



# WebTML

Web Template Markup Language

Tutorial

---

Copyright © 2002 by: Innovation Gate GmbH  
40880 Ratingen  
Harkortstr. 21-23  
Tel.: 02102 - 77 16 0 - 0  
[www.innovationgate.de](http://www.innovationgate.de)  
[info@innovationgate.de](mailto:info@innovationgate.de)

Alle Rechte vorbehalten, kein Teil dieses Dokumentes darf in irgendeiner Form (Druck, Fotokopie oder sonstige Verfahren) ohne schriftliche Genehmigung von Innovation Gate reproduziert oder vervielfältigt werden.

Änderungen am Dokument und den beschriebenen Konzepten bleiben jederzeit und ohne vorherige Ankündigung vorbehalten.

Version:	1.213
letzte Änderung:	23. September 2004
Status:	- Released -

# Inhaltsverzeichnis

- 1. Einleitung.....3**
  - Vergleich WebTML mit Script-Sprachen.....4
  - Vergleich WebTML mit XSLT.....5
  - Vergleich WebTML mit "Textersetzungsmakros".....5
  - Die Vorteile von WebTML.....6
  - Weitere Informationen.....6
- 2. Die Entstehung einer Website.....7**
  - Erstellen des Grundlayouts.....7
  - Zugriff auf Grafikressourcen: <tml:img>.....8
  - Verwenden von WebTML-Bibliotheken: <tml:include>.....9
  - Erstellen von Navigatoren: <tml:siblings>.....10
  - Bedingte Codeerzeugung: <tml:if>.....13
  - Subnavigation: <tml:children>.....14
  - Zugriff auf Content-Felder: <tml:item>.....15
  - Kollektionen in WebTML: <tml:collection>.....17
  - Zusammenfassung.....19

# 1. Einleitung

Wer in der Vergangenheit seinen Web-Auftritt auf ein professionelles Content Management System stützen wollte, machte meist die Erfahrung, dass diese Systeme zwar grundsätzlich sehr praktisch sind, aufgrund der bis heute entwickelten Programmier-Technik unterliegen sie jedoch immer gewissen Beschränkungen, vor allem im Layout. Einen dynamischen Seiten-Navigator zu konfigurieren, der basierend auf den erfassten Dokumenten beliebigen HTML-Code generiert, war in der Regel nicht ohne Sonder-Programmierung und erheblichen zusätzlichen Aufwand durch den Hersteller möglich. Und selbst dann begann bei jeder Veränderung des Layouts der Programmier-Vorgang aufs Neue.

Bislang konnten Programmierer oder Anwender von Content Management Systemen die Basiselemente einer Website nur auf einem sehr niedrigen Level gestalten. Als Hilfsmittel standen dafür je nach System Active Server Pages (Microsoft), Java Server Pages (Sun) oder Domino Server Pages (Notes) zur Verfügung. Doch mit diesen ist es bis heute nicht möglich, auf High-Level-Elemente eines CMS zuzugreifen. So gestaltet sich die Programmierung von Navigatoren oder Suchfunktionen nach wie vor äußerst aufwändig und komplex. Je nach Anspruch des Kunden müssen CMS daher oft von Hand "nachprogrammiert" werden.

Um in der Definition von Web-Templates wesentlich flexibler zu werden, hat Innovation Gate die Programmiersprache WebTML (Web-Template-Markup-Language) entwickelt.

WebTML basiert auf XML und beinhaltet Befehle (Tags), die speziell für Content Management Systeme entwickelt wurden. Die Funktionsdichte dieser Programmiersprache entsteht auch dadurch, dass WebTML-Tags beliebig ineinander verschachtelt werden können und sich dadurch komplexe Probleme lösen lassen. So können z.B. innerhalb eines Navigators ein weiterer Navigator eingeblendet oder Suchergebnisse (sogenannte "Kollektionen") dargestellt werden.

Neben Tags zur Darstellung von Navigatoren und Kollektionen sowie Tags für Schleifen (<tml:foreach>) und logische Verzweigungen (<tml:if>, <tml:elseif>, <tml:else>; <tml:case>) bietet WebTML auch spezielle "Editor-Tags", für die direkte Contentpflege mit dem Browser (inline-editing): Die Stellen, an denen Änderungen im Inhalt durchgeführt werden können bzw. dürfen, werden durch den Editor-Tag (<tml:item editor="text">) markiert. Dort erscheint im Browser dann ein grafisches Bearbeiten-Symbol. Ein einfacher Klick genügt, und ein Autor kann genau diesen Teil des Dokumentes verändern.

Abgerundet wird WebTML durch Tags für die Unterstützung von Unternehmensportalen, sowie die Unterstützung von Formularen (<tml:form>, <tml:input>).

Man kann in Web-TML zum Beispiel folgendes schreiben:

```
<tml:foreach type="loop" count="3">
  Irgendwas
</tml:foreach>
```

Nun wird zur Laufzeit nicht einfach nur Text ersetzt, sondern das, was zwischen dem foreach-Start- und -End-Tag steht wird so oft ausgegeben, wie die Schleife durchlaufen wird:

```
Irgendwas
Irgendwas
Irgendwas
```

Diese Programmierung kann natürlich intelligenter genutzt werden: Innerhalb eines Tags könnten zum Beispiel weitere Tags eingebaut werden.

Zum Beispiel gibt es auf vielen WebSites alphabetische Linkleisten (Beispiel: Messe CeBIT: Ausstellerverzeichnis): A, B, C,...Z. Klickt man auf einen dieser Buchstaben ("C"), wird eine automatische Suche durchgeführt und alle Aussteller mit "C" werden angezeigt. Das ist ein typisches Beispiel, in dem 26 mal der gleiche HTML-Code gebraucht wird, abstrakt: "zeige hier einen Link an mit dem Buchstaben A und starte dann eine Suche nach allem was mit A anfängt". Dies wird dann 26 Mal mit verschiedenen Buchstaben wiederholt. Mit WebTML kann man dieses Fragment als eine einzige Schleife realisieren, indem man innerhalb dieser Schleife so variabel formuliert, dass für alle 26 Buchstaben der richtige Inhalt berechnet wird:

```
<tml:foreach type="loop" count="26" var="c">
    <tml:evaluate var="char">
        return String.fromCharCode(64+this.c)
    </tml:evaluate>
    <a href="javascript:search('<tml:item name="char"/>')">
        <tml:item name="char"/>
    </a>
</tml:foreach>
```

Dieses Beispiel verwendet den `<tml:evaluate>`-Tag, um innerhalb der Schleife den darzustellenden Buchstaben zu berechnen und in ein Content-Feld "char" zu schreiben. Dieses Feld wird nun mit Hilfe des `<tml:item>`-Tags als Link ausgegeben.

Auf den ersten Blick wird die Programmierung mit Web-TML komplizierter: Man kann eine Website weniger mit Point und Click zusammenbauen, sondern man muss die Templates programmieren. Das ist eine Art der Programmierung, die deutlich in die Tiefe geht und komplexer ist, als die einfache Textersetzung, wie sie im Markt bislang eingesetzt wird.

