort="pattern" Die Reihenfolge der Einträge in der im Browser angezeigten Liste von Verweisen auf FIS-Dokumente erfolgt in der Reihenfolge der PATTERN-Elemente innerhalb des IN-Elementes (s.u.). Wenn die Namen mehrerer Dokumente demselben Suchmuster entsprechen, werden diese Dokumente untereinander alphabetisch gereiht.

FOLDER-Element

```
<FOLDER Sort="..."> ... </FOLDER>
```

Das optionale Attribut *Sort* kann die Werte "*alphabetic*" oder "*pattern*" annehmen. Der Defaultwert für *Sort* ist "*alphabetic*".

Sort="alphabetic" Die Reihenfolge der im Browser angezeigten Liste von untergeordneten Bäumen ist alphabetisch aufsteigend.

Sort="pattern" Die Reihenfolge der im Browser angezeigten Liste von untergeordneten Bäume erfolgt in der Reihenfolge der PATTERN-Elemente innerhalb des IN-Elementes (s.u.). Wenn die Namen mehrerer Unterordner im gleichen übergeordneten Ordner demselben Suchmuster entsprechen, werden diese untereinander alphabetisch gereiht.

IN-Element

OUT-Element

FILE- und FOLDER-Elemente können (müssen aber nicht !) jeweils ein IN- und/oder ein OUT-Tag enthalten. Innerhalb beider können noch jeweils mehrere PATTERN-Subelemente angegeben werden (s.u. Punkt PATTERN).

- Dokumente oder Ordner, deren Namen einem Suchmuster (PATTERN) innerhalb eines IN-Elementes entsprechen, werden in den Baum aufgenommen.
- Dokumente oder Ordner, deren Namen einem Suchmuster (PATTERN) innerhalb eines OUT-elementes entsprechen, werden nicht in den Baum aufgenommen.
- Ist kein IN-Element angegeben, werden alle Dokumente oder Ordner in den Baum aufgenommen, was gleichbedeutend ist mit der Angabe von

```
<IN>
     <PATTERN>*</PATTERN>
</IN>
```

- Ist ein OUT-Element angegeben, werden sämtliche Dateien oder Ordner in den Baum aufgenommen, deren Namen mit mindestens einem (!) der Suchmustern im IN-Element und mit keinem (!) Suchmuster im OUT-Element übereinstimmen.
- Ist kein oder nur ein leeres OUT-Element angegeben, werden sämtliche Dateien und Ordner in den Baum aufgenommen, deren Namen mit den Suchmustern im IN-Element übereinstimmen.
- Die Reihenfolge der PATTERN-Elemente innerhalb eines IN-Elementes ist genau dann von Bedeutung, falls im übergeordneten FILE- oder FOLDER-Element das Attribut *Sort* den Wert "*pattern*" hat.

PATTERN-Element

```
<PATTERN Case="..."> ...suchmuster... </PATTERN>
```

Der Inhalt eines PATTERN-Elementes muß ein Visual-Basic-kompatibles Suchmuster mit sog. Platzhalterzeichen sein (siehe dazu den Auszug aus der Visual-Basic-Online-Hilfe auf der nächsten Seite oben).

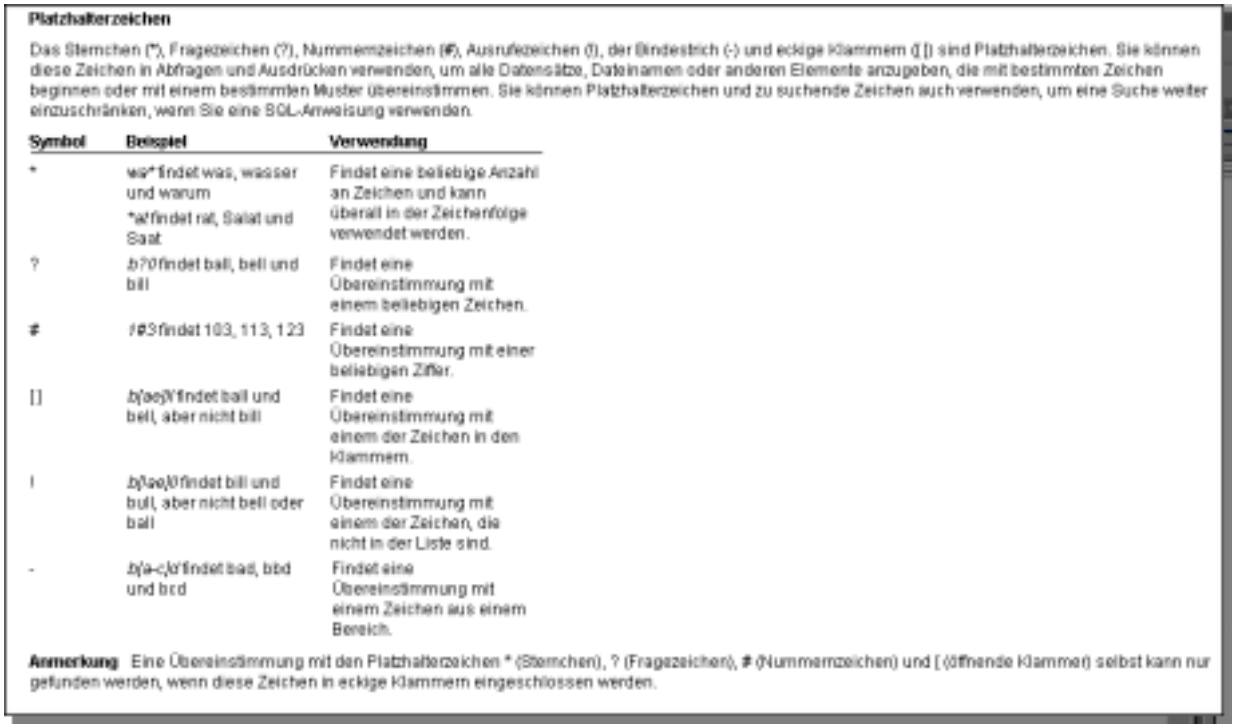
Die einzelnen Namen von Dokumenten (wenn innerhalb eines FILE-Elementes) oder Ordnern (wenn innerhalb eines FOLDER-Elementes) werden mit dem Suchmuster verglichen und entsprechend als Bestandteil des Baums aufgenommen oder eben nicht, je nachdem, ob das PATTERN-Element innerhalb eines IN- oder eines OUT-Elementes plaziert wurde.

Das optionale Attribut *Case* kann die Werte "*sensitive*" oder "*ignore*" annehmen. Der Defaultwert für *Case* ist "*ignore*".

Case="ignore" Die Groß- und Kleinschreibung ist für den Mustervergleich irrelevant.

Case="sensitive" Die Groß- und Kleinschreibung wird beim Mustervergleich berücksichtigt.

Abschließend ein kurzer Abschnitt über die verfügbaren Platzhalterzeichen für den Mustervergleich (String-Operator: *like*) aus der Visual-Basic-Online-Dokumentation, auf die hier auch für weitere Details verwiesen wird (siehe nächste Seite):



Platzhalterzeichen

Das Sternchen (*), Fragezeichen (?), Nummernzeichen (#), Ausrufezeichen (!), der Bindestrich (-) und eckige Klammern ([ ]) sind Platzhalterzeichen. Sie können diese Zeichen in Abfragen und Ausdrücken verwenden, um alle Datensätze, Dateinamen oder anderen Elemente anzugeben, die mit bestimmten Zeichen beginnen oder mit einem bestimmten Muster übereinstimmen. Sie können Platzhalterzeichen und zu suchende Zeichen auch verwenden, um eine Suche weiter einzuschränken, wenn Sie eine SQL-Anweisung verwenden.

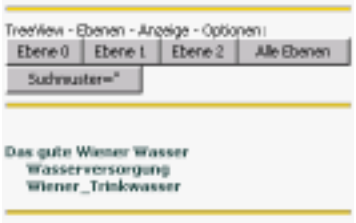
Symbol	Beispiel	Verwendung
*	wa*findet was, wasser und warum *at findet rat, Salat und Saat	Findet eine beliebige Anzahl an Zeichen und kann überall in der Zeichenfolge verwendet werden.
?	b?t findet ball, bell und bill	Findet eine Übereinstimmung mit einem beliebigen Zeichen.
#	1#3 findet 103, 113, 123	Findet eine Übereinstimmung mit einer beliebigen Ziffer.
[]	b[ae]ll findet ball und bell, aber nicht bill	Findet eine Übereinstimmung mit einem der Zeichen in den Klammern.
!	b[!ae]ll findet bill und bull, aber nicht bell oder ball	Findet eine Übereinstimmung mit einem der Zeichen, die nicht in der Liste sind.
-	b[a-c]d findet bad, bbd und bcd	Findet eine Übereinstimmung mit einem Zeichen aus einem Bereich.

Anmerkung Eine Übereinstimmung mit den Platzhalterzeichen * (Sternchen), ? (Fragezeichen), # (Nummernzeichen) und [(öffnende Klammer) selbst kann nur gefunden werden, wenn diese Zeichen in eckige Klammern eingeschlossen werden.

Beispiel

```
<TREEVIEW Type="folder/file"
          Root="/fis/content/wiener wasser"
          Title="Das gute Wiener Wasser"
          Style="list" OpenLevel="2">
   <FILE>
      <IN>
         <PATTERN>*.htm*</PATTERN>
         <PATTERN>*.asp</PATTERN>
         <PATTERN>*.pdf</PATTERN>
      </IN>
      <OUT>
         <PATTERN Case="sensitive">*TEST.*</PATTERN>
      </OUT>
   </FILE>
   <FOLDER>
      <OUT>
         <PATTERN>*files</PATTERN>
         <PATTERN>*images</PATTERN>
      </OUT>
   </FOLDER>
</TREEVIEW>
```

↓ Style="list" OpenLevel="1" ↓



↓ Style="block" OpenLevel="2" ↓



Diese TreeView-Spezifikation hat als Wurzel den Ordner "/fis/content/wiener wasser", an dessen Stelle auf der obersten Ebene des Baumes (i.e. auf Ebene 0) der Text "Das gute Wiener Wasser" angezeigt wird (siehe Abbildungen rechts neben dem obigen Beispiel).

Es werden alle Dateien vom Typ ".htm" und ".html" (eigentlich: alle, deren Typ mit ".htm" beginnt), sowie sämtliche ".asp"- und ".pdf"-Dateien bis zur 2. Ebene angezeigt, außer ihre Dateinamen (nicht der Dateityp!) endet mit der Buchstabenfolge TEST (in Großbuchstaben!).

Ordner, deren Namen mit den (Teil-)Wörtern "files" oder "images" enden, werden nicht angezeigt (und natürlich auch keine Ebenen mehr darunter). Da viele Web-Publishing-Programme Hilfsdateien – z.B. für Bilder oder HTML-Dokumente in Subframes – in solchen Unterordnern ablegen, ist dieses Ausblenden von Teilbäumen ein wichtiges Mittel zur sauberen Veröffentlichung von Dokumenten in einem in sich konsistenten FIS.

Darstellungsteil der FIS-TreeView-Komponente

Entwurfsziel

Baumstrukturen sind dem Benutzer im Zusammenhang mit Computern i.a. bekannt und in der Regel auch vertraut, weil auf solchen Strukturen ein wesentlicher Teil der Benutzerführung fußt. Als Beispiele sei hier nur auf den Datei-Explorer, das Windows-Startmenü und auch auf mehrstufige Menüs in div. Anwendungsprogrammen hingewiesen.

Ziel ist es, diese Vertrautheit im Umgang mit der Metapher "Baum" auf Seiten des Benutzers zu nutzen und ihm ein Navigationswerkzeug anzubieten, welches in der gewohnten visuellen Umsetzung und im gewohnten Verhalten Zutritt zu den Dokumenten des FIS in inhaltlich-hierarchisch strukturierter Form anbietet.

Funktionalität

Dieses Ziel vor Augen, wurden folgende Funktionen abgeleitet:

- Darstellung des Baums als rekursive Liste ("*explorer-like*")
- Öffnen und Schließen von Teilbäumen mittels Mausklick
- Aktionen Auslösen durch Klick auf einen Eintrag (i.e. Blatt) des Baumes
- Öffnen und Schließen des Baumes bis zu einer bestimmten Ebene
- Suchen nach bestimmten Inhalten im Baum

  Anmerkung: im vorliegenden Prototyp kann nur nach Dokument*namen* im *gesamten* Baum gesucht werden (und nicht z.B. nach Textpassagen innerhalb der Dokumente oder nur innerhalb eines Teilbaumes; siehe dazu die Erweiterungsmöglichkeiten auf Seite 37).

Implementierung

Die serverseitige Implementierung erfolgt im vorliegenden Prototyp mit Hilfe einer ASP-Datei (*TreeView.asp*) im Zusammenwirken mit einer Visual-Basic-ActiveX-DLL (*DirTree.dll*) und am Client mit einem *Custom Behavior* (JavaScript-Funktionen in der Datei *TreeView.htc*), welches über eine CSS-Datei (*TreeView.css*) in den Browser geladen wird.

Die Punkte (4) bis (6) im Abschnitt "Entwurfsziel und Funktionalität" im vorangegangenen Kapitel auf Seite 30 sprechen die Art und Weise der visuellen Umsetzung der Baummetapher an der Benutzerschnittstelle an. Die dort geforderte Flexibilität legt die Implementierung mit Hilfe von XSL-Stylesheets nahe (siehe dazu auch Seite 18).

Der Verzeichnisbaum wird daher intern zuerst in eine XML-Zwischenrepräsentation gebracht, welche im Anschluss mit Hilfe von geeigneten XSL-Stylesheets (*Treeview_**block**.xsl* bzw. *Treeview_**list**.xsl* im FIS-System-Ordner */fis/engine/TreeView*) in für den Browser verständlichen HTML-Code transformiert wird.

- Diese Transformation kann grundsätzlich entweder am Client oder am Server durchgeführt werden (zur Entlastung des Servers im vorliegenden Prototyp am Client)
- Die Liste verfügbarer Stile kann infolge des gewählten modularen Konzeptes mit wenig Aufwand erweitert werden, indem zusätzliche XSL-Transformationsdateien erstellt werden (z.B. *Treeview_**table**.xsl* oder *Treeview_**menu**.xsl*)

Für weitere Hinweise siehe den nachfolgenden Abschnitt "Integration der FIS-TreeView-Komponente in eine HTML-Seite", das Handbuch zur FIS-TreeView-Komponente und das FIS-Installationshandbuch.

Integration der FIS-TreeView-Komponente in eine HTML-Seite

Die Integration einer oder mehrerer Instanzen der TreeView-Komponente in eine HTML-Seite erfolgt in drei Schritten:

Schritt 1: Definition des XML-Namespaces "fis" im öffnenden HTML-Tag.

```
<HTML xmlns:fis>
...
</HTML>
```

Schritt 2: Verweis auf die CSS-Datei */fis/css/TreeView.css*, welche ihrerseits auf die für TreeView notwendigen Behavior-Dateien und sonstige Formatangaben verweist.

```
<HEAD>
     ...
     <LINK REL="stylesheet" TYPE="text/css" HREF="/fis/css/TreeView.css">
     ...
</HEAD>
```

Schritt 3: Aufruf der Komponente an einer beliebigen Stelle im HTML-Dokument unter Angabe der Datenquelle (i.e. die FIS-TreeView-Spezifikationsdatei) als src-Attribut

```
<fis:TREEVIEW   Src="...Pfad.../SpezifikationsDatei.xml"
                DefaultStyle="block/list">
</fis:TREEVIEW>
```

Der *Pfad* kann absolut oder relativ angegeben sein oder auch komplett entfallen. Im letzteren Fall muss sich die XML-Spezifikationsdatei im selben Ordner wie die HTML-Datei befinden, aus der sie aufgerufen wird.

Die Angabe das Attributs DefaultStyle ist optional und kann im Prototyp einen der beiden Werte "list" oder "block" annehmen (Defaultwert="list"). Diese Festlegung kommt nur dann zum Tragen, wenn in der TreeView-Spezifikationsdatei (Src-Attribut) keine Angabe des Styles erfolgt.

