Firebird Docwriting-Anleitung

Paul Vinkenoog

21. Juli 2013 – Dokumentenversion 1.3-de - Deutsche Übersetzung

Übersetzung ins Deutsche: Martin Köditz

Inhaltsverzeichnis

Einleitung

Gegenstand dieses Dokuments

Dieser Leitfaden behandelt die verschiedenen Aspekte des Schreibens von Dokumentation für Firebird. Er ist gedacht für Menschen, die helfen, Dokumentation für das Firebird-Projekt zu schreiben, oder dies zumindest in Erwägung ziehen. Nach der Lektüre dieses Leitfadens, besitzen Sie alle notwendigen Kenntnisse, um mit dem Schreiben von Firebird-Dokumenten in unserem gewählten Format, DocBook XML, beginnen zu können.

Vorausgesetzte Kenntnisse

Bevor Sie dieses Handbuch lesen, prüfen Sie, ob Sie bereits folgendes wissen:

- Was das Firebird manual Modul ist.
- Was CVS ist und wie ein CVS-Client verwendet wird, um das aktuelle manual Modul herunterzuladen.
- Wie die aktuelle Firebird-Dokumentation mittels des heruntergeladenen manual Modul funktioniert.

Dieses Wissen ist wichtig, wenn Sie sich daranmachen wollen, Daten zu unserem Dokumentations-Projekt beizutragen. Wenn Sie sich unsicher an einem oder mehreren dieser Punkte fühlen, lesen Sie zunächst das Firebird Docbuilding Howto, und kehren Sie dann hierher zurück.

Themen dieses Handbuchs

Wir starten mit ein paar kurzen Kapiteln über:

- Die firebird-docs mailing list.
- Auswahl eines Themas.
- Erstellung eines Abrisses für Ihr Dokument.

Danach werden wir einige Zeit benötigen, um die Grundlagen der DocBook XML zu erörtern, denn das ist das Format, dass wir gern von Ihnen bekommen möchten. Dies bezieht folgendes ein:

- DocBook XML – was ist das?
- Gründe weshalb wir DocBook anderen Formaten vorziehen.
- Werkzeuge, die Sie verwenden können, um DocBook-Texte zu erstellen.

Nicht tragisch, wenn Ihnen DocBook bis jetzt nichts sagt: das benötigte Wissen können Sie innerhalb kürzester Zeit (weniger als eine Stunde) aneignen, und die Chancen, dass Sie hiervon bei anderen Projekten profitieren ist sehr hoch.

Der nächste Abschnitt behandelt das Schreiben der Dokumente selbst - das Docwriting:

- Erstellen des eigentlichen Dokuments.

- Nutzung der DocBook-Elemente.
- Ein, zwei Worte zu Sprache und Schreibstil.
- Copyrights und die Public Documentation License.

Abschließend, werden wir Ihnen zeigen, wie Sie Ihr fertiges Dokument zum Firebird-Projekt hinzufügen. Hauptthemen dieses Abschnitts sind:

- Commiting des fertiggestellten Dokuments zum manual Modul.
- Wo Sie die Commit-Rechte erhalten, wenn Sie noch keine besitzen sollten.
- Was man tut oder besser nicht tut, sobald man die Commit-Rechte hat.
- Veröffentlichung von HTML- und PDF-Versionen auf der Firebird-Webseite.

Wo treffen sich die Dokumentenersteller?

Die Homepage des Unterprojekts

Die Homepage des Dokumentations-Unterprojektes ist hier zu finden:

http://www.firebirdsql.org/index.php?op=devel&sub=doc

Es stellt Neuigkeiten über unsere Aktivitäten, Links zu unseren bereits veröffentlichten Dokumenten, Zukunftspläne, etc.

Die firebird-docs list

Wenn Sie ernsthaft daran interessiert sind, Dokumente für Firebird zu schreiben, sollten Sie sich in der Mailing-Liste einschreiben, wo wir unsere Pläne und Arbeiten diskutieren. Diese Liste ist für jeden offen und einschreiben kostet Sie nichts. Senden Sie eine E-Mail an:

```
<firebird-docs-request@lists.sourceforge.net>
```

mit den Worten „subscribe“ entweder als Betreff oder als die erste und einzige Zeile im E-Mail-Text. Alternativ können Sie auch das Formular auf der Webseite verwenden:

http://lists.sourceforge.net/lists/listinfo/firebird-docs

Welche Methode Sie auch verwenden, Sie erhalten innerhalb weniger Minuten eine E-Mail von der Liste. Folgen Sie den Anweisungen der E-Mail und schon stehen Sie mit auf der Liste.

Die Atkin news-Schnittstelle

Es gibt außerdem eine News-Schnittstelle für diese und andere Firebird-bezogenen Mailinglisten. Manchmal arbeitet diese etwas problematisch - oder gar nicht, obwohl das selten vorkommt. Ich empfehle, die Schnittstelle nicht für das Posten von Nachrichten zu verwenden, sondern zum Archivieren und Durchsuchen - dafür ist die

wirklich großartig. Wenn Sie alle vorliegenden Meldungen vom Server holen, haben Sie eine schöne Historie aus der Zeit vor Ihrer Teilnahme.

Um auf die Newsgroup zuzugreifen, verweisen Sie Ihren Newsreader auf:

news.atkin.com

und holen Sie sich die Grouplist. Schreiben Sie sich bei den gewünschten Groups ein. Beachten Sie, dass die Liste firebird-docs list auf sourceforge.firebird-doc (ohne s) auf dem Atkin News Server verweist.

Abhängig vom verwendeten Newsreader oder Browser, kann Sie auch folgender Link direkt zur Newsgroup führen:

news://news.atkin.com/sourceforge.firebird-doc

Sie können auch an die Newsgroup schreiben (posten), wenn Sie sich nicht in die Liste eingeschrieben haben. Jedoch muss die Freigabe für die Mailing list durch einen menschlichen Moderator erfolgen. Das bedeutet natürlich, dass die Veröffentlichung bis zu einem Tag (oder mehr Tage) verzögert wird.