Der große Vorteil von Web-TML liegt darin, dass die Programmierung flexibler wird. Die Sprache bietet wesentlich mehr Flexibilität in der Gestaltung der Templates.

Mit einem einfachen Algorithmus kann man zum Beispiel einen Navigator erstellen. Dazu gibt es den Tag `<tml:siblings>` (Geschwister). Alles, was zwischen Beginn und Ende dieses Tags steht, erscheint so oft im resultierenden HTML-Code, wie es Geschwister-Dokumente gibt:

```
<tml:siblings>
    <li><tml:link/></li>
</tml:siblings>
```

In diesem Beispiel wurde in drei Zeilen ein Navigator mit HTML-Formatierung programmiert. Ausgegeben werden, als Bulleliste, alle Links, die zu Dokumenten gehören, die logisch auf der selben Hierarchie-Ebene liegen, wie das aktuell angezeigte Dokument.

```
• Link 1
• Link 2
• Link 3
• ...
• Link n
```

Mit dem in WebGate Anywhere implementierten WebTML kann jeder Site-Designer, mit Hilfe von Algorithmen, einfach Erweiterungen der Webseiten vornehmen. Dabei existiert keine Beschränkung durch die Vorgaben des Content Management Anbieters. Mit Hilfe von WebTML ist es sogar möglich, einen dynamischen Navigator anlegen, der die Links kreisförmig anordnet oder Javascript-Code für DHTML-Menüs erzeugt.

Darüber hinaus bietet WebTML die Möglichkeit, unterschiedliche Elemente auf einer Seite aus ganz unterschiedlichen Datenquellen darzustellen. Auf diese Weise lässt sich eine Navigation erstellen, die aus verschiedenen Webauftritten gespeist wird.

Insbesondere großen Unternehmen und Konzernen erlaubt diese Technik erhebliche Kosteneinsparungen bei Produktion und Administration der Inhalte: Jede Abteilung kann autark arbeiten und den erarbeiteten Content in verschiedene globale Web-Auftritte einfließen lassen.

## Vergleich WebTML mit Script-Sprachen

Es existieren eine Reihe von Script-Sprachen, mit denen webfähige Anwendungen entwickelt werden können. Dazu gehören PHP, Perl, TLC, aber auch die in HTML eingebetteten Script-Sprachen wie Active Server Pages von Microsoft (HTML mit eingebettetem Visual Basic), Java Server Pages von Sun (HTML mit eingebettetem Java) und auch die in WebGate 5 eingeführten Domino Server Pages (HTML mit eingebettetem LotusScript und Lotus-Formelsprache).

Welche Vorteile bietet nun WebTML gegenüber diesen etablierten Script-Sprachen?

Die genannten Script-Sprachen bieten Mechanismen für die Programmierung von Schleifen (loop, forall, etc.), bedingten Anweisungen (if) und bieten Zugriff auf HTTP-

spezifische Parameter (z. B. URL-Parameter, Cookie-Daten etc). Häufig bieten sie darüber hinaus die Möglichkeit, auf externe Systeme wie relationale Datenbanken zuzugreifen und das Ergebnis in die aktuelle Webseite einzufügen.

Diese Sprachen wurden entwickelt, um statische HTML-Seiten um dynamische Elemente zu ergänzen.

In Content-Management-Systemen wie WebGate Anywhere liegen die Webseiten jedoch nicht als fertige HTML-Seiten vor, sondern werden dynamisch bei jedem Zugriff berechnet und zum Browser gesendet. Während dieser Berechnung werden Layoutinformationen aus den Templates mit den vom Autor in Content-Feldern erfaßten Daten zusammengemischt. In der Regel werden darüber hinaus zusätzliche Elemente wie Navigatoren oder Kollektionen in die Seite eingefügt.

Wesentlicher Nachteil der oben genannten Script-Sprachen ist, dass sie (ohne erheblichen Programmieraufwand) keine Zugriffsfunktionen auf Content-Management-spezifische "Objekte" wie Content-Felder, Navigatoren oder Kollektionen zulassen.

WebTML ist entwickelt worden, um insbesondere solche abstrakten Zugriffe auf CMS-Objekte zu ermöglichen. So existierte etwa ein WebTML-Tag `<tml:item>`, um gezielt auf ein Content-Feld zuzugreifen oder `<tml:children>`, um einfach durch alle "untergeordneten" Webseiten (Dokumente) zu iterieren und damit Navigationselemente aufzubauen. Mit reinen Script-Sprachen sind dazu umfangreiche Kenntnisse über die interne Struktur der Content-Datenbank notwendig. Diese internen Strukturen sind in der Regel nicht offengelegt und könnten sich außerdem durchaus beim nächsten Releasewechsel ändern.

WebTML ist dabei nicht als Ersatz, sondern als sinnvolle Ergänzung zu obigen Script-Sprachen zu sehen.

## Vergleich WebTML mit XSLT

Mit XSLT steht ein standardisierter Mechanismus zur Verfügung, XML-Dokumente dynamisch in HTML zu wandeln und anzuzeigen. Da CM-Systeme wie WebGate Anywhere Seiten als XML-Dokumente zur Verfügung stellen können, kann XSLT verwendet werden, um aus diesen "Rohdaten" fertige HTML-Seiten im gewünschten Layout zu erzeugen.

Zwar ist es mit XSLT anders, als mit Script-Sprachen komfortabel möglich, auf Content-Felder zuzugreifen, XSLT kann dabei jedoch immer nur das aktuelle Dokument berücksichtigen. Um Navigationselemente zu berechnen, sind jedoch Informationen aus "benachbarten" Content-Dokumenten in die aktuelle Seite einzubinden – und genau dies kann XSLT im Gegensatz zu WebTML nicht.

Genau wie bei Script-Sprachen, ist der Einsatz von WebTML nicht als Ersatz, sondern als sinnvolle Ergänzung zu XSLT zu sehen.

## Vergleich WebTML mit "Textersetzungsmakros"

Die meisten im Markt verfügbaren CM-Systeme verwenden zur Definition von Templates einfache Makros, die in der HTML-Definition eingegeben werden und zur Laufzeit per "Textersetzung" durch das Makro-Ergebnis ersetzt werden.

Ein typisches Template könnte hier etwa wie folgt aussehen:

```
Hier steht HTML-Code
...
Feldausgabe:
Das Contentfeld "Titel" hat den Wert ##Field("Titel")##
Hier steht weiterer HTML-Code
```

Ein Makroparser sucht zur Laufzeit nach Textelementen, die zwischen `##` und `##` stehen, interpretiert diesen Text als Makrokommando, führt das Makro aus und schreibt das Makroergebnis (hier der Inhalt des Content-Feldes "Titel") an die Stelle im Text, an der das Makro steht.

Der große Nachteil dieser Arbeitsweise liegt darin, dass diese Makros nicht ineinander verschachtelt werden können. Existiert etwa ein Makro `##Navigator(P1, P2, P3, ...)##`, das einen Navigator erzeugen soll, dann wird das Erscheinungsbild des Navigators stets durch eine feste Anzahl von Parametern gesteuert. Hier kann der Navigator in genau dem

Maße beeinflusst werden, der vom Hersteller des CMS vorgesehen ist.