Das schließende </fis:TREEVIEW>-Tag muss aufgrund der zwingenden Wohlgeformtheit von XML-Tags unbedingt angegeben werden.

Da im Rahmen einer HTML-Seite – im Gegensatz zu XML – bei Element- und Attributnamen nicht zwischen Groß- und Kleinschreibung unterschieden wird, ist diese an dieser Stelle auch nicht relevant. Es wird vom Auftragnehmer jedoch dringend eine diesbezügliche Konvention und damit einhergehend eine einheitliche Schreibweise empfohlen.

Beispiel:

```
<HTML xmlns:fis>
<HEAD>
     ...
     <LINK REL="stylesheet" TYPE="text/css"
           HREF="/fis/css/TableView.css">
     ...
</HEAD>
<BODY>
     ...
     <fis:TREEVIEW Src="myFirstTreee.xml"></fis:TREEVIEW>
     ...
     <fis:TREEVIEW Src="SubFolder/mySecondTree.xml"
                   DefaultStyle="block">
     </fis:TREEVIEW>
     ...
</BODY>
</HTML>
```

Möglichkeiten zur Weiterentwicklung

Die Sprache XML ist aufgrund ihrer rekursiven Ansatzes geradezu maßgeschneidert für die Abbildung von baumartigen Strukturen. Es existieren daher jede Menge Möglichkeiten für Erweiterungen der TreeView-Komponente. Die folgende Liste erhebt keinen Anspruch auf Vollständigkeit, zeigt jedoch die naheliegenden und wichtigsten Optionen auf.

- **Icons für verschiedene Dateigruppen/-typen**

  Windows-Benutzer sind gewohnt, dass verschiedene Dateitypen mit unterschiedlichen bildhaften Symbolen (sog. Icons) symbolisiert werden. Die Angabe könnte in einem eigenen Abschnitt auf derselben Gliederungsebene wie FILE und FOLDER erfolgen, z.B.:

```
<ICONS>
     <ICON src="...">
          <PATTERN>...</PATTERN>
          ....
          <PATTERN>...</PATTERN>
     </ICON>
     ...
     <ICON src="...">
          <PATTERN>...</PATTERN>
          ....
          <PATTERN>...</PATTERN>
     </ICON>
</ICONS>
```

  Das src-Attribut enthält einen Verweis (eine URL) zu einem Icon (z.B. eine gif-Datei) und die darunterliegenden PATTERN regeln, welche Dateien mit dem Icon assoziiert werden sollen.

  Die Reihenfolge der ICON-Elemente ist ausschlaggebend bei der Spezifikation von einander "überlappenden" Pattern.

- **Spezfikation des konkreten Inhalts direkt in der XML-Datei**

  Vorstellbar sind hier "freie" Hierarchien, wo neben der grundlegenden Baumstruktur (TREE / ELEMENT) für die Elemente (Blätter) des Baumes auch die Aktion definiert sein kann, welche durch einen Mausklick ausgelöst wird, z.B.:

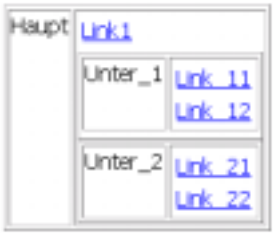
```
<TREE Name="Haupt">
     <ELEMENT Name="Link_1" Href="http://...url..."/>
     <TREE Name="Unter_1">
          <ELEMENT Name="Link_11" Href="... "/>
          <ELEMENT Name="Link_12" Onclick="callMe('Link_21')"/>
     </TREE>
     <TREE Name="Unter_2">
          <ELEMENT Name="Link_21" Onclick="callMe('Link_21')"/>
          <ELEMENT Name="Link_22" Onclick="alert('(c) by Leop')"/>
     </TREE>
</TREE>
...
<SCRIPT Language="javascript">
     function callMe( caller ) {
     ...
     ... // do something with 'caller'
     ...
     }
</SCRIPT>
```

- **Integration zusätzlicher Styles**

  Vor der Integration zusätzlicher, benutzerdefinierter "Styles" muß die intern verwendete XML-Zwischenrepräsentation eingehend überarbeitet und ausführlich dokumentiert werden.

  Als durchaus gängige Stile sind zumindest geschachtelte Tabellen ("table") und mehrstufige Menüs ("menu") vorstellbar, zB:

Style="table"



Style="menu"



- **Andere hierarchische Datenquellen**

  Die Anbindung anderer hierarchisch organisierter Datenquellen, wie z.B.

  - Oracle – "connect-by" – Bäume und
  - ADO – "shaped queries"

  sollte v.a. im Zusammenhang mit der Weiterentwicklung der FIS-TableView-Komponente betrachtet werden.

- **Erweiterte Suchfunktion**

  Das Suchen (eigentlich: das Auffinden) von Information ist eine zentrale Anforderung an jedes Informationssystem.

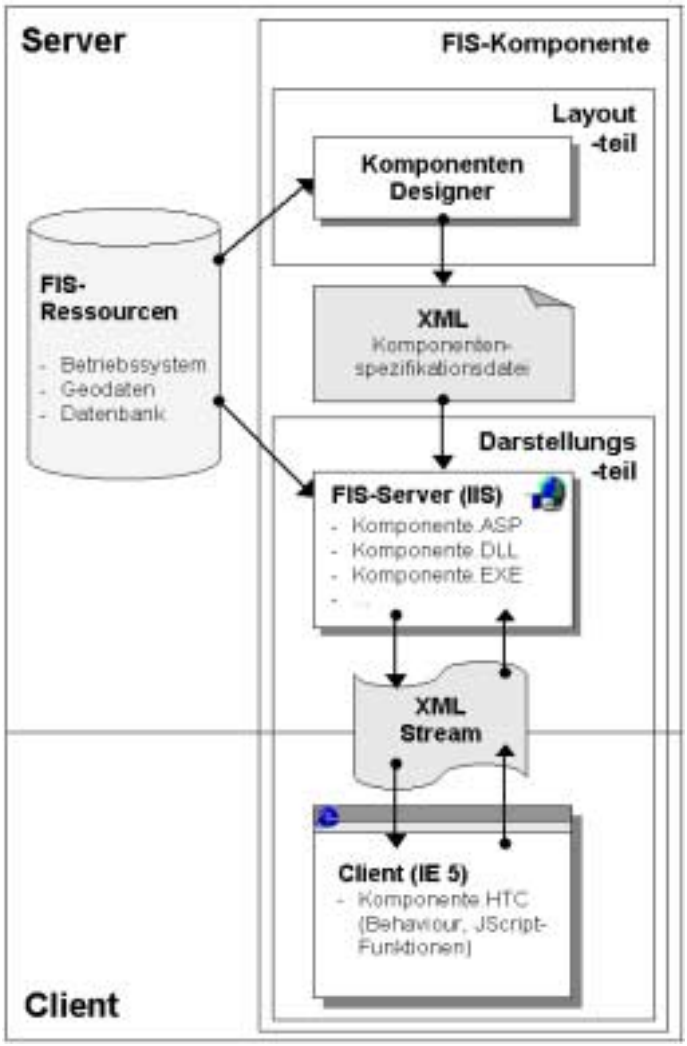
  Eine diesbezügliche Weiterentwicklung des FIS sollte zumindest folgende Operationen unterstützen:

  - Suche auf einen Teilbaum des FIS beschränken
  - Volltextsuche (d.h. auch innerhalb der Dokumente suchen, zumindest aber in den HTML-<META>-Tags)
  - Suche über eine Beschlagwortung von Dokumenten/Inhalten unter Zuhilfenahme von Thesauren und ähnlicher Schlagwortkataloge und -hierarchien

TableView

Zweck und Aufbau

Die FIS-Komponente TableView dient zur Publikation, Visualisierung und Abfrage von alphanumerischen Daten in tabellarischer Form. Sie besteht aus drei Teilen, von denen zwei prototypisch im Rahmen dieses Forschungsauftrags implementiert wurden:



- Layoutteil
  (nicht implementiert)
- serverseitiger Darstellungsteil
  (*TableView.asp*)
- clientseitiger Darstellungsteil
  (*TableView.htc,*
  TabStrip.htc,
  *div. *.htm, *.js und .gif-Dateien*)

Der als ASP-Datei implementierte serverseitige Darstellungsteil hat die Aufgabe der dynamischen Anbindung von Datenbanken via ODBC. Falls die Datensätze bereits direkt in der XML-Spezifikationsdatei enthalten sind, kann dieser Schritt entfallen. Die XML-Datei wird in diesem Fall direkt an den clientseitigen Darstellungsteil übergeben.

Für verschiedene Möglichkeiten der Ein- und Anbindung von Daten siehe vor allem das Kapitel "XML-Tagbeschreibung einer FIS-TableView-Spezifikationsdatei" ab Seite 41.

Layoutteil der FIS-TableView-Komponente

Entwurfsziel und Funktionalität

Wichtigste Rahmenbedingung beim Entwurf der Komponente ist, dass im Hintergrund des FIS Datenbanken stehen, auf die via ODBC zugegriffen wird (z.B. WUIS/Oracle).

Andockstelle des Darstellungsteils der TableView-Komponente an diese Datenquellen ist eine rechteckige Struktur aus Zeilen und Spalten – also eine Struktur, welche im Kontext relationaler Datenbanksysteme üblicherweise als *Table* oder *View* bezeichnet wird.

Der Layoutteil hat somit zwei Aufgaben:

- Das Erstellen der gewünschten tabellarischen Zielstruktur innerhalb des Datenbanksystems (z.B. einer View im WUIS/Oracle) unter Einbeziehung und Zuhilfenahme der vorhandenen Metadaten und -routinen (i.e. die GEORD-Tools der MA14, siehe dazu auch die magistratsinterne Voruntersuchung).
- Das Erzeugen einer FIS-TableView-Spezifikationsdatei auf Grundlage dieser tabellarischen Struktur (Table oder View).

Implementierung

Der Layoutteil der FIS-TableView-Komponente wurde vertragsgemäß im Rahmen dieses Forschungsauftrags nicht implementiert.

Im Rahmen des vorliegenden Prototyps wurde jedoch vom Auftragnehmer eine behelfsmäßige Lösung für die Erstellung einer FIS-TableView-Spezifikationsdatei implementiert:

- Der Layoutteil der MapView-Komponente (i.e. der FIS-MapDesigner, für Details siehe das entsprechende Kapitel in der Prototyp-Dokumentation) bietet die Möglichkeit an, eine in ArcView importierte Tabelle zur Gänze – also mit Struktur *und* Daten! – als XML-TableView-Spezifikationsdatei mit FIELDS- und ROWS-Elementen zu exportieren (für Details dazu siehe den folgenden Abschnitt).
- Falls diese FIS-Tabelle "dynamisiert" werden soll – wenn also die Datensätze erst zur Laufzeit aus der zugrundeliegenden Datenbank (z.B. WUIS) extrahiert werden und nicht fix in der Spezifikationsdatei enthalten sein sollen – muss diese Datei in einen Texteditor geladen und geeignet modifiziert werden.

  Vorgangsweise:

  - Das ROWS-Element und sämtliche darin enthaltenen ROW-Elemente löschen und
  - im TABLE-Element die Attribute *OdbcConn* und *OdbcTable* spezifizieren (für Details dazu siehe ebenfalls den folgenden Abschnitt).

XML-Tagbeschreibung einer FIS-TableView-Spezifikationsdatei

XML-Document-Element (TABLE, TABLES)

In jeder XML-Datei muss genau ein sogenanntes Document-Element (i.e. genau ein Wurzelelement) existieren. Da eine FIS-TableView-Spezifikationsdatei entweder eine oder mehrere Tabellen enthalten kann, sind hier zwei Fälle zu unterscheiden.

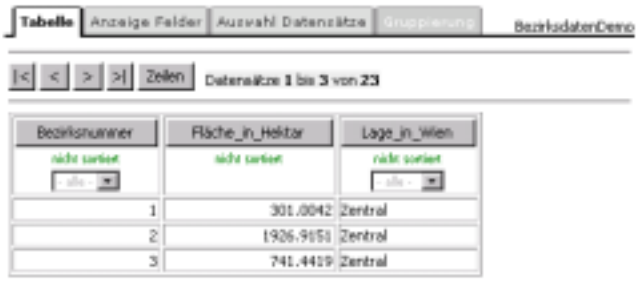
Fall 1: eine Tabelle

```
<TABLE Title="...">
...
</TABLE>
```

Die Angabe des Attributs *Title* ist obligatorisch. Der darin enthaltene Text wird im freien Bereich rechts neben den Titeln der TableView-Registerblätter angezeigt (siehe die folgende Abb. und auch das TableView-Handbuch).

Bsp.:

```
<TABLE Title="BezirksdatenDemo">
...
</TABLE>
```



Fall 2: mehrere Tabellen

```
<TABLES Title="...">
     <TABLE Title="...">
     ...
     </TABLE>
     ...
     <TABLE Title="...">
     ...
     </TABLE>
</TABLES>
```

Die einzelnen Tabellen werden wie im Fall 1 festgelegt. Zusätzlich existiert ein umhüllendes TABLES-Element als DocumentElement. Die Angabe des Attributs *Title* im TABLES-Element ist obligatorisch. Der darin enthaltene Text wird im freien Bereich rechts neben den Titeln der oberen Zeile der TableView-Registerblätter angezeigt, welche ihrerseits die Titel der einzelnen Tabellen enthalten (siehe auch TableView-Dokumentation).

Bsp.:

```
<TABLES Title="2 Tables">
     <TABLE Title="Title #1">
     ...
     </TABLE>
     <TABLE Title="Title #2">
     ...
     </TABLE>
</TABLES>
```



TABLE-Aufbau (FIELDS- und ROWS-Tags)

Eine FIS-Tabelle ist ein rechteckiges Schema von Werten und besteht grundsätzlich aus FIELDS (Felder, Attribute, Spalten) und ROWS (Datensätze, Records, Zeilen).

Eine XML-TableView-Datei kann prinzipiell entweder die Struktur (Feldbeschreibung) *und* die Daten der Tabelle enthalten oder *nur* die Struktur. Es sind daher zwei Fälle zu unterscheiden.

Fall 1: Struktur und Daten

```
<TABLE ... >
     <FIELDS>
          <FIELD ... />
          ...
          <FIELD ... />
     </FIELDS>
     <ROWS>
          <ROW ... />
          ...
          <ROW ... />
     </ROWS>
</TABLE>
```

Die einzelnen FIELD-Tags enthalten Angaben über den Namen und Datentyp der Attributspalten der Tabelle (s.u.) und die ROW-Tags enthalten die Daten selbst.

Fall 2: Nur Struktur ohne Daten

```
<TABLE OdbcConn="..." OdbcTable="...">
     <FIELDS>
          <FIELD ... />
          ...
          <FIELD ... />
     </FIELDS>
</TABLE>
```

In diesem Fall existiert innerhalb des TABLE-Tags kein ROWS-Tag (und somit auch keine ROW-Tags). Es müssen jedoch im TABLE-Tag die beiden Attribute *OdbcConn* und *OdbcTable* angegeben sein

OdbcConn: Name einer gültigen FIS-ODBC-Verbindung, welche in der Datei GLOBAL.ASA (diese befindet im virtuellen Wurzelverzeichnis von FIS) und in der Systemsteuerung über das Programm *ODBC-Datenquellen* als *System-DSN* festgelegt wurde, und zwar wie folgt:

Für jede verwendete FIS-ODBC-Verbindung müssen im vorliegenden Prototyp 3 entsprechende ASP-Application-Variablen in der Datei GLOBAL.ASA initialisiert werden, welche folgender Namenskonvention gehorchen müssen:

ODBC_<connection.name>_CON: Name der ODBC-Verbindung, so wie in der Systemsteuerung festgelegt (i.e. der "echte" ODBC-Name).