Auswahl eines Themas

Diese Richtlinien sollen Ihnen helfen ein Thema zu finden, über das Sie schreiben können:

- Machen Sie sich zunächst klar, was es bereits gibt – niemand wartet auf drei Migrations-Anleitungen von MS-SQL zu Firebird.
- Fragen Sie sich selbst, was fehlt und was hilfreich für Firebird-Anwender im Allgemeinen oder im Speziellen sein kann.
- Fragen Sie sich außerdem worüber Sie *gern* schreiben möchten. Die logischste Wahl trifft auf ein Thema mit dem Sie vertraut sind, aber Sie können natürlich auch eines wählen, über das Sie mehr lernen möchten (das ist natürlich deutlich mehr Arbeit, aber eine großartige Lernerfahrung, sofern Sie bereit sind die Zeit zu investieren).
- Sie müssen nicht zwangsläufig ein ganzen Buch, eine riesige Anleitung oder Artikel schreiben. Vielleicht arbeiten bereits andere an einem größeren Dokument, zu dem Sie etwas beisteuern können. Vielleicht schreiben Sie ein oder mehrere Kapitel für ein Buch. Sie können aber auch Rohdokumente für ein Thema, über das Sie eine Menge wissen, bereitstellen.
- Sprechen Sie über Ihre Ideen - oder suchen Sie danach - in der firebird-docs list. Es kann passieren, dass der Antwortryhtmus der Liste sehr niedrig ist, aber seien Sie versichert, dass die Nachrichten gelesen *werden* und auch beantwortet.

Vorbereitung aufs Schreiben: Erstellung eines Abrisses!

Es ist immer eine gute Idee einen Abriss zu erstellen, bevor man anfängt zu schreiben. Dies hilft Ihnen sich zu „organisieren“. Außerdem verringert er das Risiko etwas wichtiges zu vergessen und es macht das Schreiben deutlich leichter.

Folgen Sie diesen Schritten, um einen Abriss zu erstellen:

- Definieren Sie genau, was Ihre Leser von Ihrem Werk lernen sollen.
- Teilen Sie das Thema in logische Einheiten auf - Kapitel und/oder Bereich bzw. Unterbereiche.
- Stellen Sie sicher, dass die Reihenfolge der Einheiten Sinn macht. Insbesondere für Howtos, Tutorials oder Anwenderhandbücher. Das heißt: Arrangieren Sie die Einheiten so, dass der Benutzer das was er als erstes tun oder lernen muss auch zuerst liest.
- Legen Sie den Abriss der Mailinglist firebird-docs auf sourceforge.net vor und fordern Sie Kommentare hierzu.

Sobald Sie zufrieden mit Ihrem Abriss sind, schauen Sie sich diesen nochmals genau an und entscheiden Sie, ob alle notwendigen (Roh-)Informationen vorhanden sind, die Sie zum Schreiben benötigen. Idealerweise sollten Sie diese vorliegen haben, bevor Sie beginnen, da manchmal ein fehlendes Stück Info ausreicht, Ihre gesamte Dokumentenstruktur zu überarbeiten. Also ist es besser diese Information schon vorher zu haben.

DocBook XML – Eine Einleitung

Das Format der Wahl für die Dokumentation innerhalb des Firebird manual Modul ist *DocBook XML*. Sollten Sie bisher nicht vertraut mit XML und/oder DocBook sein, folgen noch kurze Einleitungen in XML und DocBook. Beachten Sie, dass diese Einleitungen nur einen vereinfachtes Bild zeigen können. Aber das ist auch der Vorteil: Sie müssen kein DocBook-Experte sein um Firebird-Dokumente zu schreiben. Sie benötigen lediglich Basiswissen - welches Sie in innerhalb einer halben Stunde von den u.a. Absätzen erlernen können - und ein wenig Erfahrung bei der Zuweisung von DocBook XML-Tags zu Ihren Texten (diese werden Sie aber schnell während des Schreibens erlernen).

Überspringen Sie die allgemeinen XML-Grundlagen, wenn Sie bereits alles über XML-Element, -Tags, -Attribute, -Rendering und -Multichannel-Publishing wissen.

Überspringen Sie beide Grundlagen, wenn Sie auch schon Erfahrungen als DocBook-Author haben.

> **Anmerkung**
>
> Obwohl wir strikt empfehlen, dass Sie zumindest versuchen sollten, Ihre Dokumente im DocBook-Format abzugeben, akzeptieren wir natürlich auch, dass einige Leute nicht die Zeit haben sich einzuarbeiten (oder ihre existieren Dokus ins DocBook-Format zu konvertieren). Wenn dies auf Sie zutrifft, sprechen Sie dies bitte in der firebird-docs list an. Wir werden sicherlich keine nützliche Dokumentation ablehnen, weil sie im falschen Format vorliegt.

Sehr allgemeine XML-Grundlagen

XML steht für *Extensible Markup Language*, was vereinfacht ausgedrückt soviel bedeutet wie, Standardtext mit Markup-Tags versehen. Ein typisches XML-Text-Fragment könnte so aussehen:

```
<paragraph>
<loud>'No!'</loud> she screamed. <scary>But the bloody hand
<italics>kept on creeping</italics> towards her.</scary>
```

```
<picture file="bloody_hand.png"/>
</paragraph>
```

Tags und Attribute

Im obigen Beispiel werden die Wörter und Sätze in spitze Klammern eingeschlossen. Dies sind die Markup-Tags. `<italics>` ist ein *Start-Tag* , `</italics>` ist ein *End-Tag*, und `<picture file="bloody_hand.png"/>` ist ein alleinstehender Tag, offiziel *empty-element tag* genannt. XML-Tags werden immer wie folgt formatiert:

Tabelle 1. Format of XML tags

Tag-Typ	Beginnt mit	Endet mit
Start tag	`<`	`>`
End tag	`</`	`>`
Empty-element tag	`<`	`/>`

Immer noch unserem Beispiel folgend, sind `paragraph`, `loud`, `scary`, `italics` and `picture` *Tag-Namen*. Im Tag `<picture.../>` ist `file="bloody_hand.png"` ein *Attribut*, mit `file` als *Attributnamen* und `bloody_hand.png` als *Attributwert*. Attributwerte stehen immer in Anführungszeichen; einfache und doppelte sind beide erlaubt.

XML erlaubt Ihnen beliebige Tags zu definieren, solange Sie diese korrekt erstellen. Somit sind `<thistag>`, `<thattag>`, und `<this_is_not_a_tag/>` wohlgeformte (well-formed) XML-Tags. (XML welches dem Standard folgt, nennt man *wohlgeformte (engl. well-formed)*; Der Begriff *valid* wird nur in spezifisch definierten Implementationen verwendet – DocBook XML zum Beispiel).