In WebTML kann der Designer den Navigator dagegen komplett frei gestalten. WebTML selbst trägt nicht zur Gestaltung bei. Es bietet lediglich die Möglichkeit, durch die Liste der Dokumente zu iterieren, die zur Erzeugung des Navigators herangezogen werden sollen. Die Formatierung ist dem Designer überlassen. Dabei können innerhalb der Iteration ohne Probleme weitere WebTML-Tags eingefügt werden, die z.B.: "Subnavigatoren" innerhalb eines Navigators einblenden etc.

Das folgende Beispiel erzeugt eine "Sitemap" mit Links auf alle Dokumente der obersten beiden Navigationsebenen:

```
<tml:siblings context="root">

    <tml:link/><br>
    <ul>
        <tml:children>
            <li><tml:link/></li>
        </tml:children>
    </ul>

</tml:siblings>
```

Innerhalb der Iteration durch die Hauptdokumente (<tml:siblings>-Tag) wird zunächst ein Link ausgegeben (<tml:link>-Tag) und dann bezüglich dieses Hauptdokumentes durch alle seine untergeordneten Dokumente iteriert (<tml:children>-Tag). Danach wird die Iteration durch die untergeordneten Dokumente des nächsten Hauptdokuments begonnen.

Ein solcher verschachtelter Navigator ist bei der Verwendung von Makros nur möglich, wenn der CMS-Hersteller dies explizit vorgesehen hat. Mit WebTML kann die Darstellung jedoch weitestgehend beeinflusst werden.

## Die Vorteile von WebTML

- WebTML bietet komfortable Zugriffsfunktion auf Content-Felder.
- WebTML bietet Schleifenkonstrukte und bedingte Ausgaben.
- WebTML-Tags können verschachtelt werden und bieten so hohe Flexibilität im Template-Design.
- WebTML unterstützt das Iterieren durch Dokumentmengen zum Aufbau von Navigatoren und Kollektionen.
- WebTML bietet Zugriffsfunktionen auf HTTP-Parameter.
- WebTML bietet Unterstützung von Portlets für Web-Portale.
- WebTML unterstützt personalisierte Web-Inhalte.
- WebTML unterstützt das "Inline-Editing" mit dem Browser.
- WebTML bietet Zugriffsmöglichkeiten auf "externen" Content.
- WebTML unterstützt Webservices zur Integration von Fremdsystemen.
- WebTML unterstützt den Aufbau von Template-Bibliotheken.
- WebTML unterstützt die direkte Verknüpfung von Web-Formularen mit Content-Feldern.
- WebTML ist leicht erlernbar durch die Verwendung von HTML- bzw. XML-Syntax.

## Weitere Informationen

Die maßgebliche Informationsquelle für alle Neuigkeiten rund um WebGate Anywhere ist [www.innovationgate.com](http://www.innovationgate.com). Auf der Website von Innovation Gate finden Sie ausführliche Informationen über Innovation Gate und seine Produkte und Leistungen und werden aktuell über Updates und neue Erweiterungen unterrichtet.

Hinweise und Referenzen für WebGate Anywhere Entwickler finden Sie unter [dev.innovationgate.com](http://dev.innovationgate.com).

## 2. Die Entstehung einer Website

Das folgende Kapitel soll Schritt für Schritt die Entstehung einer Website mit WebTML beschreiben, bevor wir in den weiteren Kapiteln die WebTML-Syntax detailliert und vollständig beschreiben.

Das Ziel dieses Kapitels ist es, mit WebTML das Layout der folgenden Website zu erstellen:



Die Website besteht dabei aus einem Grundlayout, bei dem im oberen Bereich ein Dokumentkopf mit dem Firmenlogo dargestellt wird. Darunter werden auf der linken Seite die Hauptnavigation und rechts daneben der Dokumentinhalt dargestellt. Unter dem Dokumentinhalt sollen Links auf aktuelle Pressemeldungen gezeigt werden. Schließlich folgt ein Fußbereich mit einem Copyright-Hinweis.

Wir werden in diesem Kapitel die einzelnen Schritte darstellen, die zur Erzeugung der Website notwendig sind, ohne zu diesem Zeitpunkt auf alle Details einzugehen.

### Erstellen des Grundlayouts

Wir starten mit der Erstellung des Layouts, in dem wir zunächst das Grundlayout als reines HTML in einem WebGate Anywhere Dokumentlayout erfassen. Der WebTML-Code sieht wie folgt aus:

```
<div align="center">
*** WebTML Tutorial DEMO Website ***
<table border="1" bordercolor="black" width="90%" height="90%">
<tr>
    <td colspan="2" bgcolor="black">
    ** Dokumentkopf **
```

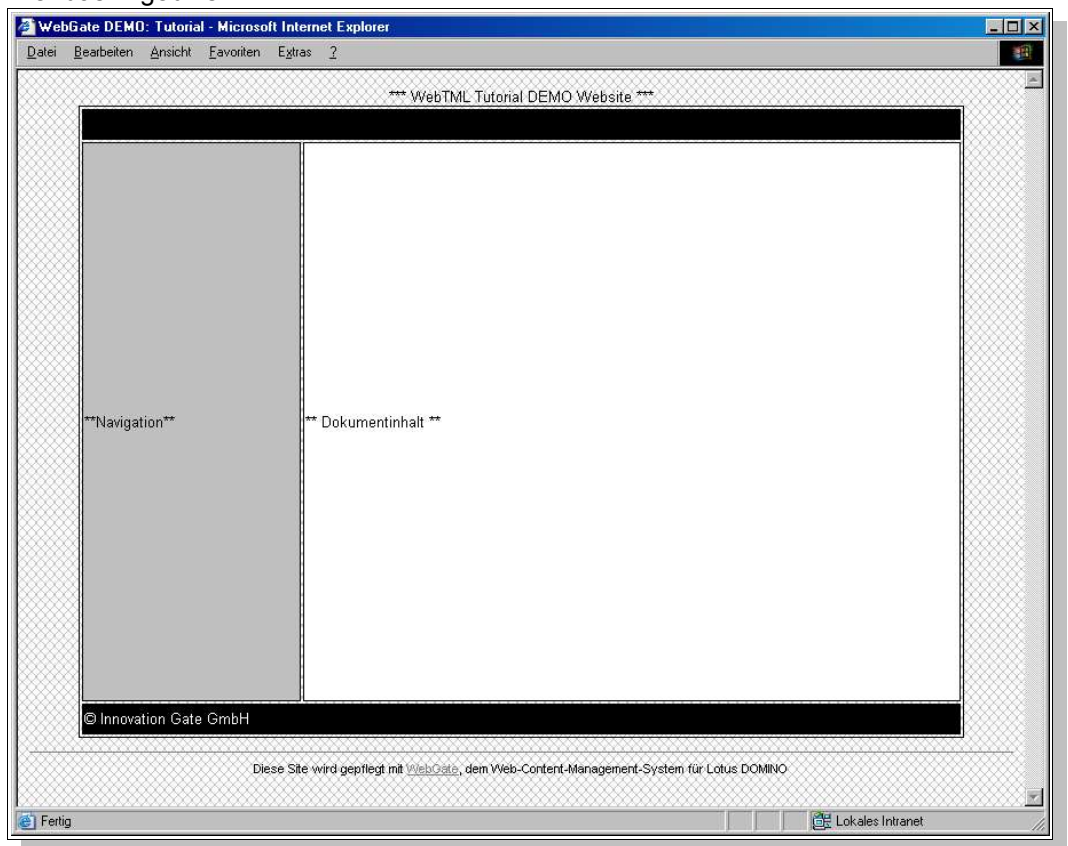
```

        </td>
</tr><tr>
  <td bgcolor="silver" width="25%">
    ** Navigation **
  </td>
  <td bgcolor="white" height="90%">
    ** Dokumentinhalt **
  </td>
</tr><tr>
  <td colspan="2" bgcolor="black">
    <font color="white">&copy; Innovation Gate GmbH</font>
  </td>
</tr>
</table>
</div>