ODBC_<connection.name>_UID: UserId, über die auf die Datenbank zugegriffen wird. Wenn die zugrundeliegende Datenbank keine Benutzerrechte verwaltet, dann muß dieser Variable ein Leerstring zugewiesen werden (s.u. Bsp.).

ODBC_<connection.name>_PWD: Passwort, über das auf die Datenbank zugegriffen wird. Wenn die zugrundeliegende Datenbank keine Benutzerrechte verwaltet, dann muß dieser Variable ein Leerstring zugewiesen werden (s.u. Bsp.).

Hier ein beispielhafter Ausschnitt aus GLOBAL.ASA für WUIS_USER (Verbindung zur Oracle-Datenbank) und TUBIB (Verbindung zu MS-Access):

```
<SCRIPT LANGUAGE="VBScript" RUNAT="Server">
...
Sub Application_OnStart
  ...
   Application( "ODBC_WUIS_USER_CON" ) = "fis_user_wuis_oracle"
   Application( "ODBC_WUIS_USER_UID" ) = "WUIS_USER"
   Application( "ODBC_WUIS_USER_PWD" ) = "WUIS_USER"

   Application( "ODBC_TUBIB_CON" ) = "tubib"
   Application( "ODBC_TUBIB_UID" ) = ""
   Application( "ODBC_TUBIB_PWD" ) = ""
...
End Sub
...
</SCRIPT>
```

OdbcTable: Name eines tabellarischen Objekts (i.e. eine Tabelle oder View) in der via OdbcConn spezifizierten Datenbank.

HINWEIS 1: Falls sowohl die beiden Attribute OdbcConn bzw. OdbcTable angegeben sind und auch ein ROWS-Tag innerhalb eines TABLE-Tags, wird in der aktuellen Implementierung die ODBC-Verbindung ignoriert.

HINWEIS 2: Die Definition der Struktur in der XML-TableView-Datei (FIELDS, s.u.) muss mit der Struktur in der Datenbank kompatibel sein (bez. Feldnamen und Datentypen).

Felddefinition (FIELD-Tag)

Die Definition der Datentypen der einzelnen Spalten der Tabelle erfolgt mittels FIELD-Tags. Folgende Attribute können/müssen im FIELD-Tag angegeben sein (Bsp siehe Seite 45):

Attribut-name	**Attributangabe notwendig?**	**Beschreibung**
Name	JA	Name des Attributs in der Tabelle, z.B. "bez"
Alias	JA	Aliasname des Attributs (wird dem Benutzer im TableViewer anstelle des Namens angeboten), z.B. "Bezirk"
Type	JA	"number", "string", "date", "shape" Anmerkung1: der Datentyp "date" ist vorgesehen, wird im Prototyp derzeit jedoch noch nicht unterstützt. Anmerkung2: der Datentyp "shape" wird wie Type= "number" und Precision="0" behandelt (i.e. Ganze Zahlen). Dies erfolgt aus Kompatibilitätsgründen zu MapObjects2, welches bei Attributabfragen auf das "shape"-Feld die RecordNummer des jeweiligen Datensatzes zurückliefert.
Visible	JA	"true"/"false". Regelt die initiale Anzeige im Tabellen-Register der browserseitigen Darstellungskomponente des TableViewers (s. auch TableView-Dokumantation)
Minvalue	nur, wenn keine <OPTION>-Tags definiert sind	kleinster vorkommender Wert in dieser Tabellenspalte

Maxvalue	nur, wenn keine <OPTION>-Tags definiert sind	größter vorkommender Wert in dieser Tabellenspalte
Precision	nur, wenn Type="number"	Anzahl der Nachkommastellen, wenn Type="number". Eine Angabe von 0 oder einer negativen Zahl bedeutet: keine Nachkommastellen
Width	nur, wenn Type="string"	maximal mögliche Länge einer Zeichenkette
Cid	JA	Zeichenkette der Bauart "C<nn>", z.B. "C03" "C<nn>" ist der Attributname im ROW-Tag, welches die tatsächlichen Datenwerte enthält und muss mit einem "C" (Großbuchstabe!) beginnen, gefolgt von einer das Feld eindeutig identifizierenden Zahl. Der Wert des Attributs Cid (=ColumnID) im FIELD-Tag muß mit dem Attributnamen in den ROW-Tags (s.u.) übereinstimmen. Motivation: Cid ist als Abkürzung gedacht, damit bei längeren Attributnamen die Gesamtmenge der über das Netzwerk zu vermittelnden Zeilen gering gehalten wird. Positive Auswirkungen auf das Laufzeitverhalten sollten sich bei sehr langen Tabellen ergeben. Bsp. siehe Ende dieses Abschnitts (Seite 45).

Kategoriale Daten (OPTION-Elemente)

Bei kategorialen Daten (i.e. wenn Type="string" oder Type="number" und "Precision="0" oder neg.), kann (muß aber nicht!) das FIELD-Tag auch noch beliebig viele OPTION-SubTags mit den in der Tabelle vorkommenden (gültigen) Werten enthalten. Diese werden dem Benutzer ggf. zur Auswahl in Form einer Liste angeboten (siehe dazu die TableView-Dokumentation).

Bsp:

```
<FIELD Alias="Lage_in_Wien" Cid="C3" Name="Lage" Type="string"
       Visible="true" Width="16">
     <OPTION>Norden</OPTION>
     <OPTION>Osten</OPTION>
     <OPTION>Süden</OPTION>
     <OPTION>Westen</OPTION>
     <OPTION>Zentral</OPTION>
</FIELD>
```

Wenn die Werte der OPTIONs Sonderzeichen enthalten können (i.e. bei Type="string" die Zeichen: "<", ">" und "&", also die von XML verwendeten Sonderzeichen für die Tag-Struktur und für Entityreferenzen), müssen die OPTIONs in Form von CDATA-Abschnitten angegeben werden. Ein CDATA-Abschnitte sind wie folgt aufgebaut

```
<![CDATA[ ...xxx... ]]>
```

also z.B.

```
<OPTION><![CDATA[01 <AA>]]></OPTION>
```

für den String "01 <AA>" (notwendig hier wegen der beiden Spitzklammern um AA!)

Die eigentlichen Daten (ROW-Elemente)

Ein ROW-Tag enthält die Daten aus genau einem Datensatz der zugrundeliegenden Datenbank. Die Attribute eines ROW-Tags können in beliebiger Reihenfolge stehen.

ACHTUNG: Die **Attributnamen** müssen mit den **Cid**-Attributwerten der korrespondierenden FIELD-Tags übereinstimmen!

ACHTUNG: Jedes ROW-Tag muß **alle Felder** enthalten (i.e. alle Cid-Attributwerte aller FIELD-Tags). Bei NODATA-Werten muß ein Leerstring verwendet werden, z.B. C12=""

Beispiel

Die folgende Tabelle enthält 23 Zeilen (ROWS) und 3 Spalten (FIELDS). Das erste und das dritte Feld ist kategorial mit entsprechenden OPTIONs. Das zweite Feld ist numerisch mit Kommastellen, und enthält daher keine OPTION-Tags. In den ROW-Tags werden die Felder über ihre Cid referenziert (C1 – C3), welche ihrerseits in den FIELD-Tags festgelegt wurden.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<TABLE Title="BezirksdatenDemo">
	<FIELDS>
		<FIELD Alias="Bezirk" Cid="C1" Name="Bez" Precision="0" Type="number" Visible="true">
			<OPTION>1</OPTION>
			<OPTION>2</OPTION>
			<OPTION>3</OPTION>
			<OPTION>4</OPTION>
			<OPTION>5</OPTION>
			<OPTION>6</OPTION>
			<OPTION>7</OPTION>
			<OPTION>8</OPTION>
			<OPTION>9</OPTION>
			<OPTION>10</OPTION>
			<OPTION>11</OPTION>
			<OPTION>12</OPTION>
			<OPTION>13</OPTION>
			<OPTION>14</OPTION>
			<OPTION>15</OPTION>
			<OPTION>16</OPTION>
			<OPTION>17</OPTION>
			<OPTION>18</OPTION>
			<OPTION>19</OPTION>
			<OPTION>20</OPTION>
			<OPTION>21</OPTION>
			<OPTION>22</OPTION>
			<OPTION>23</OPTION>
		</FIELD>
		<FIELD Alias="Fläche_in_Hektar" Cid="C2" Maxvalue="10231.8654" Minvalue="108.3723"
			Name="Area_ha" Precision="4" Type="number" Visible="false"/>
		<FIELD Alias="Lage_in_Wien" Cid="C3" Name="Lage" Type="string" Visible="true" Width="16">
			<OPTION><![CDATA[Norden]]></OPTION>
			<OPTION><![CDATA[Osten]]></OPTION>
			<OPTION><![CDATA[Süden]]></OPTION>
			<OPTION><![CDATA[Westen]]></OPTION>
			<OPTION><![CDATA[Zentral]]></OPTION>
		</FIELD>
	</FIELDS>
	<ROWS>
		<ROW C1="1" C2="301.0042" C3="Zentral"/>
		<ROW C1="2" C2="1926.9151" C3="Zentral"/>
		<ROW C1="3" C2="741.4419" C3="Zentral"/>
		<ROW C1="4" C2="179.5080" C3="Zentral"/>
		<ROW C1="5" C2="203.2971" C3="Zentral"/>
		<ROW C1="6" C2="148.4724" C3="Zentral"/>
		<ROW C1="7" C2="160.9545" C3="Zentral"/>
		<ROW C1="8" C2="108.3723" C3="Zentral"/>
		<ROW C1="9" C2="299.1510" C3="Zentral"/>
		<ROW C1="10" C2="3176.6619" C3="Süden"/>
		<ROW C1="11" C2="2324.3116" C3="Süden"/>
		<ROW C1="12" C2="820.1022" C3="Süden"/>
		<ROW C1="13" C2="3769.8205" C3="Westen"/>
		<ROW C1="14" C2="3399.9318" C3="Westen"/>
		<ROW C1="15" C2="369.1658" C3="Westen"/>
		<ROW C1="16" C2="863.8753" C3="Westen"/>
		<ROW C1="17" C2="1133.9685" C3="Norden"/>
		<ROW C1="18" C2="629.5017" C3="Norden"/>
		<ROW C1="19" C2="2490.2272" C3="Norden"/>
		<ROW C1="20" C2="566.7524" C3="Zentral"/>
		<ROW C1="21" C2="4448.3063" C3="Norden"/>
		<ROW C1="22" C2="10231.8654" C3="Osten"/>
		<ROW C1="23" C2="3201.6874" C3="Süden"/>
	</ROWS>
</TABLE>
```



(↑ Ansicht im Browser)

Darstellungsteil der FIS-TableView-Komponente

Entwurfsziel

Ein großer Teil des Fis-Datenbestandes liegt in Form von Tabellen vor. Die TableView-Komponente erlaubt es diese tabellarische Daten in eine HTML-Seite zu integrierten. Der Benutzer erhält damit Zugriff auf den vom SiteDesigner im FIS publizierten Sachdatenbestand sowie raumbezogener Attributdaten.

Die im Darstellungsteil der FIS-TableView-Komponente gebündelte Funktionalität soll dem Benutzer ein inter-/intranetbasiertes Werkzeug für den möglichst intuitiven Umgang mit diesem Datenbestand in die Hand geben. Die reine Darstellung tabellarischer Daten in Zeilen- und Spaltenform genügt den benutzerseitigen Anforderungen an ein alltagstaugliches Werkzeug insbesondere bei größeren Datenbeständen nicht.

Ein web-basiertes Hilfsmittel kann und muss nicht über die volle Funktionalität eines Tabellenkalkulationssystems verfügen, grundlegende Möglichkeiten der Einflussnahme auf Darstellungsform und Darstellungsinhalte seitens des Benutzers sind aber jedenfalls erforderlich.

Folgende generellen Ziele spielen daher bei der Entwicklung der FIS-TableView-Komponente eine wesentliche Rolle:

- Implementierung einer einheitlichen Benutzeroberfläche für die Tabellendarstellung von Daten aus unterschiedlichen Quellen (z.B. Access, ArcView, Oracle)
- Reduktion der Benutzerschnittstelle auf die notwendigen Elemente
- Zusammenfassen von Funktionalität zu übersichtlichen Darstellungseinheiten innerhalb der HTML-Seite bei gleichzeitiger Nutzung vertrauter Konzepte der Benutzerführung
- einfache Such- und Analysefunktionalität auf Basis bewährter Ansätze
- ressourcenschonende Implementierung durch Verwendung clientseitiger Mechanismen
- Möglichkeit der Identifizierung zusammengehörender Tabellen und Raumobjekte (Integration von TableView und MapView)

Funktionalität

Fis-TableView-TabellenObjekte werden vom Datenanbieter über den Layoutteil der FIS-TableView-Komponente inhaltlich und gestalterisch vordefiniert. Wie bei den MapView-Kartenobjekten handelt es sich dabei um eine "vorbereitete Sicht“ mit denen der Benutzer via Browser interagiert. Die der "Sicht“ zugrunde liegende und vom Server übermittelte Ergebnismenge eines Requests bildet die Basis für benutzerseitige Darstellungs-, Such- und Filteraktionen. Sie selbst wird durch diese Aktionen nicht verändert, d.h. die browserseitigen Gestaltungs- und Analysemöglichkeiten beziehen sich immer auf die gesamte Ergebnismenge.

Bereits die im Prototyp implementierte Funktionalität unterstützt sowohl einfache analytische Arbeitsschritte über die Veränderung von Darstellungsoptionen wie Sortierung, Feldauswahl und Feldreihenfolge, als auch die Verwendung komplexer Filter.

Eigenschaften von TableView-Tabellen

- Tabellenobjekte können Bestandteil jeder HTML-Seite sein
- ein Tabellenobjekt ist eine logische Einheit aus den tabellarischen Daten selbst und den zu deren Manipulation notwendigen Werkzeugen

- die Werkzeuge sind auf Registerkarten zu funktionalen Gruppen zusammengefasst - je Registerblatt eine Gruppe zusammengehöriger Werkzeuge für die Tabellenbearbeitung
- der Benutzer erhält über die Registerkarten einen klar strukturierten Zugriff auf die eigentliche Tabellendarstellung sowie die Darstellungs- und Filterwerkzeuge
- mehrere Tabellenobjekte können Bestandteil einer Tabellengruppe sein
  jedes einzelne Tabellenobjekt ist über eine entsprechende übergeordnete Registerkarte zugänglich, innerhalb der Tabelle stehen jeweils die tabellenspezifischen Werkzeuge über die entsprechenden Tabellen-Registerkarten zur Verfügung
- Attributdaten (TableView-Objekte) und Kartendarstellungen (MapView) sind synchronisiert, d.h. es erfolgt eine optische Kennzeichnung korrespondierender Objekte auf beiden Visualisierungsebenen

Tabellendarstellung

- Tabellenseiten
  Tabellen werden im Darstellungsbereich zu Tabellenseiten zusammengefasst, die Zahl der sinnvoll darstellbaren Tabellenzeilen je Seite ist abhängig von Benutzerpräferenzen und technischen Voraussetzungen (wie Monitorgröße und –auflösung) und steht unter der Kontrolle des Benutzers
- Seitennavigation
  über Befehlsschaltflächen – analog zu anderen Tabellen- und Datenbankprogrammen
- Sortierung
  Steuerung der Sortierreihenfolge von Datensätzen über sequentielle feldweise Sortierung (analog Excel) verbunden mit der Anzeige von Sortierinformationen
- Auswahlbasierte Filterung
  von Datensätzen anhand einfacher Kriterien (analog Excel AutoFilter-Funktion) verbunden mit der Anzeige von Filterinformationen

Feldanzeige und Feldreihenfolge

- Feldanzeige
  Ein- und Ausblenden von – in der Tabellenergebnismenge enthaltenen – Feldern in der Tabellenansicht
- Feldreihenfolge
  individuelles Festlegen der Position eines Feldes in der Tabellenansicht

Datensatzauswahl / komplexe Filterung

i.e. kriterienbasiertes Ausblenden von Datensätzen aus der Ergebnismenge und damit aus der Tabellenansicht. Alle Ausgangsdaten bleiben im Tabellenobjekt enthalten, wodurch nach Veränderungen der Auswahl nicht auf den FIS-Server, sondern lediglich auf den Client-Datenbestand zugegriffen werden muss.

- QBE-Orientierte Oberfläche zur Definition komplexer Auswahlkriterien
  QueryByExample hat sich als (relativ) intuitiv verständliches Konzept zur Abfrage in relationalen Datenbanksystemen wie z.B. MS Access bewährt
- Definition feldbezogener Abfragekriterien durch den Benutzer über ein einfaches "Feld-Vergleichsoperator-Vergleichswert"-Konzept

- einfache logische Verknüpfung mehrerer Kriterien (AND/OR) über die Anordnung von Kriterien in Kriterienzeilen und Kriterienspalten (analog zu MS Access-QBE und Excel-Suchkriterien)

Implementierung

Der serverseitige Datenbankzugriff geschieht im Rahmen eines ASP-Scripts (*TableView.asp*) über eine ODBC-System-DSN und ADO als Basiswerkzeug.

Die Implementierung des clientseitigen Darstellungsteils der FIS-TableView-Komponente erfolgt im Wesentlichen mittels eines *Custom Behavior* (*TableView.htc*), welches in das umgebende HTML-Dokument einen IFRAME mit der Datei *TableView.htm* und eine Referenz zu den abgefragten Datensätzen einbettet.

In *TableView.htm* regelt ein weiteres *Custom Behavior* (*TabStrip.htc*) den Zugriff auf die drei implementierten Ansichten einer FIS-Tabelle über Registerkarten (i.e. Tabellen-Ansicht, Feldauswahl-Ansicht und Datensatzauswahl-Ansicht)

Weitere Informationen befinden sich in der Dokumentation der FIS-TableView-Komponente.

Integration der FIS-TableView-Komponente in eine HTML-Seite

Die Integration einer oder mehrerer Instanzen der TableView-Komponente in eine HTML-Seite erfolgt in drei Schritten:

Schritt 1: Definition des XML-Namespaces "fis" im öffnenden HTML-Tag.