Natürlich sollen die Tags selbst nicht im finalen Dokument erscheinen (das Dokument wird ja von den Lesern angeschaut). Vielmehr kümmern sie sich um die Ausgabe, die durch die Tags beeinflusst wird. XML, wenn es zum Schreiben von Dokumentationen Verwendung findet, ist ein typisches *Quellformat*, vorgesehen für die Verarbeitung durch Software, welche hübsch formatierte Ausgaben generiert. Die Ausgabe wird häufig als *Rendering* bezeichnet.

Einige Tags sind unmissverständliche Make-Up-Anweisungen:

```
<italics>kept on creeping</italics>
```

was natürlich bedeutet, dass die Wörter *kept on creeping* kursiv dargestellt werden sollen. Andererseits ist

```
<loud>'No!'</loud>
```

weniger offensichtlich zu verstehen. Soll das Wort `No!` in Fettdruck erscheinen? Oder unterstrichen? Oder wieder kursiv? Vielleicht soll dieser Text von einem Sprachprogramm laut vorgelesen werden, und das Tag `<loud>` weist es an die Stimme zu heben. Diese Dinge sind alle möglich, und noch mehr: Häufig wird ein einzelnes XML Quelldokument in verschiedene Ausgabeformate gewandelt – sagen wir, ein PDF-Dokument, eine HTML-Webseite und eine Audio-Datei. Dies wird *Multichannel-Publishing* (engl. multichannel publishing) genannt. Hiermit könnte `<loud>` im PDF als Fettdruck angezeigt werden; in der HTML-Seite wird es zu einem fetten, roten Text; und das Audioprogramm erhöht die Lautstärke um 50%.

Schauen wir uns die anderen Tags an. `<picture.../>` ist offensichtlich die Anweisung das Bild `bloody_hand.png` einzufügen, und `<scary>`, gut... das ist wieder weniger klar, genauso wie `<loud>`. Vielleicht soll der Satz zwischen `<scary>` zitternt mit Blutstropfen dargestellt werden. Vielleicht wird angsteinflößende Musik gespielt. Dies hängt alles von den Leuten ab, die die Tags definieren und der Software die diese Interpretiert, also das Rendering.

Abschließend haben wir noch das Tag `<paragraph>`, welches ein Strukturtag ist. Es erzählt uns etwas über den Platz, den die Zeilen innerhalb der internen Dokumenthierarchie einnehmen. Im endgültigen Dokument können Absätze mit Leerzeilen getrennt dargestellt werden. Aber nochmal, dies hängt von der Rendering-Software ab, und denkbar sind auch Benutzerkonfigurations-Einstellungen. Andere Strukturtags sind z.B. `<chapter>`, `<section>` und `<subdocument>`.

Sonderzeichen und Entitäten

Da das Zeichen „<“ eine spezielle Bedeutung als Start eines Tags besitzt, können Sie dies nicht direkt als literalen Wert (wörtlich) verwenden. Wenn Ihre Leser eine spitze Klammer sehen sollen, müssen Sie folgendes eintippen:

```
&lt;
```

Dies ist ein kaufmännisches Und, gefolgt von den Buchstaben `l` und `t` (*kleiner als*), gefolgt von einem Semikolon. Sie können außerdem `>` (*größer gleich*) als schließende spitze Klammer „>“ verwenden, müssen dies aber nicht.

XML hat viele dieser Codes; Sie heißen *entities*. Einige repräsentieren Zeichen, wie `<` und `&auml;` (ä, kleines a mit Umlaut) und andere dienen ganz andeen Zwecken. Aber alle beginnen mit einem kaufmännischen Und und enden mit einem Semikolon.

Aber einen Moment... wenn alle Entitäten mit einem kaufmännischen Und starten, wie packen Sie dann ein literales kaufmännisches Und in Ihren Text? Tja, dafür gibt es ebenfalls eine Entität:

```
&amp;
```

Somit wird diese Zeile XML:

```
Kernigan &amp; Ritchie chose '&lt;' as the less-than operator for C.
```

schlussendlich im Dokument zu:

```
Kernigan & Ritchie chose '<' as the less-than operator for C.
```

Und hier noch die gute Nachricht: wenn Sie einen guten XML-Editor verwenden, dann können Sie wahrscheinlich einfach „<“ und „&“ eintippen wo Sie diese auch immer als Literale verwenden möchten. Der Editor wird sicherstellen, dass sie als `<` und `&` im XML gespeichert werden. An späterer Stelle werden wir einige XML/DocBook-Editoren auflisten.

Elemente

Es gibt ein weiteres wichtiges XML-Konzept über das Sie bescheid wissen sollten: Das *Element*. Ein Element ist die Kombination aus Start-Tag, einem passenden End-Tag und das ganze dazwischen. Dieses „ganze dazwischen“ wird als Element-Inhalt (engl. *content*) bezeichnet, und es kann wiederum andere Elemente enthalten. Elemente werden nach ihren Tags benannt. Somit können wir über Absatzelemente, Kursivelemente, etc. sprechen.

> **Anmerkung**
>
> Elemente sind ein grundlegenderes Konzept als Tags. Tags sind bloß die Dinger, die Elemente identifizieren. Somit wäre es besser zu sagen, dass Tags nach ihren Elementen benannt werden. Aber da Tags leichter zu erkennen sind als ein ganzes Element, werde ich diese zuerst erläutern.

Das ist ein Element:

```
<loud>'No!'</loud>
```

Das ist auch ein Element:

```
<paragraph>This is an element containing <bold>another</bold>
  element!</paragraph>
```

Ein Leerelement-Tag (engl. empty-element tag) stellt das Element selbst dar. Solche Elemente haben natürlich keinen Inhalt, da sie kein *Tag-Paar* besitzen:

```
<picture file="bloody_hand.png"/>
```

> **Wichtig**
>
> Verwechseln Sie Inhalt nicht mit Attributen. Inhalt liegt *zwischen* den Tags, Attribute *innerhalb* der Tags. Das Leerelement des letzten Beispiels besaß ein Attribut, aber keinen Inhalt.