```

Der Code enthält also zu diesem Zeitpunkt noch keinerlei WebTML-Tags.

Hier das Ergebnis:



## Zugriff auf Grafikressourcen: <tml:img>

Zunächst soll im Dokumentkopf das Firmenlogo angezeigt werden. Die entsprechende Grafikdatei soll im WebGate Anywhere Bereich Dateianhänge abgelegt sein, hier unter dem Namen "IG-Logo". Zum Zugriff auf eine Grafik in einem Dateianhang steht der WebTML-Tag <tml:img> zur Verfügung. Dieser muß mit einem name-Attribut eingegeben werden, der als Wert den Names des Dateianhanges enthält:

```
<tml:img name="IG-Logo" src="logo.gif"/>
```

Wir ersetzen also den Text "\*\*\*Dokumentkopf\*\*" durch den obigen WebTML-Tag:

```

<tr>
  <td colspan="2" bgcolor="black">
    <tml:img name="IG-Logo" src="logo.gif"/>
  </td>
</tr>

```

Hier das Ergebnis:



## Verwenden von WebTML-Bibliotheken: <tml:include>

Die nächste Aufgabe ist die Definition der Bereiche "Navigation" und "Dokumentinhalt", so dass links ein Navigator mit Verweisen (Links) auf alle "Hauptdokumente" dargestellt wird und rechts der jeweilige Dokumentinhalt.

Dazu nutzen wir die Möglichkeit aus, dass WebTML-Code in Bibliotheken verwaltet werden kann. Im konkreten Anwendungsfall wird dann der Bibliothekseintrag mit dem WebTML-Tag <tml:include> referenziert.

Wir erstellen zunächst einen Bibliothekseintrag für "Navigation" und referenzieren im Layout auf diesen Eintrag:

```
<div align="center">
*** WebTML Tutorial DEMO Website ***
<table border="1" bordercolor="black" width="90%" height="90%">
<tr>
  <td colspan="2" bgcolor="beige">
    <tml:img name="IG-Logo" src="Logo.gif"/>
  </td>
</tr><tr>
  <td bgcolor="silver" width="25%">
    <tml:include ref="Navigation"/>
  </td>
  <td bgcolor="white" height="90%">
    ** Dokumentinhalt **
  </td>
</tr><tr>
  <td colspan="2" bgcolor="black">
    <font color="white">&copy; Innovation Gate GmbH</font>
  </td>
</tr>
</table>
</div>
```

In diesem Code werden also an zwei Stellen WebTML-Tags verwendet: zum Darstellen einer Grafik aus den Dateianhängen und zum Verweisen auf einen WebTML-Bibliothekseintrag.

Als nächstes müssen wir den Code für den Bibliothekseintrag erstellen.

## Erstellen von Navigatoren: `<tml:siblings>`

Es sollen zunächst Links auf alle Hauptdokumente (Dokumente der obersten Strukturebene) der Website dargestellt werden.

In WebTML steht jeweils ein Tag zum Iterieren durch die Dokumentstruktur und ein Tag zum Darstellen des Links zur Verfügung. Der hier verwendete WebTML-Tag zum Navigieren durch die Dokumentstruktur heißt `<tml:siblings>`. Dieser Tag bietet die Möglichkeit, durch alle Dokumente der gleichen Ebene wie das aktuell dargestellte Dokument zu iterieren (siblings=Geschwister). Um sicherzustellen, dass nur Hauptdokumente dargestellt werden, fehlt noch der Zusatz `context="root"`. Dieser bewirkt, dass nicht die Geschwisterdokumente des **aktuellen** Dokumentes, sondern die Geschwisterdokumente des zum aktuellen Dokument gehörigen **Hauptdokumentes** in der Iteration verwendet werden. Der vollständige Tag lautet also `<tml:siblings context="root">`.

Wie in XHTML und in XML benötigt jeder WebTML-Tag einen entsprechenden End-Tag. Dieser lautet hier `</tml:siblings>`.

Jeder HTML- oder WebTML-Code, der zwischen dem `<tml:siblings>`Start- und dem End-Tag steht, wird so oft ausgegeben, wie es Hauptdokumente gibt.

Für die Darstellung eines Links kann der WebTML-Tag `<tml:link>` verwendet werden. Die Verwendung dieses Tags erübrigt es, die genaue URL der Dokumente zu kennen.

Hier der vollständige Code:

```
<tml:siblings context="root">
  <tml:link/><br>
</tml:siblings>
```

Beachten Sie, dass der `<tml:link>`-Tag sofort mit `/>` geschlossen wurde und dass dieser Tag keinen Code zwischen Start- und End-Tag benötigt.

Hier nun das Ergebnis:



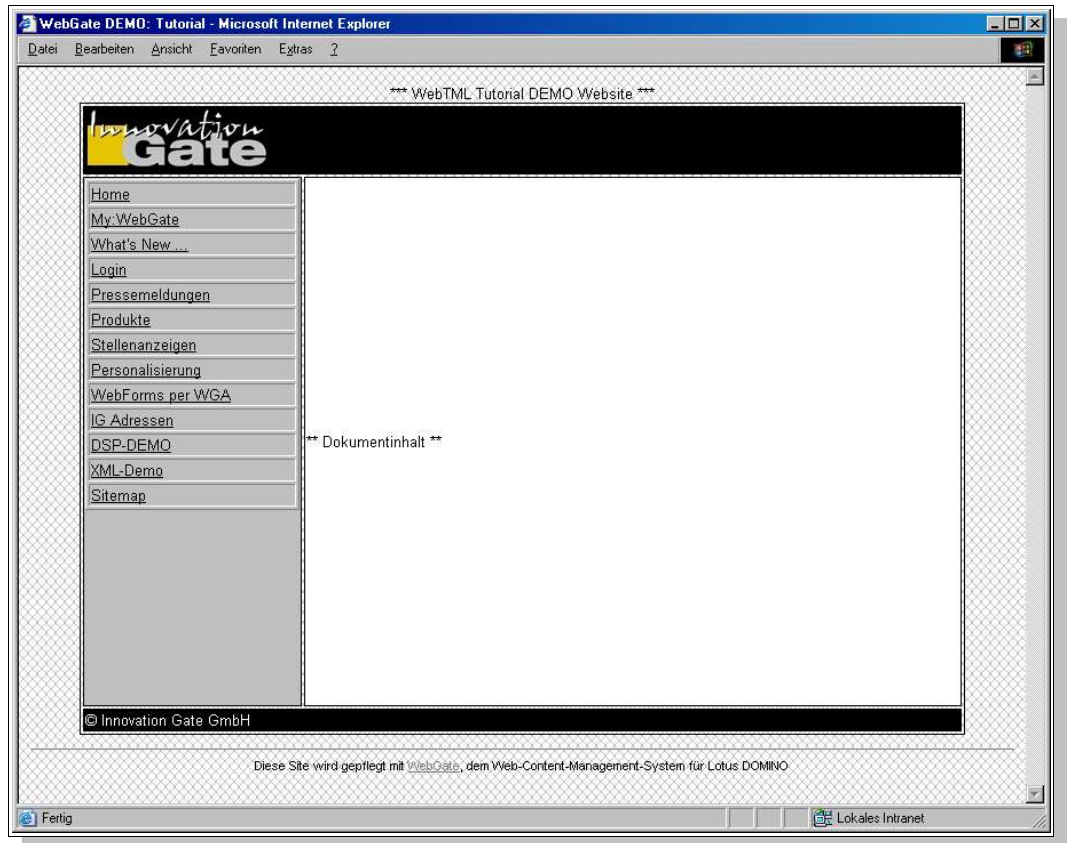
Die Links auf alle Hauptdokumente sind sichtbar. Das Layout entspricht aber noch nicht ganz unseren Vorgaben:

Zum einen sollte der Navigator oben in der linken Tabellenzelle dargestellt werden. Dies erreichen wir durch den Zusatz align="top" im <td>-Tag des Navigators.

Zum anderen sollen die Links in Form einer Tabelle dargestellt werden. Dazu verändern wir den WebTML-Code für den Navigator wie folgt:

```
<table border="1" width="100%">
  <tml:siblings context="root">
    <tr>
      <td align="top"><tml:link/></td>
    </tr>
  </tml:siblings>
</table>
```

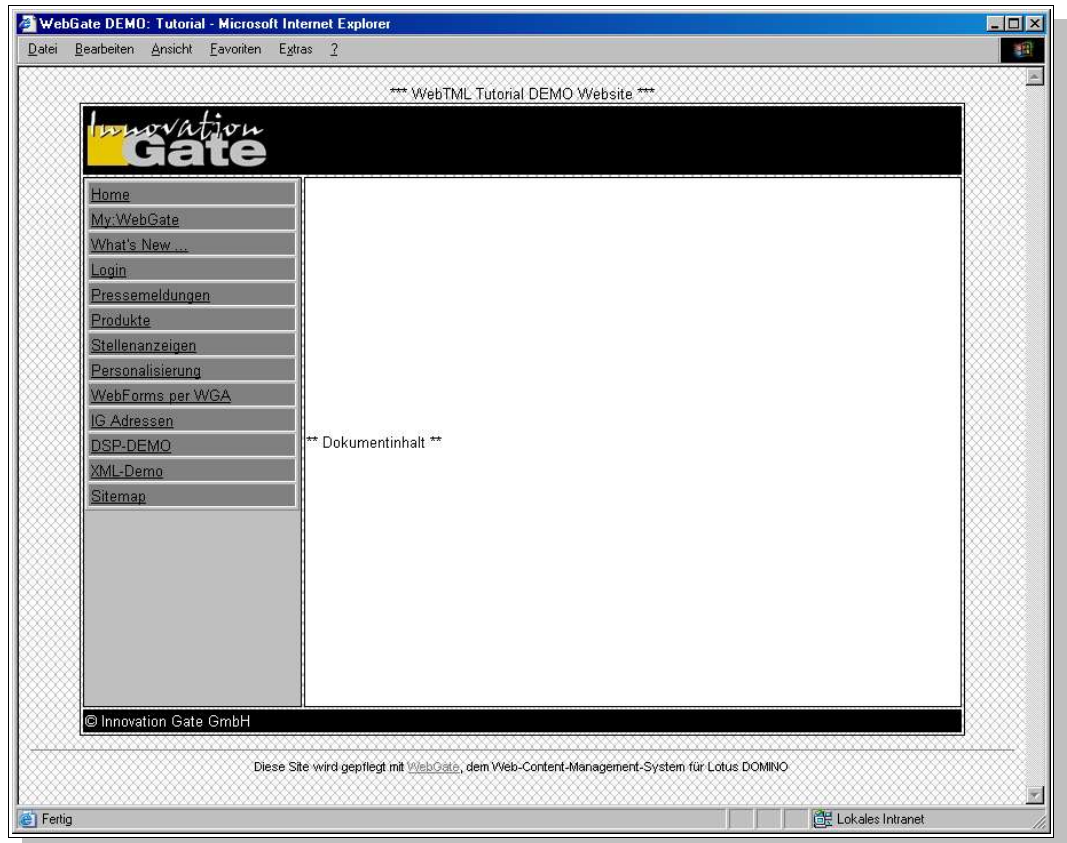
Hier das Ergebnis:



Das kommt unserem Ziel schon sehr nahe. Es fehlt noch die Einfärbung der Tabellenzellen in "Grau", z. B. durch die Angabe bgcolor="gray" im <td>-Tag im Code für den Navigator:

```
<table border="1" width="100%">
<tml:siblings context="root">
  <tr><td bgcolor="gray"><tml:link/></td></tr>
</tml:siblings>
</table>
```

Das Ergebnis:



## Bedingte Codeerzeugung: <tml:if>

Zur Orientierung auf einer Website ist es wichtig, dass der Leser weiß, in welchem "Kapitel" der Website er sich gerade befindet. In der WebGate Anywhere Terminologie bedeutet dies: unter welchem Hauptdokument sich das aktuelle Dokument befindet.