```
<HTML xmlns:fis>
...
</HTML>
```

Schritt 2: Verweis auf die CSS-Datei /fis/css/TableView.css, welche auf die für TableView notwendigen Behavior-Dateien verweisen.

```
<HEAD>
     ...
     <LINK REL="stylesheet" TYPE="text/css" HREF="/fis/css/TableView.css">
     ...
</HEAD>
```

Schritt 3: Aufruf der Komponente an einer beliebigen Stelle des HTML-Dokuments unter Angabe der Datenquelle (i.e. die FIS-TableView-Spezifikationsdatei) als src-Attribut

```
<fis:TABLEVIEW Src="...Pfad.../SpezifikationsDatei.xml"></fis:TABLEVIEW>
```

Der *Pfad* kann absolut oder relativ angegeben sein oder auch komplett entfallen. Im letzteren Fall muss sich die XML-Spezifikationsdatei im selben Ordner wie die HTML-Datei befinden, aus der sie aufgerufen wird.

Das schließende </fis:TABLEVIEW>-Tag muss aus Gründen der Wohlgeformtheit von XML-Tags unbedingt angegeben werden (siehe Seite 13).

Da vom Browser bei der Interpretation einer HTML-Seite – im Gegensatz zu XML – bei Element- und Attributnamen nicht zwischen Groß- und Kleinschreibung unterschieden wird, ist diese an dieser Stelle auch nicht relevant. Es wird vom Auftragnehmer jedoch dringend eine diesbezügliche Konvention und damit einhergehend eine einheitliche Schreibweise empfohlen.

Beispiel:

```
<HTML xmlns:fis>
<HEAD>
     ...
     <LINK REL="stylesheet" TYPE="text/css" HREF="/fis/css/TableView.css">
     ...
</HEAD>
<BODY>
     ...
     <fis:TABLEVIEW Src="myFirstTable.xml"></fis:TABLEVIEW>
     ...
     <fis:TABLEVIEW Src="SubFolder/mySecondTable.xml"></fis:TABLEVIEW>
     ...
</BODY>
</HTML>
```

Möglichkeiten zur Weiterentwicklung

- **breitere Unterstützung von Datentypen und WUIS-Metainformationen**

  wie z.B. Datums- und Zeit-Datentypen, sowie eine stärkere Einbindung der GEORD-Tools der WUIS/Oracle-Datenbank in den Layoutteil der TableView-Komponente.

- **Formate**

  Im Prototyp werden z.B. Zahlen immer rechtsbündig dargestellt, egal wieviele Kommastellen sie enthalten, was zu einer schlechten Lesbarkeit der Tabelle führen kann, wenn die einzelnen Werte verschieden viele Nachkommastellen aufweisen.

  - Angabe von Formatstrings, in der Art wie z.B. in MS-Excel

    Die Spezifikation könnte in der Form eines zusätzlichen Format-Attributs im FIELD-Element erfolgen, z.B.

```
<FIELD Name="aant" Alias="Ausländeranteil" Format="0,00%" ... />
<FIELD Name="bev" Alias="Wohnbevölkerung VZ 91" Format="#.##0" ... />
```

- **Angabe des Skalenniveaus**

  Die explizite Angabe des Skalenniveaus (nominal, ordinal, metrisch) wirkt sich i.a. auf die Zulässigkeit verschiedener statistischer Auswertungs- und Visualisierungsverfahren aus, kann jedoch auch bei der direkten visuellen Umsetzung von Einzelwerten dienlich sein.

  - Umsetzung von metrisch skalierten Variablen in Balken mit "scaled GIFs" oder VML
  - Umsetzung von nominal und ordinal skalierten Variablen in Icons

  Die Spezifikation könnte auch hier in der Form eines zusätzlichen Scale-Attributs im FIELD-Element und eines Icon-Attributs in den darunterliegenden OPTION-Elementen erfolgen, z.B.

```
<FIELD Name="status" Alias="Stand der Dinge" Scale="nominal" ... />
   <OPTION Icon="planung.gif">0</OPTION>
   <OPTION Icon="baustelle.gif">1</OPTION>
   <OPTION Icon="fertig.gif">2</OPTION>
</FIELD>
```

- **Hierarchisierung: Gruppierung, Teil- und Gesamtergebnisse, Kreuztabellen**

  Einfache statistische Auswertungen, wie z.B. das Berechnen einer Summe oder eines Mittelwertes sollten auf jeden Fall vom Benutzer durchgeführt werden können.

  Weitergehende Analysen, wie z.B. das Erstellen einer Kreuztabelle mit Randsummen und –prozentwerten, müssen im Zusammenhang mit einer allfälligen Weiterentwicklung der FIS-TreeView-Komponente und der Angabe des statistischen Skalenniveaus (s.o.) gesehen werden (Eine Gruppierungsvariable muß z.B. nominal oder ordinal sein).

- **Diagramme**

  Diagramme dienen der visuellen Unterstützung der statistischen Auswertung

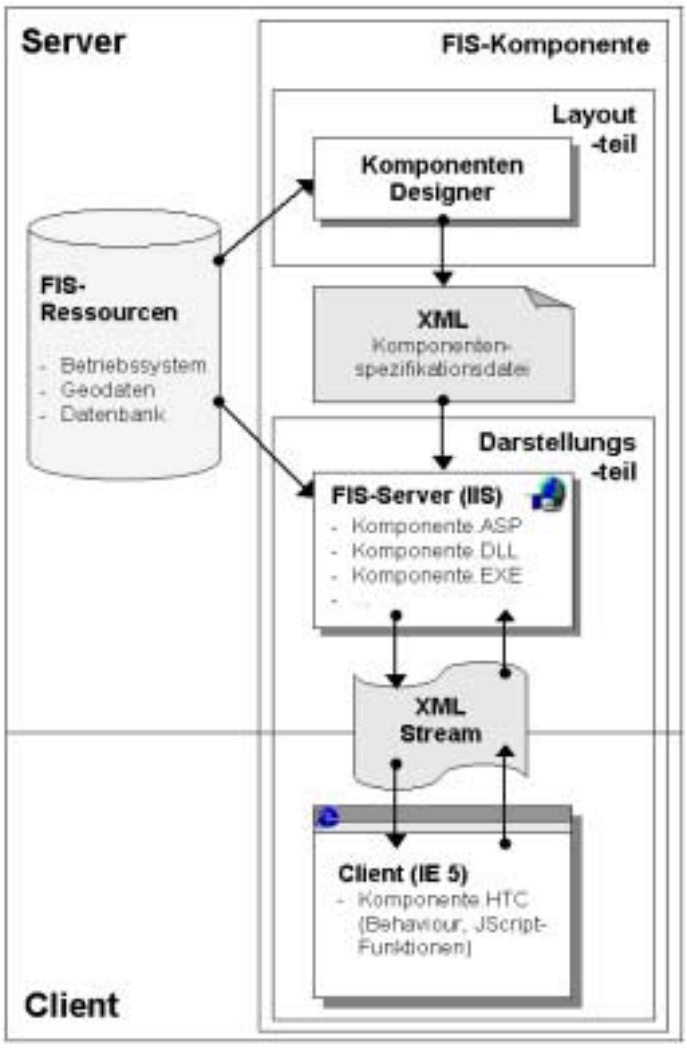
  Als Plattform bietet sich browserseitig VML an, einer Anwendung von XML für Vektorgraphiken von Microsoft (voller Support im IE5), welche dem Internetstandardisierungsgremium W3C zur Begutachtung vorliegt. Siehe dazu http://www.w3.org/TR/NOTE-VML.

MapView

Zweck und Aufbau

Die FIS-Komponente MapView dient zur Publikation, Visualisierung und Abfrage von Daten mit konkretem Raumbezug in Form von interaktiven Internet-Karten.

Die FIS-Komponente MapView besteht aus drei Teilen, von denen im vorliegenden Prototyp in Abwandlung und **Erweiterung des ursprünglichen Auftrags** neben den server- und clientseitigen Darstellungsteilen zusätzlich auch der Layoutteil implementiert wurde (i.e. der FIS-MapDesigner als ArcView-Extension *xml.avx*).



- Layoutteil
  (FIS-MapDesigner
  ArcView-Extension *xml.avx*)
- serverseitiger Darstellungsteil
  (*MapEngine.asp,*
  MapOverview.asp,
  MapObjects 2.0,
  MO2XMLDisplay.exe,
  div. andere dll-Dateien,
  siehe Installationshandbuch)
- clientseitiger Darstellungsteil
  Übersichtsfenster in der Startseite
  und div. andere Kartenfenster
  (*div. HTML- und JavaScript-Dateien in /fis/engine/MapView*)

Abb.: Teile der MapView-Komponente

Die visuelle Umsetzung der XML-Spezifikation und der interaktiven Benutzeranfragen erfolgt im Rahmen der ASP-Scripts *MapEngine.asp* und *MapOverview.asp* mittels der ActiveX-EXE-Datei *MO2XMLDisplay.exe,* die unter Verwendung von *MapObjects 2.0* entsprechende Bilddateien für die Karte und Legende wahlweise im JPG- oder PNG-Format erzeugt.

MapObjects 2.0 ist eine Programmbibliothek und umfasst geographische Klassen und Methoden, welche mit den verwendeten geographischen Datenbeständen in der MA22 und MA14 kompatibel sind und die Entwicklung von geographischen Applikationen mit Visual Basic erlauben. Die Verwendung von MapObjects ist laut Vertrag eine Vorgabe des Auftraggebers.

Anbindung externer Tabellen

Ein Grossteil der Attributdaten für das FIS liegt in Form von Tables der WUIS-Oracle-Datenbank vor. Voraussetzung für die Visualisierung in thematischen Karten ist die Anbindung dieser externen Tabellen an die geographischen Datenschichten über eindeutige Schlüssel. ESRI's MapObjects2 bietet dafür die geeignete Funktionalität über den Befehl *AddRelate* an.

Vor allem bei detaillierten flächendeckenden Daten für Wien (etwa auf Baublockebene, oder für die BIOMON-Erhebungseinheiten) sind im Zuge der Verknüpfung von Geo- und Sachdaten erhebliche Datenmengen zu verarbeiten. In den äußeren Stadtbezirken existieren beispielsweise über 23.000 Erhebungseinheiten in Form von Polygonen.

Problem bei langen Tabellen

Im Zuge der Bearbeitung wurden im Anschluss an die Bereitstellung der raumbezogenen Daten aus BIOMON ausgiebige Leistungstest durchgeführt. Dabei zeigten sich bald deutliche Schwächen von MapObjects bei der Anbindung externer Tabellen.

So benötigte etwa die Verknüpfung im Shapefile der 23.000 BIOMON-Erhebungseinheiten mit der zugehörigen Tabelle über eine Minute. Die Tests wurden mit den folgenden Tabellentypen mittels ODBC-Verbindungen durchgeführt:

- dBase
- Access (97 und 2000)
- Oracle 8i
- Text (Tab-delimited)

Es konnten keine signifikanten Unterschiede hinsichtlich der Verarbeitungsgeschwindigkeit bei den getesteten Tabellentypen festgestellt werden, was auf grundsätzliche Performance-Mängel von MapObjects 2.0 schließen lässt. Unterstützt wird diese These durch den Umstand, dass auch die Verknüpfung in einer ausschließlich für Testzwecke entwickelten Visual-Basic Applikation (unter Ausschaltung von MS-Internet Information Server und MS-Transaction Server) nur unwesentlich schneller erfolgte.

Weiters zeigte eine Internet-Recherche u.a. bei ESRI-Diskussionsforen ganz klar ähnliche (wenn nicht sogar deutlich ernüchterndere) Ergebnisse anderer MapObjects Entwickler, die durch eine Anfrage beim österreichischen Distributor grundsätzlich bestätigt wurde. Auf der vom Auftragnehmer eingesetzten Test-Hardware erfolgte die Verknüpfung von kleineren Datensätzen jedenfalls deutlich schneller als in den überprüften Diskussionsbeiträgen angegeben wurde.

Resümee

Liegt die Zahl der Raumobjekte im aktuellen Ausschnitt bei etwa 500 Polygonen und entsprechend vielen Records aus der externen Tabelle, so erreichen die Antwortzeiten ein gerade noch akzeptables Niveau von etwa 5 Sekunden. Im Gegensatz dazu benötigt MapObjects2 bei Daten, die innerhalb der Attributtabelle vorliegen (d.h. im dbf- bzw. INFO-File der geographischen Daten) – auch bei Datenbeständen in der Größenordnung mehrerer Tausend Datensätze – unter 5 Sekunden für die vollständige Visualisierung.

Soweit es sich zum gegenwärtigen Zeitpunkt abschätzen lässt, liegt die festgestellte Leistungsschwäche außerhalb des Einflussbereiches des Entwicklers (Auftragsnehmer). In MapObjects2 wurden alle dokumentierten Möglichkeiten zur Beschleunigung der Verknüpfung ausgeschöpft (siehe die MapObjects2-Dokumentation zu AddRelate).

Bei dem festgestellten Mangel handelt es sich allerdings um ein Problem, welches einen der Kernbereiche des Projektes betrifft. Ohne die Möglichkeit auch große externe Tabellen sinnvoll anzubinden können wesentliche Projektziele nur über Umwege erreicht werden.

Lösungsansätze

Folgende Lösungsansätze werden vorgeschlagen, wobei der implementierte Prototyp im Darstellungsteil prinzipiell beide Optionen unterstützt:

- ODBC-Anbindung über AddRelate von dynamischen und gleichzeitig relativ kleinen Datenbeständen
- Integration von Attributinformationen großer Datenbestände in das ShapeFile/INFO-File

Aus der Integration der Attributdaten ins Shapefile resultiert vor allem bei veränderlichen Daten beträchtlicher Verwaltungsaufwand für den Datenersteller. Bei jedem Update der tabellarischen Daten ist eine Aktualisierung des dBase- bzw. INFO-Files notwendig. Die beiden wesentlichen Einwände gegen diesen Ansatz sind der zusätzliche Arbeitsaufwand für die Aktualisierung und die Doppelgleisigkeit (Redundanz) bei den Attributinformationen.

Da ein leistungsfähiger Weg der Tabellenverknüpfung in der Basissoftware aktuell nicht in Sicht ist, müssen diese Nachteile bei der Verwendung von MapObjects2 aus heutigem Blickwinkel offenbar in Kauf genommen werden.

Das Laufzeitverhalten von MapObjects2 beim Zugriff auf Daten, welche direkt im Shapefile stehen, ist demgegenüber sehr zufriedenstellend.

Eine **Strategie**, auf welche Art und Weise und in welchen Zyklen eine Aktualisierung der Attributinformation bei der zweiten Option durchgeführt wird, gilt es jedoch noch im Zuge einer allfälligen Weiterentwicklung des FIS gemeinsam mit dem Auftraggeber und der MA14 zu entwickeln.

Layoutteil der FIS-MapView-Komponente (FIS-MapDesigner)

Entwurfsziele

Oberstes Entwurfziel bei der Entwicklung des Layoutteils der MapView-Komponente ist die Forderung nach einer möglichst einfachen und raschen Veröffentlichung von Karten im Intra- bzw. Internet. Der FIS-MapDesigner wurde daher vollständig in die gewohnte Arbeits- und Software-Umgebung der potentiellen Kartenautoren (i.e. der inhaltlichen Sachbearbeiter der MA22) integriert. Da im Magistrat der Stadt Wien vorwiegend Produkte der Firma ESRI verwendet werden (allen voran ArcView und ARC/INFO), erfolgte die Implementierung des Layoutteils als Erweiterungsmodul (*extension*) auf der Basis von ArcView-GIS Version 3.1.

Ziel ist eine weitgehend automatische und verlustfreie Übersetzung von in ArcView generierten kartographischen Ansichten in XML-Kartenspezifikationsdateien, welche im Anschluss vom serverseitigen Darstellungsteil der MapView-Komponente interpretiert und in ein Bild mit den gewünschten Inhalten und dem gewählten Raumausschnitt übersetzt wird, und in der Folge dem browserseitigen Darstellungsteil übergeben wird.

Funktionalität

Die Funktionalität des FIS-MapDesigners wird im vollen Umfang in der FIS-MapDesigner-Dokumentation beschrieben. Hier folgt nur ein kurzer Überblick über die wichtigsten Aspekte und Möglichkeiten. Für die detaillierte Beschreibung der XML-Elemente der MapView-Spezifikationsdatei siehe weiter hinten in diesem Abschnitt ab Seite 56.

Folgende Standard-ArcView-Einstellungen werden automatisch übernommen:

- Der Name einer View als Kartenname
- Die Hintergrundfarbe einer View
- Die Reihenfolge und Namen von Themen (Image- und Vektor-Themen) in einer View
- Die Legenden von Vektor-Themen (Symbolisierung und Labels)
- Die (initiale) Anzeige von Themen
- Die zulässigen Darstellungsmaßstäbe von Themen
- Zu einem Thema gehörige Labels als MapObjects-Labelrenderer
- Der (initial) dargestellte Raumausschnitt (*"area of interest"*)
- Kommentare zu Themen und zur View als sog. ToolTips.
- Die Koordinateneinheit (*map units*)

Der FIS-MapDesigner enthält folgende Erweiterungen der Grundfunktionalität von ArcView

- Verwaltung von benannten Raumausschnitten und deren Übernahme in die XML-Spezifikationsdatei
- Erzeugen von FIS-Imagekatalogen aus mehreren (gekachelten) Images
- Verwaltung von Web-spezifischen Einstellungen, wie zB. themen- und view-spezifische Hyperlinks zu weiteren Internetseiten.

Folgende Restriktionen bestehen bei der Umsetzung von ArcView-Inhalten in die XML-Spezifikation der Web-Karte:

- Themen auf Basis von ARC/INFO-Grids werden nicht übernommen, weil die verwendete Basissoftware im Darstellungsteil (MapObjects 2.0 der Fa. ESRI) diese nur sehr eingeschränkt darstellen kann.
- Der Link zur Datenquelle von mit ODBC angebundenen Datensätzen (ArcView-Befehl *SQL-Connect*) wird nicht in die XML-Spezifikation übernommen. Diese Angabe muß im Prototyp mit Hilfe eines Texteditors von Hand aus in die MapView-XML-Spezifikationsdatei eingegeben werden (RELATE-Element innerhalb eines LAYER-Elements, siehe Seite 56ff). Für eine Betrachtung der grundsätzlichen Probleme bei der Anbindung von externen Datenquellen zu geographischen Daten mit MapObjects2 siehe den vorangehenden Abschnitt (ab Seite 52).
- "Freie" Graphiken in einer View werden im aktuellen Prototyp (noch) nicht übernommen. Der Darstellungsteil von MapView ist jedoch darauf ausgelegt, solche Graphiken im sog. *"tracking layer"* von MapObjects 2.0 darzustellen.
- Da ArcView bezüglich Symbolisierung von Raumelementen (Formatierung der Legende) weit mehr bzw. andere Möglichkeiten anbietet als MapObjects 2.0, erfolgt eine entsprechende Umsetzung unter Verlust von einigen Darstellungsparametern.

  Beispielsweise konnten folgende Punkte nicht oder nur teilweise implementiert werden:

  - skalierbare Symbole von MO2

    die in MO2 implementierte werteabhängige Skalierung von Value-Maps über die Inhalte eines SizeField ist in ArcView nicht möglich

  - frei drehbare Schraffuren in ArcView (bei SymbolKlasse VectorFill)

    MO2 bietet nur eine relativ bescheidene Anzahl von Einstellungsmöglichkeiten für Schraffuren, Winkel sind mit 0°, 45° und 90° vorgegeben – bei der Übersetzung in die XML-Struktur werden die ArcView Einstellungen über Schwellwerte in diese 3 Typen umgewandelt

  - CompositeMarker / CompositeFill / CompositePen von ArcView::

    vergleichbare Elemente fehlen in MO2; im MapDesigner-Export wird das erste (i.e. das unterste) Symbol bzw. ein Default-Symbol in die XML-Struktur übernommen

  - gebrochene/punktierte Liniensymbole („Pattern“) in ArcView

    in MO2 gibt es lediglich 5 vordefinierte LineStyles. MapDesigner nähert die in ArcView verwendeten Linientypen diesen LineStyles bestmöglich an.

Implementierung

Die prototypische Implementierung des Layoutteils der FIS-MapView-Komponente erfolgte in Form einer ArcView-GIS-3.1-Extension mit dem Namen *xml.avx*.

Zur Installation und zur detaillierten Funktionsbeschreibung dieser Extension wird hier auf die diesbezüglichen Abschnitte in der Installationsdokumentation und auf die Dokumentation zum FIS-MapDesigner verwiesen.

XML-Tagbeschreibung einer FIS-MapView-Spezifikationsdatei

Grundsätzliche Struktur

Die generelle Schachtelung von XML-Elementen in einer FIS-MapView-Komponentenspezifikationsdatei stellt sich im groben Überblick wie folgt dar:

```
<MAP>
   <LAYERS>
      <LAYER>
        <RENDERER>...</RENDERER>
      </LAYER>
      ... mehrere <LAYER>-Elemente ...
   </LAYERS>
   <GRAPHICS>
      <GRAPHIC>
        <GEOMETRY>...</GEOMETRY>
        <SYMBOL>...</SYMBOL>
      </GRAPHIC>
      ... mehrere <GRAPHIC>-Elemente ...
   </GRAPHICS>
   <EXTENTS>
      <EXTENT>...</EXTENT>
      ... mehrere <EXTENT>-Elemente ...
   </EXTENTS>
   <EXTENT>...</EXTENT>
   <FULLEXTENT>...</FULLEXTENT>
   <BACKCOLOR ... />
   <COMMENTS>...</COMMENTS>
</MAP>
```

Eine FIS-Karte besteht demnach auf der obersten Ebene aus einem MAP-Element mit

- einer Liste (LAYERS) mit einer oder mehreren Kartenschichten (LAYER) und zugehörigen Legenden (RENDERER),
- einer optionalen Liste (GRAPHICS) von zusätzlichen Graphiken und Texten (GRAPHIC) mit jeweils ihrer Geometrie (GEOMETRY = POINT, LINE, POLYGON, RECTANGLE, OVAL) und Formatierung (SYMBOL),
- einer optionalen Liste (EXTENTS) von benannten Ausschnitten (EXTENT),
- einem standardmäßig darzustellenden Ausschnitt (EXTENT),
- einem optionalen maximal darzustellenden Ausschnitt (FULLEXTENT),
- einer optionalen Farbe des Kartenhintergrundes (BACKCOLOR),
- einem optionalen Kommentar (COMMENTS).

Eine allgemeine Anmerkung:

Sämtliche Angabe von Fließkommazahlen müssen das Zeichen Punkt (".") als Dezimalkomma verwenden (und nicht den in Österreich üblichen Beistrich) !

In weiterer Folge werden die einzelnen XML-Elemente einer FIS-MapView-Karte, sowie ihre Attribute und Subelemente im Detail und umfassend beschrieben.

MAP-Element

Das MAP-Element ist das XML-DocumentElement (Wurzelelement) und dient als Behälter für sämtliche Angaben zur Darstellung einer FIS-Karte.

MAP-Attribute:

Die Attribute eines MAP-Elements legen im Wesentlichen das Ergebnis der Interpretation dieser XML-Datei fest, nämlich die binäre Bilddatei mit dem Bild (*image*) der Karte.

Attribut-name	Angabe notwendig?	Beschreibung
Name	JA*)	Der Basisname der Bilddatei (d.h. ohne Pfad und ohne Dateierweiterung).
Path	JA*)	Der Ordner (Pfad), in dem die Bilddatei gespeichert werden soll.
Height	JA*)	Die Höhe des Bildes in Pixel bei der ersten Anzeige.
Width	JA*)	Die Breite des Bildes in Pixel bei der ersten Anzeige.
Format	NEIN*)	Das Dateiformat der Bilddatei bei der ersten Anzeige: "JPG<n>" mit <n> aus 0 bis 100 ganzzahlig (0=schlechteste Qualität, 100=beste Qualität) "PNG<n>" mit <n> aus { 8, 24 } (Farbtiefe) Defaultwert ist "JPG85".
Unit	NEIN	Maßeinheit der Koordinaten: "m", "inch", "km", "ft". Defaultwert ist "m" (Meter). Wird für die Maßstabsabhängige Darstellung von Kartenschichten und für die Maßstabsleiste in der Karte benötigt
Alias	JA	Name der Karte, wie er in der Legende der Karte im linken Browserframe (TOC, *table of content*) dargestellt werden soll.
Url	NEIN	Optionale URL zu einem (HTML-)Dokument mit weiteren Informationen zur Karte. Muss absolut (im Ggs. zu *relativ*) angegeben werden und, falls außerhalb des FIS-Baumes, unter Angabe des verwendeten Internetprotokolls (meist "http://")
Draw	NEIN*)	"true" / "false". Defaultwert ist "true" Regelt, ob eine Bilddatei erzeugt werden soll. Wird automatisch bei einer Info-Abfrage auf "false" gesetzt, sonst auf "true".

*) Wird automatisch, oder ggf. gemäß Benutzereingabe, vom serverseitigen Darstellungsteil *MapEngine.asp* gesetzt.

MAP-Subelemente:

Elementname	Angabe notwendig?	Beschreibung
BACKCOLOR	NEIN	Hintergrundfarbe der Karte. Defaultfarbe ist Schwarz. Zur Syntax siehe das COLOR-Element auf Seite 61.
COMMENTS	NEIN	Beliebiger Kommentar zur Karte. Wird, falls angegeben, im Browser als Tool-Tip über dem Namen der Karte im Legendenbereich angezeigt. Falls der Kommentar XML-spezifische Sonderzeichen enthält ("<", ">", "&"), muss er zusätzlich in einer CDATA-Section eingebettet werden, z.B.: `<COMMENTS><![CDATA[ Eine Karte mit Quellenangabe in Spitzklammern <Quelle: "xxx"> ]]></COMMENTS>`
EXTENT	JA	Dargestellter Kartenausschnitt bei der ersten Anzeige der Karte. (automatischer Name "Standard" in der PullDown-Liste der benannten Ausschnitte im Browser; siehe MapView-Dokumentation) Zur Syntax siehe das RECTANGLE-Element auf Seite 59f.
EXTENTS	NEIN	Optionaler Behälter für eine untergeordnete Liste von EXTENT-Elementen. Diese EXTENT-Elemente müssen ein Attribut Name="...xxx..." enthalten. Diese Namen werden in der PullDown-Liste der benannten Auschnitte im Browser angezeigt (siehe MapView-Dokumentation). Die Liste der benannten Ausschnitte wird in der hier spezifizierten Reihenfolge angezeigt.
FULLEXTENT	NEIN	Maximal darstellbarer Kartenausschnitt. Falls nicht angegeben, kann jeder Ausschnitt dargestellt werden (genauer: es kann beliebig weit herausgezoomt werden). Der Fullextent wird beim Entwurf mit dem MapDesigner automatisch auf den vollen View-Extent plus 10% Rand gesetzt. Zur Syntax siehe das RECTANGLE-Element auf Seite 59f.
GRAPHICS	NEIN	Optionaler Behälter für eine untergeordnete Liste von GRAPHIC-Elementen (Syntax siehe Seite 59ff). Die einzelnen GRAPHIC-Elemente werden in der hier festgelegten Reihenfolge gezeichnet.
LAYERS	JA	Behälter für eine **mindestens einelementige** untergeordnete Liste von LAYER-Elementen (Syntax siehe Seite 67ff). Die einzelnen LAYER-Elemente werden **umgekehrt** zur hier spezifizierten Reihenfolge gezeichnet (d.h. "oberstes zuletzt").

GRAPHIC-Element

Eine Graphik besteht aus

- einem geometrischen Primitiv (POINT, LINE, POLYGON, RECTANGLE, OVAL oder TEXT) und
- einer Zeichen-(i.e. Formatierungs-)Anweisung (SYMBOL, TEXTSYMBOL)

in genau dieser Reihenfolge (d.h. zuerst die Geometrie, dann das zugehörige Symbol!).

Bsp. Ein rotes Dreieck mit einer 3 Punkte dicken gelben Umrandung (Details zu den einzelnen Elementen siehe weiter unten):