Ich überstrapaziere das Konzept der Elemente etwas, da die meisten Dokumentationen dazu tendieren von „Kapitelelementen“, „Titelelementen“, etc. zu sprechen. Richtiger wären aber „Kapitel-Tags“ und „Title-Tags“. Die Begriffe werden oft synonym verwendet, aber an manchen Stellen ist es wichtig die Unterschiede zu kennen.

XML Zusammenfassung

So – das war alles was Sie über XML wissen müssen. Sie sollten nun eine grundlegende Idee haben was XML ist, wie es aussieht, was und wofür Tags sowie Elemente sind. Wie ich schon sagte: Das Gesamtbild ist deutlich vereinfacht, aber für unsere Zwecke ausreichend.

Es sollte außerdem klar sein, dass reines Schreiben von selbsterstelltem XML relativ sinnlos ist, solange Sie keine Software besitzen, die *Ihre* Tags verstehen. Wie sonst können Sie Ihre XML-Quellen in ein hübsch formatiertes Dokument umwandeln?

Glücklicherweise müssen wir uns hierüber keine Sorgen machen. Es stehen uns bereits einige formalisierte XML-Typen zu Verfügung, wovon jedes einen Satz Tags und, genauso wichtig, einen Satz Regeln beinhaltet. Letztere sagen uns wie die Tags zu verwenden sind. DocBook XML ist einer dieser XML-Typen.

Eine DocBook XML Einleitung

DocBook wurde entworfen, um das Schreiben strukturierter Dokumente unter Verwendung von SGML oder XML zu verbessern (machen Sie sich keine Gedanken über SGML - wir nutzen den XML-Stamm). Dies betrifft insbesondere das Schreiben technischer Dokumente und Artikel, vor allem für Computer-relevante Themen. DocBook ist durch seine *Document Type Definition* bzw. *DTD* definiert: Ein Satz von Definitionen und Regeln, die genau beschreiben, wie ein gültiges DocBook-Dokument strukturiert ist. DocBook wurde schnell zum de

facto-Standard für Computer-technische Dokumente und wird durch eine wachsende Anzahl von Werkzeugen und Anwendungen unterstützt.

DocBook XML-Charakteristiken

Wichtige Eigenschaften von DocBook - im Gegensatz zum „allgmeinen“ XML – sind:

- Die DocBook DTD definiert eine begrenzte Anzahl Tags und gibt genaue Regeln aus, wie diese zu verwenden sind: Welche Attribute sind möglich für Tag A, ob Element B in Element C verschachtelt werden kann, u.s.w.. Wenn Sie undefinierte Tags benutzen oder den Regeln nicht folgen, ist Ihr Dokument auch kein DocBook mehr (und DocBook-unterstützende Anwendungen versagen möglicherweise Ihren Dienst).

- DocBook-Tags vermitteln immer die Struktur und Semantiken (Bedeutungen), *niemals* Aussehen. In DocBook werden Sie Struktur-Tags finden wie `<book>`, `<part>`, `<chapter>`, `<section>`, `<para>`, `<table>`; and semantic tags like `<filename>`, `<warning,>` `<emphasis>`, `<postcode>`; but nothing like `<font>`, `<bold>`, `<center>`, `<indent>`, `<backgroundcolor>` – nichts was mit dem Layout oder Aussehen zu tun hat.

- Deshalb muss irgendwo eine Entscheidung getroffen werden, wie die DocBook-Tags in ihr endgültiges Aussehen übersetzt werden sollen. Diese Entscheidung (oder besser: die Rendering-Regeln) können statisch in die Tools einprogrammiert werden, aber dies würde die Dinge sehr unflexibel gestalten. Darum sind die Regeln meistens in den *Stylesheets* definiert. Ein Stylesheet ist ein Dokument, welches dem Tool Zeug wie dieses erzählt:

  > „Zeige Kapitelüberschriften als 24 Punkte großer, schwarzer Schrift an; beginne jedes Kapitel auf einer neuen Seite; benutze kursiv für Betonungen; zeige Warnungen in fett und 12 Punkten an; benutze Großschreibung für Abkürzungen; etc., etc.“

  Dieser Ansatz ermöglicht es dem Benutzer die Stylesheets nach seinen Bedürfnissen zu verändern. Es würde deutlich aufwändiger sein - wenn nicht unmöglich - die Tools selbst anzupassen.

  > **Anmerkung**
  >
  > Stylesheets die benutzt werden, um DocBook XML in andere Formate zu wandeln, werden *transformation stylesheets* genannt. Sie werden in einer anderen Art XML geschrieben, *XSLT* (eXtensible Stylesheet Language for Transformations) genannt.

Vorteile von DocBook XML

DocBook hat eine Menge Vorteile für alle, die technische Dokumentationen erstellen. Dies sind die wichtigsten:

- Ein DocBook XML-Dokument besteht aus reinem, unverfälschtem *Inhalt*. Sie werden sich niemals Gedanken über das Aussehen machen müssen, während Sie schreiben. Sie können sich ganz auf die Struktur und Information konzentrieren. Diese Praxis mag Ihnen etwas altertümlich vorkommen, wenn Sie Text sonst in z.B. Word verfassen, aber ich verspreche Ihnen: Sie werden es lieben lernen.

- Da sich DocBook nur um die Struktur und Bedeutung kümmert, wird es überraschend einfach sein, Ihren kleinen Abriss in ein DocBook-Skelett zu konvertieren.

- Viele Leute erstellen Dokumente für das manual Modul. Wenn sie alle verschiedene Formate, oder gar ein einheitliches Format wie Word oder HTML verwendeten, würden Ihre Ergebnisse sehr unterschiedlich aus-

sehen, da jeder sein eigenes Aussehen präferiert. Natürlich könnten wir einen Satz Regeln hierfür erstellen, aber dann müsste jeder Schreiber sich dieser Regeln annehmen und diese auch noch selbst implementieren. Es ist also besser die Regelsätze an einem zentralen Ort zu halten: Die Stylesheets, und somit den Dokumenterstellern Zeit für ihre Dokumentation zu lassen. Sie müssen sich nicht um das Aussehen kümmern. Die Stylesheets werden sicherstellen, dass alle unsere Dokumentationen gleich Aussehen.