Unsere nächste Aufgabe wird also sein, denjenigen Hauptdokument-Link, unter dem sich das aktuelle Dokument befindet, mit einer anderen Farbe darzustellen, als die restlichen Links des Navigators. Dazu stehen in WebTML Tags zur Verfügung, die eine bedingte Ausführung ähnlich der if-Anweisung in anderen Programmiersprachen bewirken. Der benutzte WebTML-Tag heißt <tml:if>. Die allgemeine Syntax des <tml:if>-Tags lautet:

```
<tml:if condition="die Bedingung">
  <tml:then>
    dieser Code wird angezeigt, wenn die Bedingung erfüllt
    ist
  </tml:then>
  <tml:else>
    dieser Code wird angezeigt, wenn die Bedingung nicht
    erfüllt ist
  </tml:else>
</tml:if>
```

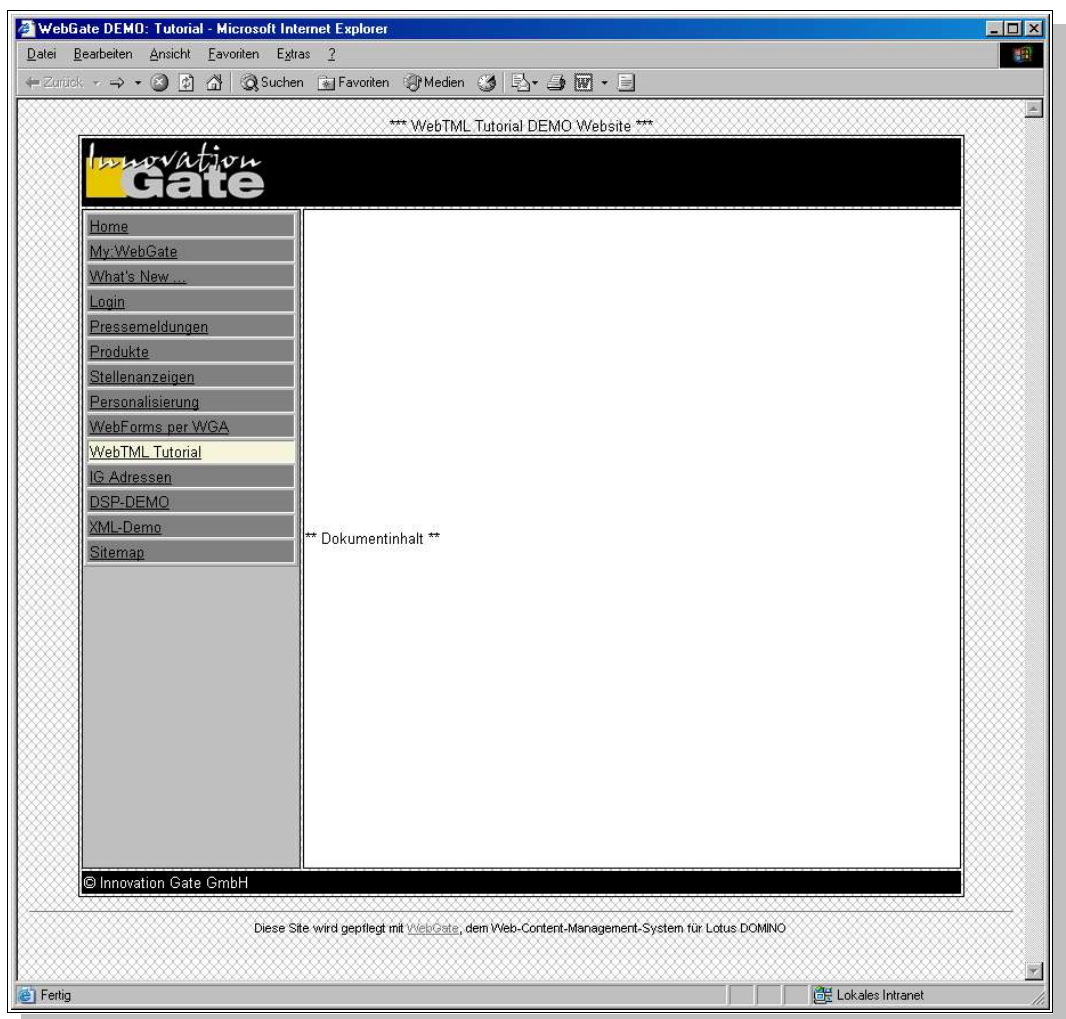
Für unseren Fall verwenden wir eine Spezialversion des Tags mit dem Attribut `isselected="True"`:

```
<tml:if isselected="True">
  <tml:then>
    dieser Code wird angezeigt, wenn sich das fragliche Dokument
    in der Hierarchie oberhalb des aktuellen Dokumentes
    befindet.
  </tml:then>
</tml:if>
```

Dieses Codefragment fügen wir in den Code unseres Navigators ein:

```
<table border="1" width="100%">
<tml:siblings context="root">
  <tr>
    <tml:if isselected="True">
      <tml:then>
        <td bgcolor="beige">
      </tml:then>
      <tml:else>
        <td bgcolor="gray">
      </tml:else>
    </tml:if>
    <tml:link/>
  </td>
</tr>
</tml:siblings>
</table>
```

Hier das Ergebnis:



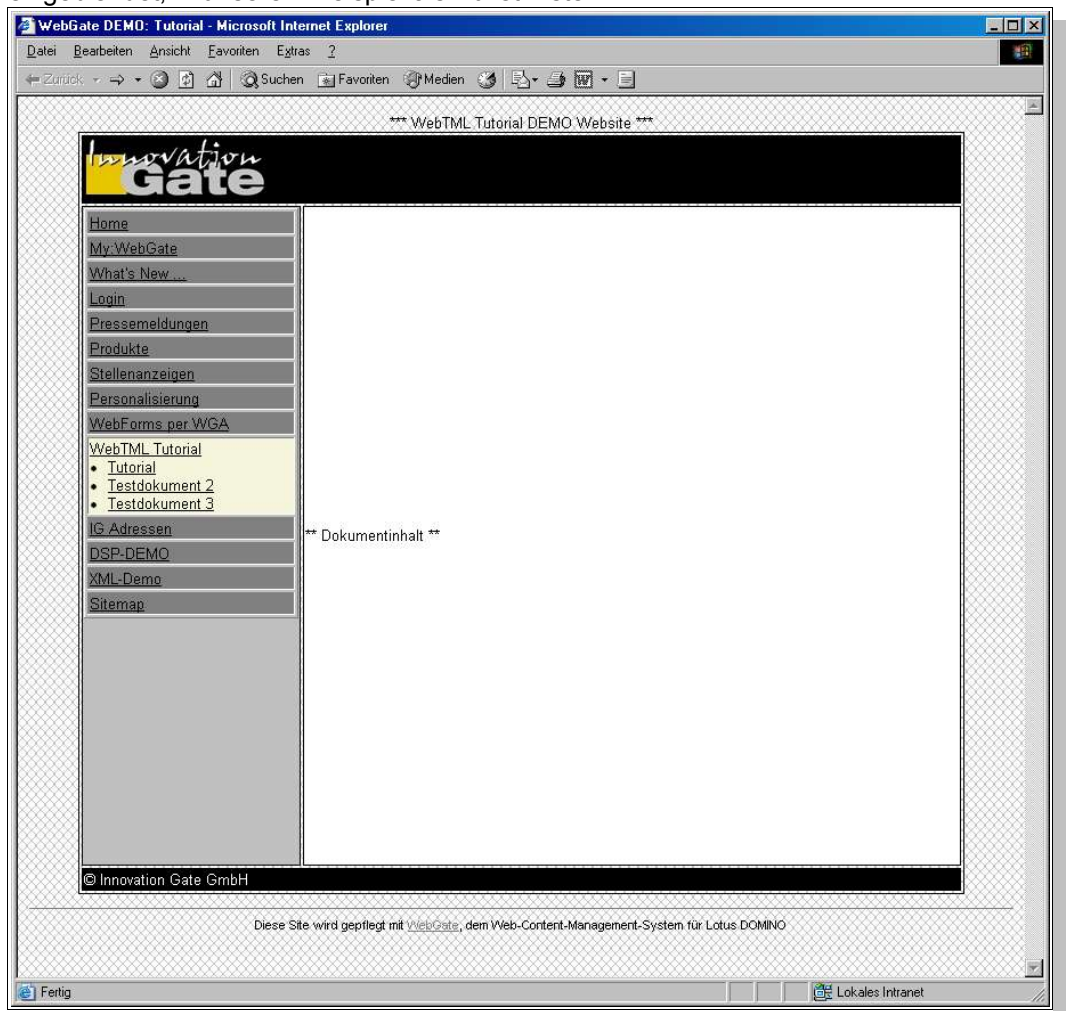
## Subnavigation: <tml:children>

Als nächstes möchten wir unterhalb des "aktiven Hauptdokumentes" die nächste Ebene der Navigation einblenden, also Links auf die Dokumente in der ersten Ebene unterhalb des aktuellen Hauptdokumentes. Dazu dient der WebTML-Tag <tml:children>, der in der gleichen Weise verwendet wird, wie der bereits verwendete WebTML-Tag <tml:siblings>. Im Unterschied zu diesem bietet <tml:children> die Möglichkeit, durch alle "Child-Dokumente" unterhalb des betreffenden Dokumentes zu iterieren.

Ein letztes mal modifizieren wir den Code des Navigators wie folgt:

```
<table border="1" width="100%">
<tml:siblings context="root">
  <tr>
    <tml:if isselected="True">
      <td bgcolor="beige">
        <tml:link/>
        <tml:children>
          <li><tml:link/></li>
        </tml:children>
      <tml:else>
        <td bgcolor="gray">
          <tml:link/>
        </tml:else>
      </tml:if>
    </td>
  </tr>
</tml:siblings>
</table>
```

Innerhalb der Iteration durch die Hauptdokumente im <tml:siblings>-Tag fügen wir den neuen <tml:children>-Tag ein, falls die Bedingung isselected="True" erfüllt ist. Dadurch wird unterhalb des aktuellen Hauptdokumentes eine weitere Ebene der Navigation eingeblendet, in unserem Beispiel als Bullet-Liste:



Da WebTML-Tags beliebig ineinander verschachtelt werden können, kann auf gleiche Weise auch eine dritte oder vierte Navigationsebene bis hin zur gesamten Sitemap eingeblendet werden.

## Zugriff auf Content-Felder: <tml:item>

Wir wenden uns nun dem Dokumentinhalt zu, der auf der rechten Seite unserer Webseite dargestellt werden soll.

Wie im Fall des Navigators, erstellen wir dazu einen Bibliothekseintrag für den Dokumentinhalt. Wir nennen diesen Eintrag "Dokumentinhalt" und referenzieren im Dokumentlayout darauf:

```
<div align="center">
*** WebTML Tutorial DEMO Website ***
<table border="1" bordercolor="black" width="90%" height="90%">
<tr>
  <td colspan="2" bgcolor="beige">
    <tml:img name="IG-Logo" src="logo.gif"/>
  </td>
</tr><tr>
  <td bgcolor="silver" width="25%">
    <tml:include name="Navigation"/>
  </td>
  <td bgcolor="white" height="90%" valign="top">
    <tml:include ref="Dokumentinhalt"/>
  </td>
</tr><tr>
  <td colspan="2" bgcolor="black">
    <font color="white">&copy; Innovation Gate GmbH</font>
  </td>
</tr>
</table>
</div>
```

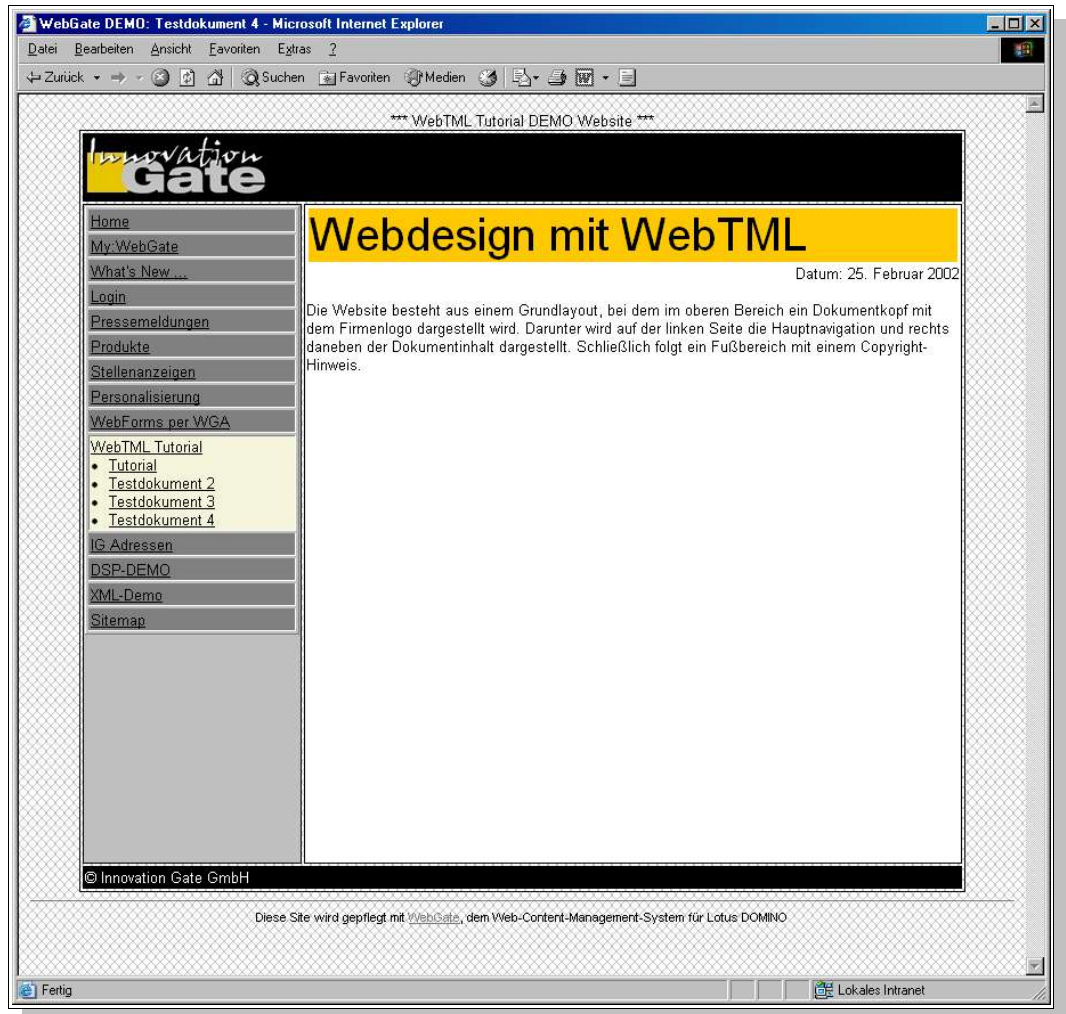
Typischerweise besteht der "Dokumentinhalt" aus Feldern, in denen die Autoren Inhalte (Content) eingegeben haben. In WebTML kann auf solche Content-Felder über den Tag `<tml:item>` zugegriffen werden. Der `<tml:item>`-Tag wird in der Regel mit dem Attribut `name="Name des Content-Feldes"` verwendet.