```
<GRAPHIC>
   <POLYGON>
      <POINT X="0" Y="0" />
      <POINT X="0" Y="10" />
      <POINT X="10" Y="10" />
   </POLYGON>
   <SYMBOL>
      <STYLE>0</STYLE>
      <SIZE>3</SIZE>
      <COLOR Red="255" />
      <OUTLINECOLOR Red="255" Green="255" />
   </SYMBOL>
</GRAPHIC>
```

GRAPHIC – geometrische Subelemente:

Element bzw. Attributname	Angabe notwendig?	Beschreibung
POINT		Punkt mit xy-Koordinaten in MAP-Units. Bsp.: `<POINT X="17.5" Y="-22.9" />`
X	JA	X-Koordinate des Punktes (Units siehe MAP-Element)
Y	JA	Y-Koordinate des Punktes (Units siehe MAP-Element)

LINE		Polylinie mit beliebig vielen Stützpunkten. Bsp.:
		`<LINE>`
		`<POINT X="0" Y="0" />`
		`<POINT X="0" Y="10" />`
		`<POINT X="10" Y="10" />`
		`</LINE>`

POLYGON		Geschlossenes Polygon mit beliebig vielen Stützpunkten. Der letzte Punkt wird automatisch mit dem ersten Punkt verbunden. Bsp.:
		`<POLYGON>`
		`<POINT X="0" Y="0" />`
		`<POINT X="0" Y="10" />`
		`<POINT X="10" Y="10" />`
		`</POLYGON>`

RECTANGLE	achsenparalleles Rechteck.
	Die Angabe erfolgt über vier Subelemente mit den Randkoordinaten.
	Bsp.:
	`<RECTANGLE>`
	`<LEFT>1000</LEFT>`
	`<RIGHT>2000</ RIGHT >`
	`<BOTTOM>1000</BOTTOM>`
	`<TOP>2000</TOP>`
	`</RECTANGLE>`
	Dieselbe Syntax wird auch für die XML-Elemente EXTENT und FULLEXTENT verwendet.

OVAL	achsenparallele Ellipse.
	Die Angabe erfolgt über vier Subelemente mit den Randkoordinaten. Bsp.:
	`<OVAL>`
	`<LEFT>1000</LEFT>`
	`<RIGHT>2000</ RIGHT >`
	`<BOTTOM>1000</BOTTOM>`
	`<TOP>2000</TOP>`
	`</OVAL>`

TEXT	freier Text.
	Ein Text wird über Subelemente mit einen Referenzpunkt (POINT) und einem Textwert (VALUE) festgelegt. Beide Angaben sind notwendig.
	Bsp.:
	`<TEXT>`
	`<POINT X="12.3" Y="-45.6" />`
	`<VALUE>Ich bin ein Text</VALUE>`
	`</TEXT>`
	Falls der Text XML-spezifische Sonderzeichen enthält ("<", ">", "&"), muss er zusätzlich in einer CDATA-Section eingebettet werden, z.B.:
	`<TEXT>`
	`<POINT X="12.3" Y="-45.6" />`
	`<VALUE>`
	`<![CDATA[ Ich habe & und <Spitzklammern> ]]>`
	`</VALUE>`
	`</TEXT>`
	ACHTUNG: das nachfolgende Symbol muss ein TEXTSYMBOL sein und kein SYMBOL (siehe Seite 65) !

Unmittelbar nach der Angabe der Geometrie erfolgt eine Anweisung, wie die Figur gezeichnet werden soll (i.e. die Spezifikation eines Formates). Da die Art und Weise der Interpretation und die Gültigkeit der Formatangaben vom Geometrietyp abhängig ist, erfolgt die nachstehende Beschreibung der Symboltypen gruppiert nach punktförmigen, linienhaften und flächigen Symbolen (SYMBOL), sowie für Textsymbole (TEXTSYMBOL).

COLOR - Element

Farben werden generell über ihre Rot-, Grün- und Blau-Werte angegeben

Element bzw. Attributname	Angabe notwendig?	Beschreibung
COLOR		Farbangabe im RGB-System
Red	NEIN	Rotwert auf einer Skala von 0 bis 255. Default ist 0.
Green	NEIN	Grünwert auf einer Skala von 0 bis 255. Default ist 0.
Blue	NEIN	Blauwert auf einer Skala von 0 bis 255. Default ist 0.

Die selbe Syntax gilt auch für die XML-Elemente OUTLINECOLOR, STARTCOLOR und ENDCOLOR.

Bsp.

Die Farben rot, gelb und hellgelb:

```
<COLOR Red="255" />
<COLOR Red="255" Green="255" Green="0" />
<COLOR Red="255" Green="255" Green="120" />
```

SYMBOL-Element

GRAPHIC – SYMBOL – Subelemente für Punkte:

Elementname	Angabe notwendig?	Beschreibung
COLOR	NEIN	Textfarbe. Defaultfarbe ist Schwarz. Zur Syntax siehe das COLOR-Element auf Seite 61.
STYLE	JA	Eine Zahl zwischen 0 und 4 Angabe des Markerstils (siehe auch die MapObjects-2.0-Dokumentation) 0 Kreis 1 Quadrat 2 gleichseitiges Dreieck 3 Kreuz 4 TrueType-Font: die Angabe des Markers erfolgt im FONT- und CHARACTERINDEX-Element
SIZE	JA	Größe des Punktsymbols in Punkten
FONT	nur bei STYLE 4	Name eines am Server installierten Font. Für eine Liste der von der Fa. ESRI mit MapObjects-2.0 und ArcView-GIS-3.x mitgelieferten Fonts siehe die jeweilige Dokumentation und/oder Online-Hilfe.
CHARACTERINDEX	nur bei STYLE 4	Indexnummer des gewünschten "Buchstaben" im angegebenen Font.
ROTATION	NEIN	Rotationswinkel in Grad des Punktsymbols im Gegenuhrzeigersinn. Defaultwert ist 0.

Bsp:

ein dunkelgrauer, um 45° gegen den Uhrzeigersinn gedrehter, 50 Punkte großer Nordpfeil (siehe MapObjects-2.0-Dokumentation)

```
<GRAPHIC>
   <POINT X="1234.5" Y="6789.0" />
   <SYMBOL>
      <STYLE>4</STYLE>
      <SIZE>50</SIZE>
      <FONT>ESRI Cartography</FONT>
      <CHARACTERINDEX>47</CHARACTERINDEX>
      <ROTATION>45</ROTATION>
      <COLOR Red="50" Green="50" Blue="50" />
   </SYMBOL>
</GRAPHIC>
```

GRAPHIC – SYMBOL – Subelemente für Linien:

Elementname	Angabe notwendig?	Beschreibung
COLOR	NEIN	Linienfarbe. Defaultfarbe ist Schwarz. Zur Syntax siehe das COLOR-Element auf Seite 61.
STYLE	JA	Eine Zahl zwischen 0 und 4 Angabe des Linienrstils (siehe auch die MapObjects-2.0-Dokumentation) 0 durchgezogene Linie 1 strichlierte Linie 2 punktierte Linie 3 strich-punktierte Linie 4 strich-punkt-punktierte Linie
SIZE	JA	Breite der Linie in Punkten

Bsp:

eine hellgraue, 0.5 Punkt dünne, punktierte Line in Form eines "L"

```
<GRAPHIC>
   <LINE>
      <POINT X="0" Y="20" />
      <POINT X="0" Y="0" />
      <POINT X="10" Y="0" />
   </LINE>
   <SYMBOL>
      <COLOR Red="200" Blue="200" Green="200" />
      <STYLE>2</STYLE>
      <SIZE>0.5</SIZE>
   <SYMBOL>
</GRAPHIC>
```

GRAPHIC – SYMBOL – Subelemente für Flächen (Polygone, Rechtecke, Ellipsen):

Elementname	Angabe notwendig?	Beschreibung
COLOR	NEIN	Füllfarbe. Defaultfarbe ist Schwarz. Zur Syntax siehe das COLOR-Element auf Seite 61.
STYLE	JA	Eine Zahl zwischen 0 und 10 Angabe des Füllstils (*fill style*, siehe auch die MapObjects-2.0-Dokumentation) 0 vollflächig 1 transparent 2 horizontal schraffiert 3 vertikal schraffiert 4 diagonal schraffiert (links oben nach rechts unten) 5 diagonal schraffiert (links unten nach rechts oben) 6 Kreuzschraffur (achsenparallel) 7 Kreuzschraffur (diagonal) 8 punktiert (hell) 9 punktiert (mittel) 10 punktiert (dunkel)
OUTLINE	NEIN	true/false. Regelt, ob eine Umrandungslinie gezeichnet werden soll. Defaultwert ist true.
OUTLINECOLOR	NEIN	Farbe der Umrandungslinie. Default ist Schwarz.
SIZE	NEIN	Breite der Umrandungslinie in Punkten

Bsp:

ein hellrot ausgefülltes Dreieck mit einem dunkelblauen, 2 Punkte dicken Rand

```
<GRAPHIC>
   <POLYGON>
      <POINT X="0" Y="20" />
      <POINT X="0" Y="0" />
      <POINT X="10" Y="0" />
   </POLYGON>
   <SYMBOL>
      <COLOR Red="255" Green="200" Blue="200" />
      <STYLE>0</STYLE>
      <OUTLINE>true</OUTLINE>
      <OUTLINECOLOR Blue="150" />
      <SIZE>2</SIZE>
   </SYMBOL>
</GRAPHIC>
```

TEXTSYMBOL-Element

Subelemente für TEXTSYMBOL:

Elementname	Angabe notwendig?	Beschreibung
COLOR	NEIN	Textfarbe. Defaultfarbe ist Schwarz. Zur Syntax siehe das COLOR-Element auf Seite 61.
BOLD	NEIN	true / false. Default ist false. Gibt an, ob der Font in fetter Schrift verwendet wird.
ITALIC	NEIN	true / false. Default ist false. Gibt an, ob der Font in kursiver Schrift verwendet wird.
STRIKETHROUGH	NEIN	true / false. Default ist false. Gibt an, ob der Font durchgestrichen verwendet wird.
UNDERLINE	NEIN	true / false. Default ist false. Gibt an, ob der Font unterstrichen verwendet wird.
FITTED	NEIN	true / false. Default ist false. Gibt an, ob der Text zwischen zwei Punkten einer Linie eingepasst werden soll. Hat nur Auswirkungen, wenn das TEXTSYMBOL in einem LabelRenderer mit einem Linienthema verwendet wird (siehe Seite 71ff)
FONT	JA	Name eines am Server installierten Font. Für eine Liste der von der Fa. ESRI mit MapObjects-2.0 und ArcView-GIS-3.x mitgelieferten Fonts siehe die jeweilige Dokumentation und/oder Online-Hilfe.
SIZE	nur, wenn HEIGHT nicht angegeben	Größe der Schrift in Punkten Die Schriftgröße wird beim Zoomen nicht mitskaliert.
HEIGHT	nur, wenn SIZE nicht angegeben	Größe der Schrift in MapUnits Die Schriftgröße wird beim Zoomen mitskaliert.
ROTATION	NEIN	Rotationswinkel der Schrift in Grad im Gegenuhrzeigersinn. Default ist 0 (keine Rotation)
HALIGN	NEIN	Zahl 3, 4 oder 5. Default ist 5 Horizontale Lage der Schrift bezogen auf den Referenzpunkt (siehe MapObjects-2.0-Dokumentation zu Alignment-Konstanten). 3 Referenzpunkt liegt links neben der Schrift 4 Referenzpunkt liegt rechts neben der Schrift 5 Referenzpunkt liegt zentriert bez. der Schrift

VALIGN	NEIN	Zahl 1, 2, 5 oder 6. Default ist 5 Vertikale Lage der Schrift bezogen auf den Referenzpunkt (siehe MapObjects-2.0-Dokumentation zu Alignment-Konstanten). 1 Referenzpunkt liegt über der Schrift 2 Referenzpunkt liegt unter der Schrift 5 Referenzpunkt liegt zentriert bez. der Schrift 6 Referenzpunkt liegt auf der Basislinie der Schrift

Bsp.:

Ein fetter, kursiver, roter Text in Arial 20pt rechts unter dem Referenzpunkt (0,20)

```
<GRAPHIC>
   <TEXT>
      <POINT X="0" Y="20" />
      <VALUE>Ein fetter, kursiver, roter Text in Arial 20pt
             rechts unter dem Referenzpunkt (0,20)
      </VALUE>
   </TEXT>
   <TEXTSYMBOL>
      <COLOR Red="255" />
      <FONT>Arial</FONT>
      <BOLD>true</BOLD>
      <ITALIC>true</ITALIC>
      <SIZE>20</SIZE>
      <HALIGN>3</HALIGN>
      <VALIGN>1</VALIGN>
   </TEXTSYMBOL>
</GRAPHIC>
```

LAYER-Element

Ein LAYER steht für eine geographische Datenschicht.

Das LAYER-Element besteht aus Attributen, welche u.a. Angaben über Typ und Speicherstruktur der Datenschicht enthalten.

Das LAYER-Element kann aber auch selbst Subelemente enthalten, welche v.a. über die Klassifizierung und Symbolisierung in der Legende und die Struktur der Attributtabelle Auskunft geben (z.B. RENDERER, siehe Seite 71ff).

LAYER-Attribute:

Attributname	Angabe notwendig?	Beschreibung
ID	JA	Eindeutige numerische ID des Layers innerhalb der Karte (MAP)
Alias	JA	Name der Datenschicht, wie er im linken Browserframe (TOC, *table of content,* Legende) dargestellt werden soll.
Name	JA	Name der Datenschicht.
LayerType	JA	"map", "cover", "image". Typ der Datenschicht: "map" ArcView-Shapefile "cover" ArcINFO-Coverage "image" Ein von Mapobjects-2.0 unterstütztes Bilddatenformat (Z.B. TIFF, SUN, GIF, BMP, ERDAS, etc. ;siehe Mapobjects-2.0-Dokumentation)
DataConn	JA	Der Ordner (Pfad), wo die Datenschicht gespeichert ist inklusive Laufwerksbezeichnung.
GeoDataset	JA	Dateiname der Datenschicht ACHTUNG: mit Dateierweiterung bei Image-Dateien, ohne Dateierweiterung (.shp) bei Shapefiles
FeatureType	nur bei LayerType = "map" oder "cover"	"point", "line", "polygon", "imagecatalog". Art der geometrischen Primitive in der Datenschicht (Shapefile oder Coverage): "point" Punkte "line" Linien "polygon" Polygone "imagecatalog" Es liegt ein FIS-Imagecatalog in Form eines Shapefiles (oder einer Coverage) vor. Für eine ausführliche Beschreibung siehe die FIS-MapDesigner-Dokumentation. In diesem Fall muss das Attribut *SourceField* angegeben sein!

SourceField	nur bei FeatureType= "imagecatalog"	Name des Text-Attributes im Shapefile/Coverage mit den Dateinamen der einzelnen Image-Dateien des Katalogs. Das Feld wird bei der Verwendung des FIS-MapDesigners automatisch "Filename" genannt.
Visible	JA	"true" / "false". Regelt, ob der Layer bei der ersten Anzeige im Browser automatisch gezeichnet werden soll oder nicht.
LegendVisible	NEIN	"true" / "false". Der Defaultwert wird vom Attribut *Visible* übernommen. Regelt, ob die Legende des Layers bei der ersten Anzeige im Browser automatisch angezeigt werden soll oder nicht.
DrawAlways	NEIN	"true" / "false". Defaultwert ist "false". Wenn "true", wird dieser Layer immer gezeichnet und kann vom Benutzer im Browser nicht ausgeschaltet werden.
MinScale	JA	"<positive zahl>".
MaxScale	JA	Die Kartenschicht wird nur dann angezeigt; wenn der aktuelle Kartenmaßstab zwischen *MinScale* und *MaxScale* liegt. ACHTUNG 1: Der effektive Maßstab hängt von der Auflösung des Monitors am Server (!!) ab, da ja dort die Kartenberechnung stattfindet. Es ist daher zu empfehlen, am Server einen (für die meisten Benutzer) "typischen" Monitor mit einer "typischen" Bildschirmauflösung zu verwenden. ACHTUNG 2: MinScale muß immer größer als MaxScale sein, weil ja eigentlich der Kehrwert verwendet wird (europäische Logik im Ggs. zum amerikanischen Gebrauch) Bsp.: Bei *MinScale="10000"* und *MaxScale="2000"* wird der Layer nur dann angezeigt, wenn der Maßstab zwischen 1:2.000 (i.e. *MaxScale*, der größtmögliche Maßstab) und 1:10.000 (i.e. *MinScale*, der kleinstmögliche Maßstab) liegt.
Url	NEIN	Optionale URL zu einem (HTML-)Dokument mit weiteren Informationen zur Datenschicht (z.B. mit einer genauen Beschreibung der Klassen und Symbole). Falls eine Url angegeben ist, wird der Name des Layers in der Legende (*Alias*) als A-Verweisanker formatiert. Muss absolut (im Ggs. zu *relativ*) angegeben werden und, falls außerhalb des FIS-Baumes, unter Angabe des verwendeten Internetprotokolls (meist "http://")

Bsp:

```
<LAYER Alias="Wiener Stadtgebiet" DataConn="E:\fis_gis\basis" DrawAlways="true"
       FeatureClass="Polygon" GeoDataset="wien" Id="5" LayerType="Map"
       MaxScale="0" MinScale="100000000" Name="wien.shp"
       Url="/fis/content/wien/Übersicht.htm" Visible="true">
...
</LAYER>
```

LAYER – Subelemente

Bei (Vektor-) Datenschichten mit LayerType="map" (Shapefile) oder LayerType="cover" (Coverage) können innerhalb des LAYER-Elementes weitere Subelemente angegeben sein:

Element-name	Angabe notwendig?	Beschreibung
RELATE	NEIN	Dient zum Einbinden von Attributinformationen aus einer externen Datenquelle via ODBC über eine gemeinsame Attributspalte (s.u.)
FIELDS	NEIN	Dient zur Beschreibung der Felder (Attribute, Spalten) der zu dieser Datenschicht gehörigen Attributtabelle. Für genaue Informationen zur Syntax der darunterliegenden FIELD-Subelemente sei hier auf Seite 43 der TableView-XML-Spezifikation verwiesen. Es werden nur diejenigen Felder in die interaktive INFO-Abfrage aufgenommen, welche an dieser Stelle explizit angegeben sind. HINWEIS: Wenn kein FIELDS-Element angegeben ist, dann ist auch keine interaktive INFO-Abfrage für diese Datenschicht möglich, und im Browser wird das entsprechende Kontroll-kästchen nicht angeboten, d.h. die FIELDS-Angabe steuert die Möglichkeit zur interaktiven INFO-Abfrage.

FILTER	NEIN	Bietet die Möglichkeit zur Angabe von 2 Arten von Filtern mit Hilfe einer SQL-Bedingung in einer CDATA-Section:
		Type="definition" Vorselektion von Datensätzen (entspricht dem ArcView-Befehl *Theme / Properties / Definition*). Diese Auswahl wird immer durchgeführt (Texte müssen SQL-konfom zw. einfachen Hochkammas ' stehen!)
		`<FILTER Type="definition"><![CDATA[ Bezirk<20 ]]>`
		`<FILTER Type="definition"><![CDATA[ Typ='Wiese' ]]>`
		Type="user" Benutzerdefinierte Selektion von Datensätzen.
		HINWEIS: Die Angabe eines Filters mit Type="user" dient nur zur Steuerung, ob der Benutzer Datensätze interaktiv über eine Bedingung Datensätze auswählen darf. Eine ev. hier in der XMM-Datei spezifizierte Bedingung wird im Prototyp ignoriert
		`<FILTER Type="user"><![CDATA[]]>`

SYMBOL	nur wenn kein RENDERER definiert ist	Das Symbol, mit dem sämtliche Raumobjekte gezeichnet werden, wenn kein Renderer angegeben ist (entspricht einer SingleSymbol-Legende). SYMBOL-Details siehe Seite 62ff.
RENDERER	NEIN	MapObjects-2.0-Renderer, mit welchem die Raumobjekte einzelnen Klassen und damit Symbolen zugeordnet werden (Details siehe Seite 71ff).

RELATE-Element

Mit Hilfe des RELATE-Elementes kann eine externe Datenquelle an die Attributtabelle der Datenschicht über gleiche Werte eines "gemeinsamen" Attributfeldes gebunden werden (i.e. ein sog. relationaler Join).

RELATE-Attribute:

Attributname	Angabe notwendig?	Beschreibung
Database	JA	Name der Datenquelle. Muß über die Windows-Systemsteuerung mit dem Programm "ODBC-Datenquellen" als System-DSN definiert werden.
User	JA	Username
Password	JA	Passwort
Tablename	JA	Name einer Tabelle (oder View) in der externen Datenbank
FromField	JA	Name des "gemeinsamen" Feldes in der externen Datei. Muss den selben Datentyp haben wie das ToField.
ToField	JA	Name des "gemeinsamen" Feldes in der Attributdatei der Datenschicht. Muss den selben Datentyp haben wie das FromField.
CheckFields	NEIN	"true" / "false". Defaultwert ist "true". Es wird von MapObjects-2.0 überprüft, ob ein Index oder die Generierung eines Indexes für eine schnellere Verknüpfung möglich ist.

Wann genau ein RELATE erfolgreich ist, kann im Detail der MapObjects-2.0-Dokumentation zur Methode *AddRelate* entnommen werden.

Weiters sei hier auf die im Laufe der Untersuchung offenbar gewordenen, die Zugriffs-geschwindigkeit betreffenden, technischen Probleme im Kapitel "Anbindung externer Tabellen" hingewiesen (siehe Seite 52).

Bsp:

```
<RELATE Database="WUIS_Oracle_Treiber"
        User="WUIS_USER"
        Password="WUIS_USER"
        Tablename="WIEN_PARKEN_1996"
        FromField="RBW_CODE"
        ToField="Beznr" />
```

RENDERER-Element

Ein Renderer übersetzt – ganz allgemein gesprochen – in Zahlen gefasste Attributdatenwerte von Raumobjekten in graphisch-visuelle (kartographische) Symbole.

Im vorliegenden Prototyp sind drei MapObjects-Renderer implementiert:

- **ValueMapRenderer**

  Ein ValueMapRenderer ermöglicht die kartographische Visualisierung von Daten mit nominalen und ordinalen Skalenniveau (i.e. eindeutige diskrete Werte als Zeichenketten oder ev. ganze Zahlen codiert). Jedem einzelnen Wert wird dabei genau ein graphisches Symbol zugeordnet

- **ClassBreakRenderer**

  Ein ClassBreakRenderer ermöglicht die kartographische Visualisierung von Daten mit metrischem Skalenniveau (als Fließkommazahlen oder ganze Zahlen codiert). Die Daten werden dabei über Schwellwerte in Klassen eingeteilt, welchen dann ein entsprechendes Symbol zugeordnet wird.

- **LabelRenderer**

  Ein LabelRenderer ermöglicht die kartographische Visualisierung von Daten, welche in Textform vorliegen, und hauptsächlich annotativen Character aufweisen, wie z.B. Straßen- oder Bezirksnamen.

Im Attribut *Type* des RENDERER-Elements wird spezifiziert, welcher Renderer auf den Layer angewandt werden soll.

ValueMapRenderer

Attributname	Angabe notwendig?	Beschreibung
Type	JA	"ValueMapRenderer"
Field	JA	Name des Attributfeldes mit den zugrundeliegenden (diskreten, klassifizierten) Werten
RotationField	NEIN	optionales numerisches Feld mit dem Rotationswinkel für Punktsymbole (hat Auswirkungen nur, wenn im LAYER-Element FeatureType="point" angegeben ist)
ScalingField	NEIN	optionales numerisches Feld mit der Symbolgröße für Punktsymbole (hat Auswirkungen nur, wenn im LAYER-Element FeatureType="point" angegeben ist)
UseDefault	NEIN	"true" / "false". Defaultwert ist "false". "true" Es werden alle Raumobjekte, für die kein explizites VALUE-Subelement angegeben ist, mit einem Default-symbol gezeichnet. Dieses Defaultsymbol muß als direktes SYMBOL-Unter-element im RENDERER-Element angegeben sein. "false" es werden nur diejenigen Raumobjekte visualisiert, für welche ein VALUE-Element existiert.

VALUE-Element

Es können innerhalb des RENDERER-Elements mehrere VALUE-Elemente angegeben werden. Jedes VALUE-Element muss jedenfalls genau ein Symbol-Element als Subelement enthalten!

Attribut-name	Angabe notwendig?	Beschreibung
Text	JA	Text, der in der Legende angezeigt wird
Value	JA	Wert aus dem zugrundeliegenden Feld

Bsp:

```
<RENDERER Type="ValueMapRenderer" Field="Beznr" UseDefault="true">
   <SYMBOL>
      <STYLE>1<STYLE>
   </SYMBOL>
   <VALUE Value="1" Text="Innere Stadt">
      <STYLE>2<STYLE>
      <COLOR Red="255" Green="255">
   <VALUE>
   <VALUE Value="2" Text="Leopoldstadt">
      <STYLE>0<STYLE>
      <COLOR Red="255">
   <VALUE>
   <VALUE Value="4" Text="Wieden">
      <STYLE>7<STYLE>
      <COLOR Blue="255">
   <VALUE>
</RENDERER>
```

Es wird der (Wiener Gemeinde-)Bezirk mit der *Beznr*

- 1 mit gelber Horizontalschraffur (*Style* 2),
- 2 mit roter Vollflächenfüllung (*Style* 0),
- 4 mit blauer Diagonalkreuzschraffur (*Style* 7)
- und alle anderen Bezirke mit transparenter (*Style* 1) Flächenfüllung (Defaultsymbol)

gezeichnet. Alle Polygone (auch diejenigen, welche nicht in der VALUE-Liste vorkommen!) bekommen defaultmäßig eine dünne schwarze Linie als Umrandung.

In der Legende werden neben den Symbolen die entsprechenden Texte angezeigt ("Innere Stadt" für *Beznr="1"*, "Leopoldstadt" für *Beznr="2"*, "Wieden" für *Beznr="4"* und für alle anderen Bezirke der Text "kein Wert" = der im Prototyp fixe Text für das Defaultsymbol).

ClassBreaksRenderer

Attribut-name	Angabe notwendig?	Beschreibung
Type	JA	"ClassBreaksRenderer"
Field	JA	Name des Attributfeldes mit den zugrundeliegenden (Zahlen-) Werten

BREAK-Element

Es müssen innerhalb des RENDERER-Elements mehrere (mindestens 2) BREAK-Elemente mit den Schwellwerten in aufsteigender Reihenfolge (!) angegeben werden. Jedes BREAK-Element muss seinerseits genau ein SYMBOL-Element als Subelement enthalten!

Attribut-name	Angabe notwendig?	Beschreibung
Text	JA	Text, der in der Legende angezeigt wird
Value	JA	Schwellwert (Obergrenze der Klasse)

Es werden "Anzahl-der-BREAK-Elemente" Klassen erzeugt. Der Schwellwert der letzten Klasse wird ignoriert und muss demzufolge auch nicht explizit angegeben werden. Das Symbol und der Text des letzten BREAK-Elementes wird für alle Werte verwendet, welche größergleich dem Schwellwert der vorletzten (!) Klasse sind.

Weiters können für einen ClassBreaksRenderer optional noch eine kontinuierliche Farbskala (RAMPCOLORS-Subelement) und eine wertabhängige Größenabstufung (SIZESYMBOLS-Subelement) angegeben werden. Eventuell vorhandene Angaben der relevanten Symboleigenschaften COLOR und SIZE in den einzelnen SYMBOL-Subelementen der BREAK-Elemente werden dadurch überschrieben!

```
<RAMPCOLORS>
   <STARTCOLOR Red="..." Green="..." Blue="..." />
   <ENDCOLOR   Red="..." Green="..." Blue="..." />
</RAMPCOLORS>

<SIZESYMBOLS StartSize="..." EndSize="..." />
```

Bsp.: Dreiteilige Bevöllkerungsdichteskala: blau: unter 50 Personen/ha; weiß: 50 bis unter 100 Personen/ha, rot: 100 und mehr Personen/ha)

```
<RENDERER Type="ClassBreaksRenderer" Field="BevDichte"
   <BREAK Value="20" Text="bis unter 50 Personen/ha">
      <SYMBOL>
         <STYLE>0</SYTLE>
         <COLOR Red="0" Green="0" Blue="255" /></SYMBOL>
   </BREAK>
   <BREAK Value="100" Text="bis unter 100 Personen/ha">
      <SYMBOL>
         <STYLE>0</SYTLE>
         <COLOR Red="255" Green="255" Blue="255" /></SYMBOL>
   </BREAK>
   <BREAK Text="100 und mehr Personen/ha">
      <SYMBOL>
         <STYLE>0</SYTLE>
         <COLOR Red="255" Green="0" Blue="0" /></SYMBOL>
   </BREAK>
</RENDERER>
```

LabelRenderer

Die meisten Angaben dienen zur direkten Unterstützung von ArcINFO-Annotation-Texten und wurden im Rahmen des Prototyps zwar prinzipiell ausprogrammiert, jedoch nicht umfangreich getestet (zu den einzelnen Angaben siehe auch die MapObjects-2.0-Dokumentation).

Attributname	Angabe notwendig?	Beschreibung
Type	JA	"LabelRenderer"
Field	JA	Name des Attributfeldes mit den anzuzeigenden Texten
FittedField	NEIN	Feldname mit Infos, ob der Text zwischen 2 Punkten eingepasst werden soll (nur für Linien)
HeightField	NEIN	Feldname mit numerischer Information über die Textgröße in MapUnits (d.h. es wird beim Zoomen mitskaliert)
LevelField	NEIN	Feldname mit numerischen Annotation-Levels für die selektive Anzeige von Texten (siehe MinLevel, MaxLevel)
RotationField	NEIN	Feldname mit numerischen Rotationswinkeln in Grad gegen den Uhrzeigersinn
SymbolField	NEIN	Feldname mit ganzzahligen Indexwerten >= 0 (s. Text unten)
XOffsetField	NEIN	Feldname mit numerischen Verschubwerten in x-Richtung
YOffsetField	NEIN	Feldname mit numerischen Verschubwerten in y-Richtung
AllowDuplicates	NEIN	"true" / "false". Default ist "true". Gibt an, ob doppelt vorkommende Labeltexte nur einmal ("false") oder immer ("true") gezeichnet werden.
DrawBackground	NEIN	"true" / "false". Bewirkt bei "true", dass vor dem Zeichnen der Labeltexte die räumlichen Objekte selbst (z.B. Polygone) mit einem Defaultsymbol visualisiert werden.
MinLevel	NEIN	Zahl. (siehe LevelField) Es werden nur Texte mit mindestens diesem Level angezeigt
MaxLevel	NEIN	Zahl. (siehe LevelField) Es werden nur Texte mit höchstens diesem Level angezeigt
SplinedText	NEIN	"true" / "false". Default ist "true". Gibt an, ob sich bei linienförmigen Raumobjekten der Text an die Linie anpassen soll.

Innerhalb des Label-RENDERER-Elements muss mindestens ein TEXTSYMBOL-Subelement spezifiziert werden. Werden mehrere TEXTSYMBOL-Subelemente angegeben, kann für jedes Label eines davon mit dem *SymbolField* über die Attributtabelle spezifiziert werden.

Bsp:

```
<RENDERER DrawBackground="false" Field="Annotext" Type="LabelRenderer">
   <TEXTSYMBOL>
      <COLOR Blue="100" Green="100" Red="100"/>
      <FONT>Arial</FONT>
      <SIZE>9</SIZE>
   </TEXTSYMBOL>
</RENDERER>
```

Darstellungsteil der FIS-MapView-Komponente

Entwurfsziele

Karten sind ein zentrales Element im Umgang mit dem großteils raumbezogenen Datenbestand des FIS. Sie bilden eine wichtige Grundlage intuitiver Interpretation raumbezogener Sachverhalte und werden in dieser Funktion durch tabellarische Informationen, Abbildungen und Textdokumente ergänzt.

Ziel ist es daher dem Benutzer ein Werkzeug zur Verfügung zu stellen, mit dessen Hilfe er kartographisch aufbereitete Inhalte aus dem Datenbestand des FIS abrufen und mit dieser Karte interagieren kann. Eine einfach zu bedienende Oberfläche soll gewährleisten, dass die Benutzer ohne hohen Schulungsbedarf in der Lage sind, möglichst das gesamte Potenzial des Werkzeuges effizient zu nutzen.

Vor dem Hintergrund der geforderten Einfachheit ist dieses Potenzial zwar grundsätzlich beschränkt, im Rahmen der Möglichkeiten muss die FIS-MapView-Komponente jedoch ein vielseitiges und flexibles Hilfsmittel für den täglichen Umgang mit interaktiven Karten sein.

Folgende Anforderungen ergeben sich daher an das Werkzeug

- weitgehend selbsterklärende Benutzeroberfläche - Übersichtlichkeit
- Benutzerführung über vertraute Web-Konzepte
- räumliche und inhaltliche Orientierbarkeit
- größtmögliche Flexibilität bei maximaler inhaltlicher und räumlicher Fokusierbarkeit
- Anzeigen und Ausblenden von thematischen Kartenschichten
- Ver“Link“barkeit

Funktionalität

Karten werden im FIS vom Datenanbieter über den Layoutteil der FIS-MapView-Komponente inhaltlich vordefiniert. Sie bestehen aus einer oder mehreren Datenschichten, die jeweils nach thematischen Gesichtpunkten vom Kartenautor symbolisiert werden. Der Darstellungsteil der FIS-MapView-Komponente erzeugt auf Basis der Voreinstellungen des Layouters Karten im Sinne "vorbereiteter Sichten", auf die der Benutzer über den Browser interaktiv zugreift. Auf die Zusammenstellung der Inhalte und die Darstellungsart der einzelnen Themenlayer innerhalb der Karte hat der Benutzer bei der interaktiven Abfrage jedoch keinen unmittelbaren Einfluss – dies bleibt dem Kartenautor vorbehalten.

Eine Karte besteht in der prototypischen Implementierung aus folgenden Elementen:

- Kartendarstellung und Navigationselemente (für jede Karte in einem eigenen Browser-Fenster, dem MapViewer-Fenster)
- Kartenlegende und Steuerelemente für die vorhandenen Layer (im MapViewer-Fenster)
- Übersichtskarte und Steuerelemente in der FIS-Startseite

Es folgt eine detaillierte Darstellung:

Kartendarstellung und Navigationselemente

Das Browserinterface der FIS-MapView-Komponente bietet dem Benutzer folgende einfache Navigationsmöglichkeiten:

- Zoomen
  - Veränderung des dargestellten Kartenausschnittes um einen bestimmten Faktor über Steuerelemente,
  - Definition des Ausschnittes über einen interaktiv festgelegten Auswahlbereich durch Mausklick in die Kartendarstellung
- Pan
  - Verschieben des Kartenausschnittes zum unmittelbar benachbarten Ausschnitt (in der gleichen Größe) mittels Befehlsschaltflächen
  - Setzen des neuen Zentrums durch Mausklick in die Karte oder die Übersichtskarte
- Verändern des Kartenausschnittes mittels "benannter Ausschnitte“
  - benutzerspezifische Definition von benannten Ausschnitten (gültig innerhalb der aktuellen Arbeitssitzung)
  - Setzen des anzuzeigenden Ausschnitts über bereits zur Entwurfszeit festgelegte benannte Ausschnitte
- Abrufen von Attributinformationen durch Mausklick in die Kartendarstellung.
  Abfrageergebnisse für selektierte Raumobjekte werden in Tabellenform dargestellt (siehe FIS-TableView-Komponente)

Weitere Steuerungsmöglichkeiten im Browserinterface

- Auslösen der Aktualisierungsfunktion von Kartendarstellung und Legende (Neuzeichnen)
- Definition von Kartenformaten, i.e. Bilddatenformat der dargestellten Kartenabbildung und deren Größe (Breite und Höhe) in Pixel

Kartenlegende

Ziel ist eine einfache und klare inhaltliche Orientierung des Benutzers durch die Legendendarstellung. Jedem Kartenlayer ist eine Legende zugeordnet, wobei die Symbolisierung jener in der Karte entspricht und ebenso wie deren Klassenbeschriftungen aus der im Layoutteil der Komponente definierten Komponentenspezifikationsdatei übernommen werden.

Im Sinne einer klaren Struktur sind Legende und layerbezogene Steuerungsfunktionen in einem eigenen Bereich – dem Table of Contents (TOC) links neben der eigentlichen Karte – zusammengefasst. Über den TOC erfolgt (jeweils in Abhängigkeit von den Vorgaben des Kartenautors)

- das Ein- und Ausblenden von Themen in der Karte,
- das Ein- und Ausblenden von Legenden im TOC,
- die Aktivierung/Deaktivierung der Info-Abfrage-Funktion,
- die Aktivierung/Deaktivierung der Filterfunktion über Attributwerte.

Übersichtskarte

Als räumliche Orientierungshilfe für den Benutzer enthält das FIS-Startfenster eine Übersichtskarte. Diese zeigt das Raumbezugssystem der dargestellten Karte (im implementierten Prototyp ist das die Bezirkskarte von Wien) und veranschaulicht den angezeigten Kartenausschnitt mittels Rechteck und Fadenkreuz.

Die Übersichtskarte hat bezüglich der geöffneten MapView-Karten v.a. Orientierungs- und Steuerungsfunktion:

- Karte und Übersichtskarte sind synchronisiert, d.h. bei jeder Veränderung des Ausschnittes in der Karte wird die Darstellung der Übersichtskarte entsprechend aktualisiert
- die Überblickskarte unterstützt mausgesteuerten Zoom/Pan und aktualisiert nach einer Aktion die Karte im zuletzt aktiven MapView-Fenster
- die Übersichtskarte informiert mittels Steuerelement, welche Karte aktiv ist und erlaubt das Setzen des Fokus auf andere geöffnete Karten
- im Interface der Überblickskarte sind für das Raumbezugssystem der Karte im aktiven Karten-Fenster "benannte Ausschnitte“ vordefiniert bzw. können dort vom Benutzer festgelegt werden, d.h. der Benutzer kann von der Übersichtskarte aus den anzuzeigenden Kartenausschnitt auf einen dieser "benannten Ausschnitte“ setzen.
- bei mehreren geöffneten Kartenfenstern kann der aktuelle Raumausschnitt einer Karte auf alle übrigen Karten übertragen werden, die dann ebenfalls diesen Ausschnitt zeigen (Synchronisieren des räumlichen Ausschnittes)

Implementierung

Die Implementierung des Darstellungsteils der FIS-MapView-Komponente erfolgt serverseitig mit Hilfe von zwei ASP-Scripts (*MapEngine.asp* und *MapOverview.asp*), die beide eine ActiveX-EXE-Datei (*MO2XMLDisplay.exe*) aufrufen, welche die durch interaktive Benutzer-Requests modifizierte XML-Kartenspezifikationsdatei (*.xmm) in eine Bilddatei (Format JPG oder PNG) und ggf. in eine XML-Table-Struktur mit den Info-Abfrageergebnissen übersetzt. Diese Ergebnisse werden dem Browser als sog. *Response* übergeben.

Sämtliche Programme und Hilfsdateien (**.asp, *.xmm, leg_pkt2.**) sind zentral im FIS-Systemordner */fis/engine/MapView* abgelegt. Die Datei *MO2XMLDisplay.exe* befindet sich im Ordner */fis/dll* und muss als ActiveX-Server-EXE registriert sein (siehe Installationsdokumentation).

Clientseitig übernimmt pro Karte eine Frameset aus speziellen HTML-Dokumenten in einem eigenen Browser-Fenster unter Zuhilfenahme verschiedener JavaScript-Funktionen die Anzeige und interaktive Bearbeitung der Karte. Sämtliche Dateien (**.htm, *.js*) sind zentral im FIS-Systemordner */fis/engine/MapView* abgelegt.

Die Anzeige der Übersichtskarte erfolgt im aktuellen Prototyp immer im oberen Frame der FIS-Startseite (siehe MapView-Dokumentation).

Für jede Attribut-Infoabfrage wird automatisch ein eigenes Fenster mit einer FIS-TableView-Komponente geöffnet (siehe MapView- und TableView-Dokumentation).

Integration der FIS-MapView-Komponente in eine HTML-Seite

Der Aufruf von interaktiven Web-Karten erfolgt im Allgemeinen über die FIS-TreeView-Komponente, welche den innerhalb des FIS vorstrukturierten Zugriffspfad zu sämtlichen Inhalten bildet (so auch zu Karten, für eine ausführliche Behandlung siehe Seite 27).

Im vorliegenden Prototyp wird für jede Karte ein eigenes Fenster am Client geöffnet. Eine direkte Einbettung in ein HTML-Dokument in dem Sinne, daß die Karte ein Element unter vielen in dieser Seite im selben Browserfenster bildet, ist daher zur Zeit nicht möglich (siehe dazu auch Erweiterungsmöglichkeiten auf Seite 80ff).

Die Spezifikation eines Verweises (Link, Aufruf) auf eine Karte wurde jedoch vom Auftragnehmer grundsätzlich vorgesehen. Alternativ zur erwähnten impliziten Veröffentlichung von Karten im FIS über die TreeView-Komponente ist es möglich, einen Verweis von einer beliebigen Stelle innerhalb eines FIS-Web-Dokuments auf eine Karte zu erzeugen.

Der Verweis erfolgt standardgemäß mit Hilfe eines A-Tags und eines HREF-Attributs:

```
<A HREF="/fis/engine/mapview/MapEngine.asp?template=/..Pfad../KartenName.xmm">
Link zur Karte</A>
```

Hinweis 1

Der *Pfad* zur Karte muss – analog zum Pfad zur *MapEngine.asp* – absolut, ausgehend von der Wurzel des virtuellen FIS-Verzeichnisses, angegeben werden. Die Angabe der Dateierweiterung .xmm ist ebenfalls notwendig, also z.B.

```
<A HREF="...?template=/fis/content/wasserwerke/standorte.xmm"> ... </A>
```

Hinweis 2

Voraussetzung ist jedoch, daß jede HTML-Seite, welche zwischen dem Hauptindex von FIS (i.e. die Eingangsseite des FIS mit der Überblickskarte) und der aktuellen Seite einen Link auf die FIS-Standard-Formatvorlage enthält.

```
<LINK REL="stylesheet" TYPE="text/css" HREF="/fis/css/Standard.css"></LINK>
```

Der tiefere Grund dafür ist, dass das Standardverhalten der A-Tags bei seiner Aktivierung (*onclick*) über ein *Custom Behavior* (HTC-Datei) verändert werden muss, damit von jeder Seite im FIS das Fenster mit der Überblickskarte (die FIS-Startseite) durch Rückverfolgen "gefunden" werden kann.

Die Überblickskarte ist im vorliegenden Prototyp die zentrale Schaltstelle zur Verwaltung sämtlicher geöffneter Karten. Über sie laufen sämtliche Öffnen-, Schließ- und Synchronisationsaktionen.

Möglichkeiten zur Weiterentwicklung

- **"Freie" Graphiken aus ArcView-GIS übernehmen**

  In ArcView-GIS kann eine kartographische Ansicht (*view*) zusätzliche Annotationen in Form von freien Texten und Graphiken enthalten, wovon manche darüber hinaus noch mit einer bestimmten Datenschicht (Layer, Thema) assoziiert sein können.

  Die Anzeige von graphischen Formen ist im serverseitigen Darstellungsteil der MapView-Komponente prinzipiell bereits implementiert (in *MO2XMLDisplay.exe*), wird aber im Prototyp zur Zeit lediglich zur Anzeige der Maßstabsleiste direkt in der Karte verwendet.

  Der Integration weiterer derartiger Graphiken in FIS-Karten steht somit weiter nichts entgegen – genau zu spezifizieren ist dabei

  - ob und wie die Graphiken ein- und ausgeschaltet werden können und
  - ob und wie eine allfällige Verknüpfung zu einer Kartenschicht aus ArcView übernommen werden soll.

- **Weitere kartographische Darstellungsformen unterstützen**

  Zu überlegen ist die Implementierung weiterer in MapObjects 2.0 verfügbarer Renderer, wie z.B. Chart- , Z-DotDensity-, Event- und Group-Renderer.

- **Hotlinks**

  Via Hotlinks könnten einzelne in einer Karte dargestellten Raumobjekte als Verweisanker zu weiteren HTML-Dokumenten dienen. Im vorliegenden Prototyp ist dies nur indirekt über die Info-Abfrage möglich, wenn der Name einer Attributspalte mit den Zeichen "URL_" beginnt (siehe FIS-TableView-Dokumentation).

  Die Implementierung könnte in HTML über sog. AreaMaps erfolgen:

```
<MAP id="...">
     <AREA coords="..." href="..." target="..."></AREA>
     ...
     <AREA coords="..." href="..." target="..."></AREA>
</MAP>
```

  Da sich der Aufbau derartiger AreaMaps bei einer großer Zahl von Raumobjekten sehr (zeit-)aufwendig gestalten kann, ist eventuell eine Maßstabsabhängigkeit für diese Möglichkeit vorzusehen.

- **ToolTips**

  Ein ToolTip ist ein meist kurzer und informativer Text, der automatisch in einem kleinen Rechteck erscheint, wenn der Mauszeiger für eine bestimmte Zeit über einem Element im Browser verweilt.

  Die Implementierung erfolgt in HTML über das sog. Titel-Attribut (Titel="...infotext...") und könnte im Kartenbereich wie bei den Hotlinks über eine AreaMap erfolgen.

  Es könnten hier automatisch ausgewählte Attributspalten nach einer zuvor erfolgten Info-Abrage angezeigt werden. Welche Attribute in den ToolTip aufgenommen werden, sollte dann im clientseitigen Darstellungsteil der TableView-Komponente einstellbar sein.

- **Integrationsmöglichkeit von Karten in beliebige HTML-Dokumente**

  Der serverseitige Darstellungteil der MapView-Komponente könnte – analog TableView und TreeView – mit Hilfe von *Custom Behaviors* (HTC-Dateien) und IFRAMES direkt in den Quelltext von beliebigen HTML-Dokumenten eingebettet werden.

  In diesem Zusammenhang ist die zentrale Funktion der FIS-Übersichtskarte im Prototyp grundsätzlich zu überdenken und ggf. neu zu entwerfen.

- **Verwaltung verschiedener räumlicher Bezugssysteme**

  Diese Aufgabe kann unter Einbeziehung der geographischen WUIS-Metadaten gelöst werden. Es muss zumindest feststellbar sein, welchem konkreten räumlichen Bezugssystem und welcher damit verbundenen kartographischen Projektion eine anzuzeigende Karte bzw. Datenschicht angehört.

  Eine derartige Erweiterung hätte zur Folge, daß mehrere Überblickskarten gleichzeitig im Browser zu verwalten wären. Eine Möglichkeit für deren Organisation besteht in der – im FIS schon mehrfach verwendeten und somit dem Benutzerkreis vertrauten – Form von Registerblättern im Übersichtsfenster.

  Die Angabe in der XML-Komponenten-Spezifikationsdatei könnte über ein zusätzliches Attribut im MAP-Element (Universe="...") und die Auswahl der XML-Spezifikation für die Übersichtskarte über eine Namenskonvention erfolgen, z.B.:

`<MAP Universe="wien_rbw"...>...</MAP>`	→ Overview_wien_rbw.xmm
`<MAP Universe="austria_lambert"...>...</MAP>`	→ Overview_austria_lambert.xmm
`<MAP Universe="europa"...>...</MAP>`	→ Overview_europa.xmm

- **Adress-Komponenten**

  Die Angabe eines Standortes über eine Wohnadresse (z.B. "Wien 4, Karlsgasse 13") ist weit verbreitet und gehört im Rahmen von räumlichen Informationssystemen (auch web-gestützten) zum fixen Repertoire an Standardfunktionen.

  Auch während der Arbeitssitzungen (SRF-MA14-MA22) im Laufe der Auftragsbearbeitung wurde dieses Thema mehrfach angesprochen. Eine eigene Adress-Komponente wurde jedoch im Zuge des Forschungsprojektes "FIS – Konzept eines Fachinformationssystems" (noch) nicht im Prototyp implementiert. Die Integration einer derartigen Komponente ist prinzipiell möglich und für eine weitere Ausbaustufe bereits in Form einer (vorerst noch deaktivierten) Befehlsschaltfläche in der FIS-MapView-Komponente vorgesehen.

  Das Thema "Adressen" musste jedoch aus Sicht des Auftragnehmers grundsätzlich in einem größeren Zusammenhang diskutiert und tiefgehend behandelt werden, nämlich vor dem Hintergrund möglicher

  - verschiedener räumlicher Bezugssysteme,
  - räumlicher Hierarchien,
  - "normaler" Wohnadressen, und
  - benannter Ausschnitte.

  Dieses Erweiterung müsste ggf. auch in Verbindung mit der Weiterentwicklung der FIS-TreeView-Komponente betrachtet werden, welche in dieser Frage wertvolle Hilfestellung leisten könnte.

- ENDE -