- Wenn wir das Aussehen unserer Dokumente nicht mögen, können wir dieses einfach ändern, wenn die Regeln hierfür in Stylesheets vorliegen. Nichts muss in den DocBook-Quellen selbst geändert werden; alles was wir zu tun haben, nachdem die Stylesheets angepasst wurden, ist die Dokumente neu zu rendern. Neu erstellte Dokus werden automatisch das neue Aussehen erhalten. Versuchen Sie das mal zu erreichen, wenn Sie Anweisungen für Aussehen und Styling in den Dokumenten verstreut sind!

- Ein weiterer Vorteil ist das DocBook ein offener Standard ist, nicht an kommierzielle Belange oder ein bestimmtes System gekoppelt. Wenn Sie das Firebird manual Modul herunterladen, können Sie die HTML- und PDF-Dokumente aus den DocBook-Quellen unter Linux und Windows erstellen - und wir können mehr Betriebssysteme unterstützen, wenn notwendig.

- Ein DocBook-Dokument ist reiner text, welcher ideal für die Nutzung im CVS ist. Ja, ein CVS-Baum kann natürlich auch Binärdateien enthalten, aber viele nützliche Eigenschaften, die CVS bietet (z.B. die Anzeige der Unterschiede zwischen zwei Versionen einer Datei) funktionieren nur mit Text.

Zugegebenermaßen, keiner der beschriebenen Vorteile gilt einzig und allein für DocBook. Aber DocBook hat sie alle. Und es wird weitreichend unterstützt. Das macht es zur perfekten Wahl für unsere Firebird-Dokumentationen.

DocBook-Dokumentationen im Internet

Hier sind ein paar Links, sollten Sie sich weiter über DocBook informieren wollen:

- http://opensource.bureau-cornavin.com/crash-course/

  Writing Documentation Using DocBook – A Crash Course von David Rugge, Mark Galassi und Eric Bischoff. Ein sehr schönes Tutorial, auch wenn viele der vorgestellten Werkzeuge nich von uns verwendet werden.

- http://docbook.org/tdg/en/

  DocBook – The Definitive Guide, von Norman Walsh und Leonard Muellner. Erwarten Sie nicht, ein anfängerfreundliches Tutorial zu finden - tatsächlis ist der erste Teil einschüchternt, wenn sie ein Anfänger sind. Der Grund weshalb ich dieses hier aufführe ist, dass es eine tolle Online-Referenz ist, welche ich oft beim Schreiben verwende.

- http://www.tldp.org/HOWTO/DocBook-Demystification-HOWTO/

  Das *DocBook Demystification Howto* ist interessant, wenn Sie etwas mehr über XML und DocBook erfahren wollen als wir Ihnen bisher erzählt haben. Es enthält außerdem einiges an Material über SGML und - nochmal - Werkzeuge, die wir nicht für das Firebird-Dokumentations-Unterprojekt verwenden.

- http://sourceforge.net/projects/docbook

  Das DocBook Open-Source-Projekt bei SourceForge.

Wenn Sie weitere gute Online-Resourcen kennen, lassen Sie uns dies bitte wissen. Schreiben Sie eine Nachricht an die firebird-docs list.

DocBook XML Schreibwerkzeuge

Text-Editoren

Da DocBook ein nicht-binäres Format ist, können Sie reine Texteditoren wie emacs, pico, Windows Notepad oder vi zum Schreiben nutzen. Und in der Tat machen einige genau das, weil sie hiermit volle Kontrolle über ihren Text haben und die handgeschriebenen Tags jederzeit sichtbar sind. Auf der anderen Seite können diese Text-Editoren keine *Validierung* Ihres DocBook-Dokuments durchführen: Sie bemerken Ihre Fehler erst während des Erstellvorgangs. Und die Struktur eines Dokuments - insbesondere große Dokumente - ist ebenfalls schwer im Textmodus zu analysieren. Etwas Abhilfe schafft hier die Einrückung mehrerer Ebenen.

Wenn Sie diesen Weg einschlagen wollen oder zumindest ausprobieren möchten, dann ist es eine gute Idee einen Editor zu wählen, der zumindest XML Syntax-Highlighting unterstützt. Ein guter Editor, der auch noch frei verfügbar ist, ist ConText, zu finden unter http://www.fixedsys.com/context/. Unglücklicherweise kann ConText nicht im UTF-8-Format speichern. Für US-ASCII-Dokumente stellt dies kein Problem dar (speichern Sie den Text als DOS oder Unix und alles ist gut), aber sobald Sie irgendwas oberhalb von ASCII 127 verwenden, wird ConText so gut wie nutzlos. Eine gute, freie Alternative ist SciTE unter http://scintilla.sourceforge.net/SciTEDownload.html. Es ist weniger intuitiv, aber sehr mächtig.

> **Warnung**
>
> Speichern Sie Dokumente, die keinen US-ASCII-Inhalt besitzen, niemals in 8 Bit, weder in ConText noch einem anderen Editor! Alles was über US ASCII hinaus geht, muss in einer Unicode-Kodierung, wie UTF-8 (für die meisten Sprachen) oder UTF-16 (für einige Sprachen, falls die UTF-16-Dateilänge geringer oder zumindest nicht viel größer als UTF-8 ist), gespeichert werden. Die Kodierungsprobleme stellen weitere gute Gründe dar, einen XML-Editor zu nutzen: Sie werden die Daten normalerweise automatisch in der korrekten Codierung speichern.

XML-Editoren

Dedizierte XML-Editoren haben häufig grafische Benutzeroberflächen, die Tags schön (und manchmal irritierend) hervorheben; viele erlauben es Ihnen die Elemente ein- und auszuklappen, so dass Sie die Struktur Ihres Dokumentes sehen und gleichzeitig in das aktuelle Element hineinzoomen können. Die meisten können Ihr Dokument gegen das DocBook DTD validieren und einige haben einen speziellen DocBook-Bearbeitungsmodus, der es Ihnen ermöglicht, fast wie in einer Textverarbeitung zu schreiben.

Der Autor dieser Anleitung hat einige der Werkzeuge ausprobiert (freie, günstige und Evaluierungsversionen) und empfand XMLMind XML Editor als am nützlichsten. Das ist eine persönliche Einschätzung; Ihre Erfahrung kann natürlich ganz anders aussehen.

Einige XML-Editoren die Sie versuchen sollten:

- XMLMind XML Editor, kurz XXE. Die Standard-Edition ist kostenfrei.

  http://www.xmlmind.com/xmleditor/

  Läuft unter: Linux, Windows, Mac OS X. Benötigt Java, aber Sie benötigen Java eh, da Sie die Dokumente sonst nicht aus den Quellen erstellen können – mehr unter Firebird Docbuilding Howto.