Nehmen wir an, unsere Dokumente bestehen aus Feldern `"_Headline"`, `"_Date"` und `"_Body"`, in denen die Autoren Inhalte erfaßt haben. Unser Bibliothekseintrag für "Dokumentinhalt" kann dann wie folgt aussehen:

```
<table width="100%">
<tr>
  <td bgcolor="#FFC800"><font size=18>
    <tml:item name="_Headline"/>
  </font></td>
</tr>
</table>

<div align=right>
Datum: <tml:item name="_date"/>
</div>
<br>
<tml:item name="_body"/>
<p>
```

Und hier das Ergebnis aus Sicht des Web-Users:



## Kollektionen in WebTML: <tml:collection>

Damit haben wir unser Ziel fast erreicht. Was noch fehlt, ist die Darstellung der aktuellen Pressemeldungen auf unserer Website. Dabei soll es egal sein, wo in der Dokumentstruktur die Pressemeldungen erfasst wurden.

Eine solche Darstellung von Inhalten, unabhängig von der Dokumentstruktur, heißt in WebGate Anywhere "Kollektion". Sie ist das Ergebnis einer Suchabfrage, die vom Website-Designer formuliert wurde.

Zur Formulierung von Kollektionen (Suchabfragen) dienen in WebTML die Tags <tml:collection>, <tml:query> und <tml:foreach>. Der <tml:query>-Tag wird dabei immer innerhalb des <tml:collection>-Tags verschachtelt:

```
<tml:collection>
  <tml:query>
    hier steht die Suchformel
  </tml:query>
  <tml:foreach>
    Hier wird durch das Ergebnis iteriert
  </tml:foreach>
</tml:collection>
```

Nehmen wir an, alle unsere Pressemeldungen haben ein (in der Regel berechnetes) Content-Feld "\_news", in dem immer der Wert "J" steht. Dann kann die Suchabfrage wie folgt formuliert werden:

```
<tml:collection>
  <tml:query>
    content.items["_news"].text="J"
  </tml:query>
  <tml:foreach>
```

```
Hier wird durch das Ergebnis iteriert
</tml:foreach>
</tml:search>
```

Die Formulierung der Query hängt dabei von der verwendeten Datenbank ab. Das obige Beispiel geht von der Verwendung einer SQL-Datenbank als ContentStore aus. In diesem Fall wird die Querysprache "HQL" (Hibernate Query Language) verwendet. Mehr zur Syntax und Verwendung von HQL finden Sie im WebGate Anywhere Administrationshandbuch.

Die Iteration durch das Suchergebnis geschieht in der gleichen Weise, wie die Iteration durch Geschwister-Dokumente in Navigatoren. Möchten wir beispielsweise nur die Überschriften aller Pressemeldungen als Bulletliste ausgeben, so können wir den folgenden WebTML-Code verwenden:

```
<ul>
<tml:collection>

  <tml:query>
    content.items["_news"].text="J"
  </tml:query>

  <tml:foreach>
    <li><tml:link/></li>
  </tml:foreach>

</tml:collection>
</ul>
```

Hier das Ergebnis:

WebGate DEMO: Testdokument 4 - Microsoft Internet Explorer

\*\*\* WebTML Tutorial DEMO Website \*\*\*

**Innovation Gate**

Home  
My WebGate  
What's New ...  
Login  
Pressemeldungen  
Produkte  
Stellenanzeigen  
Personalisierung  
WebForms per WGA  
WebTML Tutorial  
• Tutorial  
• Testdokument 2  
• Testdokument 3  
• Testdokument 4  
IG Adressen  
DSP-DEMO  
XML-Demo  
Sitemap

## Webdesign mit WebTML

Datum: 25. Februar 2002

Die Website besteht aus einem Grundlayout, bei dem im oberen Bereich ein Dokumentkopf mit dem Firmenlogo dargestellt wird. Darunter wird auf der linken Seite die Hauptnavigation und rechts daneben der Dokumentinhalt dargestellt. Schließlich folgt ein Fußbereich mit einem Copyright-Hinweis.

- [Aachener und Münchener](#)
- [Intranet der Deutschen BA startet Jungferflug](#)
- [WebShop Brings E-Commerce Solutions into the Future](#)
- [Dynamische WebSites mit Domino Server Pages®](#)
- [Content Management Software mit Shop-Lösung](#)
- [Webauftritt nach Maß bei Miele](#)
- [SULO bringt es an den Tag](#)
- [Web-Auftritte professionell managen](#)
- [Chemetall Aerospace Technologies setzt auf WebGate](#)
- [Problemlöser rund um Haus und Technik](#)
- [WebGate zeigt die Zähne](#)
- [Pflegen Sie Ihre WebSite doch womit Sie wollen!](#)

© Innovation Gate GmbH

Diese Site wird gepflegt mit [WebGate](#), dem Web-Content-Management-System für Lotus DOMINO

Wir haben dabei den Code unterhalb des WebTML-Codes für den Dokumentinhalt platziert. Hier noch einmal der komplette WebTML-Code für den Bibliothekseintrag

"Dokumentinhalt":

```
<table width="100%">
<tr>
  <td bgcolor="#FFC800"><font size=18>
    <tml:item name="_Headline"/>
  </font></td>
</tr>
</table>

<div align=right>
Datum: <tml:item name="_date"/>
</div>
<br>
<tml:item name="_body"/>
<p>
<ul>
<tml:collection>
  <tml:query>
    content.items["_news"].text="J"
  </tml:query>
  <tml:foreach>
    <li><tml:link/></li>
  </tml:foreach>
</tml:collection>
</ul>
```

## Zusammenfassung

WebTML bietet einfache Möglichkeiten, durch Dokumente der Content-Datenbank zu iterieren – entweder basierend auf der Dokumentenstruktur oder basierend auf dem Ergebnis einer Suchabfrage.

WebTML bietet einfachen Zugriff auf Content-Felder, Grafiken und Dokumentenlinks.

WebTML bietet dabei eine Umgebung, bei der ein Web-Designer alle Freiheiten zu Gestaltung der Website behält.

Neben den hier dargestellten Möglichkeiten bietet WebTML eine Vielzahl weiterer Features: Schleifenkonstrukte, Web-Formulare, Portalfunktionen, Zugriffsmöglichkeiten auf externe Datenbanken, bis hin zur Integration von WebServices (News, Wetterberichte, Auskünfte, etc.) oder Unternehmensdaten in den Webauftritt.

Die detaillierte WebTML-Syntax, alle WebTML-Tags und deren Attribute werden in den folgenden Kapiteln ausführlich beschrieben.