Features: Baumansicht (alle Elemente einklappbar) und gestylte Ansichten (Kapitel und Bereiche klappbar). Ich arbeite normalerweise mit letzterem: Die Ansicht zeigt das Dokument in grundlegendem, aber klarem Textverarbeitungsähnlichen Layout, das in einem Stylesheet definiert ist, dass mit der Anwendung ausgeliefert wird. Beide Ansichten können gleichzeitig aktiviert werden. Der DocBook-Modus lässt Sie nur DocBook-Anweisungen einfügen. Element-Wähler. Attribut-Editor. Suchen und ersetzen. Rechtschreibprüfung. Sonderzeichenauswahl. Schnellzugriffsfunktionen um häufig verwendete Elemente, wie sections, lists, tables, etc., aufzurufen. Was ich vermisse ist eine Klartextansicht des XML-Quellcodes.

- Oxygen XML Editor. $ 48 für nichtkommerzielle Zwecke. Freie 30-Tage-Testversion.

  http://www.oxygenxml.com

  Läuft unter: Windows, Mac OS X, Linux, Eclipse. Benötigt Java.

  Features: XML-Quellcode-Editor. Baumansicht. Attribut-Editor. Gliederungsbereich. Tooltips für DocBook-Tags. XSLT-Debugger (ein leistungsfähiges Tool, irrelevant für das Schreiben, aber großartig, wenn Sie mit unseren Transformations-Stylesheets arbeiten wollen). Validierung, Refactoring, Rechtschreibprüfung, etc.. Ein sehr guter XML-Editor.

- epcEdit. € 89 für nichtkommerzielle Nutzung. 60-Tage Probierversion.

  http://www.epcedit.com

  Läuft unter: Linux, Windows, Solaris. Benötigt Tcl/Tk 8.1 oder höher (im Paket enthalten).

  Features: Struktur-Baumansicht. Element-Wähler. Attributeditor. Documentbereich kann zwischen Klartext und grafischen XML-Modus umschalten. Kein spezieller DocBook-Modus, kann aber jedes XML-Dokument validieren, dass auf DTD basiert.

- Altova XMLSpy. The Home Edition ist mittlerweile frei.

  http://www.altova.com/products_ide.html

  Läuft unter: Windows, Eclipse. (Läuft angeblich auch mit Wine unter Linux und auf Mac OS X mittels Virtual PC 6.)

  Features: Text- und Browseransichten. Alle Elemente klappbar in der Browseransicht. Browseransicht nur lesbar. Elementwähler. Bearbeitungs- und Suchfunktionen. Attributwähler. Sonderzeichenauswahl.

  Es gibt eine Feature-Matrix, die die Versionen Home, Professional und Enterprise miteinander vergleicht http://www.altova.com/matrix_x.html.

Diese Liste erhebt keinen Anspruch auf Vollständigkeit, aber wenn Sie einen *guten* XML-Editor kennen (gut aus der Perspektive eines Firebird-Dokumentenschreibers), der hier fehlt, lassen Sie es uns über die Mailingliste wissen.

Bereiten Sie Ihr DocBook-Dokument vor

Hallo – Sie sind ja noch da! Ich weiss, ich habe viel Zeit darauf verwendet, XML und DocBook zu erklären, aber ich hatte wirklich das Gefühl, dies tun zu müssen, da dies für viele Menschen neue Konzepte sind. Einfach nur ein paar Links in den Raum zu werfen und Sie dann allein zu lassen, würde vermutlich dazu führen, dass uns ein paar brauchbare Schreiber verloren gingen.

Egal, nun sind wir hier: Schlussendlich bereit loszulegen. Schreiben wir unser Dokument. Dieser Abschnitt klärt die Einrichtung Ihres DocBooks; der nächste zeigt wie die korrekten Tags und Attribute an den richtigen Stellen gesetzt werden.

Erstellung des Dokuments

Jedes Stück der Dokumentation in unserem manual Modul ist Teil eines Dokumentensatzes, engl. `<set>`. Dies ist das allererste Element in der the DocBook-Hierarchie. Ein Dokumentensatz enthält eine Anzahl an Büchern (`<book>`s), welche wiederum Kapitel (`<chapter>`s) enthalten, und so weiter.

Ein Vorteil Bücher in einem Dokumentensatzes zu platzieren, ist, dass Sie diese untereinander referenzieren können. Das heißt, Sie könnten beispielsweise Links einfügen, die auf einen Bereich eines anderen Buchs verweisen. Dieser Vorteil wird dadurch relativiert, dass diese Links nicht im PDF funktionieren (im Gegensatz zu HTML). Ein anderer Vorteil ist die automatische Erstellung eines Inhaltsverzeichnisses.

Glücklicherweise bedeutet die Platzierung von Büchern im gleichen Satz nicht, dass diese alle in der gleichen, großen Datei liegen müssen. DocBook erlaubt es Ihnen, ein Hauptdokument, wie im Anschluss gezeigt, anzulegen. (Machen Sie sich keine Gedanken über den Bereich, der mit "`<!DOCTYPE`" startet – Sie müssen keines dieser fürchterlichen Dinge selbst schreiben. Im schlimmsten Fall müssen Sie diesen Teil nur kopieren und anpassen, wenn Sie einen existierenden Dokumentensatz übersetzen.)

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE set PUBLIC "-//OASIS//DTD DocBook XML V4.1.2//EN"
  "docbookx/docbookx.dtd" [
    <!ENTITY preface SYSTEM "firebirddocs/preface.xml">
    <!ENTITY fb-intro SYSTEM "firebirddocs/firebirdintro.xml">
    <!ENTITY ...>
    <!ENTITY ...>
]>

<set id="firebird-books">
  &preface;
  &fb-intro;
  ...
  ...
</set>
```

Mit dem Einrichten des Hauptdokumentes, wie gerade beschrieben, werden die Bücher in je eigenen Dateien vorgehalten: `preface.xml`, `firebirdintro.xml`, etc.. Diese können nun unabhängig voneinander bearbeitet werden. Eine solche Datei - Ihre zum Beispiel - ist allgemein so strukturiert:

```
<?xml version="1.0" encoding="UTF-8"?>

<book id="fbintro">
  <chapter id="fbintro-preface">
    ...
    ...
  </chapter>
  <chapter id="fbintro-installing-firebird">
    ...
    ...
  </chapter>
  ...
  ...
</book>
```

Natürlich muss das neue Dokument dem Haupt-Dokumentensatz bekannt gemacht werden, aber das ist etwas worüber wir mit Ihnen sprechen werden, wenn Sie mit dem Schreiben starten. (Wir geben hier keine allgemeine Richtlinie aus, da dies davon abhängt, was Sie schreiben wollen - ein Buch, Artikel, ein oder mehrere Kapitel... - und Ihr Werk soll ja zum Rest passen.)

Jede DocBook-Datei muss mit der folgenden Zeile beginnen:

```
<?xml version="1.0" encoding="UTF-8"?>
```

(Hinweis: für einige Sprachen wird UTF-16 eine bessere Wahl sein.)

Wenn Sie Ihre Dokumentation „händisch“ schreiben, z.B. mit einem Texteditor, müssen Sie diese Zeile selbst hinterlegen. Wenn Sie einen speziellen XML-Editor verwenden, wird dies automatisch eingefügt, sobald Sie ein neues Dokument erstellen.

Dateipfade für verschiedene Dokumentensätze

Dateien für das englische Benutzerdokumentations-set liegen im Verzeichnis `manual/src/docs/firebird-docs`. Nicht-englische Dokumente werden im sprachabhängigen Baum wie z.B. `manual/src/docs/firebirddocs-fr`, `manual/src/docs/firebirddocs-es`, etc. abgelegt.

Seit Januar 2006 haben wir die Möglichkeit weitere Basis-Dokumentensätze anzulegen, das erste war `rlsnotes`, der Satz für Release Notes. Die gleiche Logik finden wir auch hier wieder: Englisches Zeug für die Release Notes liegt unter `manual/src/docs/rlsnotes`, französisches in `manual/src/docs/rlsnotes-fr`, und so weiter.

Jeder dieser Verzeichnisbäume – `firebirddocs`, `firebirddocs-es`, `firebirddocs-nl`, `rlsnotes`, `rlsnotes-fr`, etc. – beinhaltet ein eigenes `<set>`, mit einem Hauptdokument und eine beliebige Anzahl Dateien.

Texte schreiben

Wenn Sie Ihr DocBook-XML in einem Texteditor wie Notepad, emacs oder ConText bearbeiten, können Sie Zeilenumbrüche, Einrückungen und mehrere Leerzeichen verwenden, wenn Sie möchten. Jedes Auftreten eines Leerraums (engl. *whitespace*, eine Sequenz von einem oder mehreren `Leerzeichen`, `Tabs`, `Zeilenvorschüben` oder `Steuerzeichen`) wird in ein einzelnes Leerzeichen in der Ausgabe umgewandelt. So wird aus:

```
<section><title>Firebird Architectures</title><para>Now let's have a
look at Firebird's different architectures.</para><itemizedlist>
<listitem><para>First, there's the so-called <firstterm>Classic Server
</firstterm>.</para></listitem><listitem><para>Then there is <firstterm>
Superserver</firstterm> architecture.</para></listitem><listitem><para>
And finally, with the release of Firebird 1.5 we also have the
<firstterm>embedded server</firstterm>.</para></listitem></itemizedlist>
</section>
```

diese Ausgabe:

```
<section>
  <title>Firebird Architectures</title>
  <para>Now let's have a look at Firebird's different
    architectures.</para>
  <itemizedlist>
    <listitem>
      <para>First, there's the so-called
```

```
        <firstterm>Classic Server</firstterm>.</para>
    </listitem>
    <listitem>
      <para>Then there is <firstterm>Superserver</firstterm>
        architecture.</para>
    </listitem>
    <listitem>
      <para>And finally, with the release of Firebird 1.5 we also
        have the <firstterm>embedded server</firstterm>.</para>
    </listitem>
  </itemizedlist>
</section>
```

Nicht nötig zu sagen, dass die zweite Variante deutlich leichter zu lesen und zu verstehen ist. Wenn Sie also Ihr XML mit der Hand schreiben sollten, formatieren Sie den Text so, dass die Struktur so klar wie möglich ist.

Wenn Sie einen speziellen XML-Editor verwenden, beachten Sie bitte, dass **Enter** möglicherweise automatisch das aktuelle `<para>`-Tag schließt und ein neues öffnet. Stellen Sie sicher, dass Sie wissen, wie sich Ihr Editor in diesem Bezug verhält. Prüfen Sie außerdem wie sich der Editor bei zu vielen Leerzeichen verhält. Manche nutzen spezielle Tricks um diese zu behalten.

Häufig verwendete Elemente

Dieser Abschnitt behandelt die am häufigsten verwendeten DocBook-Elemente. Er beinhaltet viele Beispiele im DocBook XML-Format. Wenn Sie einen XML-Editor verwenden, wird die Ausgabe vermutlich nichts mit diesen Beispielen zu tun haben. Öffnen Sie Ihre Datei hingegen in einem Texteditor - oder wählen die reine Textansicht in Ihrem XML-Editor - werden Sie den aktuellen XML-Code sehen. Sehen Sie auch die vorhandenen XML-Quellen des manual Modul ein, um herauszufinden wie andere Autoren ihre Dokumente zusammenbauen und Tags zuweisen.

Lesen Sie bitte den Unterabschnitt über hierarchische Elementstrukturen, auch wenn Sie ein professioneller DocBook-Schreiber sind, da es ein paar spezifische Projektvorgaben enthält. Danach können Sie den Rest der DocBook-Unterabschnitte überspringen.

Wenn Sie erstmalig mit DocBook arbeiten, seien Sie nicht von der Länge dieses Abschnitts entmutigt. Mein Rat ist, dass Sie den Abschnitt über hierarchische Elemente *gründlich* lesen und den Rest überfliegen. Es ist kein Problem, wenn Sie nicht gleich alles verstehen. Das Verständnis kommt bei der Anwendung. Haben Sie diesen Leitfaden nur zur Hand, wenn Sie ihr Dokument schreiben, und schauen Sie immer mal wieder in die Elementabschnitte rein (so wie sie es gerade brauchen).

Hierarchische Elemente

Die allgemeinste Hierarchie, startet mit: `<set>` – `<book>` – `<chapter>` – `<section>` – `<para>`. Ein Buch (book) besteht möglicherweise aus Artikeln (`<article>`s) anstatt aus Kapiteln (`<chapter>`s).

Der nächste Unterabschnitt wird einige Aspekte bezüglich der Dokumentstruktur erklären.

Das `id`-Attribut

Sets, books, chapters, articles und Top-Level-Abschnitte sollten immer ein `id`-Attribut besitzen. Andere Elemente können ebenfalls eines haben. Die ID erlaubt es uns ein Element von anderen Stellen des Dokuments

zu referenzieren, sogar aus anderen Dokumenten innerhalb des Satzes (set). IDs sind nicht sichtbar im gerenderten Dokument (mit Ausnahme des HTML-Quelltextes), aber sie werden verwendet für die Namensvergabe der HTML-Dateien.

Alle `id`-Attribute müssen eindeutig innerhalb des Buchsatzes (bookset). Beachten Sie, dass die verschiedenen Sprachversionen in je einem eigenen `set` liegen, wodurch es OK ist, die originalen `id`s in der Übersetzung zu behalten.

Innerhalb eines Buchs (book) oder Artikels (article), sollten die `id`s mit den gleichen kleingeschriebenen Wörtern beginnen, z.B. `usersguide`, gefolgt von einem Trennzeichen, gefolgt von einem oder mehreren kleingeschriebenen Wörtern. Beispiele hierfür sind `usersguide-intro` und `usersguide-download-install`. Dies ist keine DocBook-Voraussetzung, sondern unsere eigene Konvention.

Die `lang`-Attribute in nicht-englischen Dokumentensätzen

Wenn Sie einen neuen Dokumentensatz erstellen, oder einen vorhandenen übersetzen, müssen Sie das Sprachattribut `lang` im Hauptelement verwenden:

```
<set id="firebird-books-fr" lang="fr">
```

Hiermit wird sichergestellt, dass die korrekten Beschriftungen für Hinwise, Warnung, etc. erstellt werden und dass die sprachabhängigen Anführungszeichen verwendet werden. Es ist außerdem eine gute Praxis, diese Attribute für die individuell gestalteten Dokumente zu verwenden, wenn diese jemals außerhalb Ihres Sets erstellt werden.

Für englische Dokumentensätze ist das `lang`-Attribut optional.

Titles

set, book, chapter, article und section müssen immer ein `title`-Attribut besitzen – entweder als direktes Kindelement oder innerhalb eines *xxx*`info`-Elementes (siehe unten). Es ist außerdem möglich diesen in beiden Elementen anzugeben, in diesem Fall jedoch *müssen* die zwei `title` identisch sein, unabhängig davon, ob `id` ein Attribut oder `title` ein Element ist. Und auch unabhängig davon, ob `title` in der Ausgabe erscheinen wird.

Wenn `title` zu lang ist, sollten Sie ein `titleabbrev`-Element direkt dahinter hinzufügen. Dieses enthält eine gekürzte Fassung des Titels. Der Hauptgrund hierfür ist, dass jede erstellt HTML-Seite eine sogenannte Hierarchieleiste enthält. So eine Art „Sie befinden sich hier-Zeile“ am Beginn und Ende. Diese Hierarchieleiste zeigt Ihnen jede Abstufung vom höchsten Element (dem `set`) bis runter zu Ihrer aktuellen Seite. Da die Namen auch anklickbar sind, gibt Ihnen die Leiste jederzeit bekannt, wo sich sich gerade innherhalb der Hierarchie befinden. Außerdem ist die Navigation zu höheren Elemente hierdurch kinderleicht. Am besten sieht die Leiste aus, wenn die Namen von der Größe her in eine Zeile passen, gleiches gilt für `titleabbrev`, welches nur angezeigt wird, wenn es auch vorhanden ist. Ist dies nicht der Fall wird `title` verwendet. Das gleiche Schema setzt sich für die Lesezeichen (die Navigation im linken Bereich) im PDF-Dokument fort.

Info-Elemente

Wenn Sie ein Buch oder Artikel verfassen, müssen Sie ein `bookinfo`- oder `articleinfo`-Element am the Start definieren. Innerhalb dieser können Sie Autoreninformationen und mehr erstellen. Weitere *xxx*`info`-Elemente existieren, aber Sie werden diese selten benötigen.

```
<book id='usersguide' lang='en'>
  <bookinfo>
```

```
    <title>Firebird Users Guide</title>
    <author>
      <firstname>William</firstname>
      <surname>Shakespeare</surname>
    </author>
    <edition>25 January 2006 – Document version 1.2</edition>
  </bookinfo>
  ...
  ...
</book>
```

Wenn der Autor einer Firma oder anderen Organisation oder Gruppe angehört, auf die Sie verweisen möchten, nutzen Sie `corpauthor` anstelle von `author`:

```
<corpauthor>IBPhoenix Editors</corpauthor>
```

Gibt es mehrere Autoren und Sie möchten diese einzeln aufführen, erstellen Sie ein `author` (oder `corpauthor`)-Element für jeden von ihnen und fassen Sie diese in einem `authorgroup`-Element zusammen – alles innerhalb des *xxx*`info`-Elements.

Abschnittsarten

Abschnittselemente unterscheiden sich etwas vom Rest in zwei Arten:

- Erstens, das `<section>`-Element wie zuvor beschrieben. Es kann rekursiv genutzt werden, beispielsweise könnten Sie ein `<section>` innerhalb eines anderen `<section>` wiederum innerhalb eines anderen `<section>`... verschachteln. Dies hat den Vorteil, dass Sie den gesamten Unterbaum hoch und runter wandern können, ohne die Tags zu ändern.

- Zweitens gibt es den `<sect1>`, `<sect2>` ... `<sect5>`-Bereich. Diese Elemente werden üblicherweise verschachtelt, mit `<sect1>` am Anfang, `<sect2>` innerhalb von `<sect1>` etc. Sie können `<sect3>` nicht direkt in `<sect1>` einbinden. Das ist weniger flexibel als `<section>` und in der Praxis auch nervenaufreibend. Nichtsdestotrotz trifft die gleiche „Rigidität“ auf die Elemente `<set>`, `<book>` und `<chapter>` zu und wir können auch damit leben.

> **Anmerkung**
>
> In früheren Versionen diese Leitfadens, wurde die `<sect`*N*`>`-Methode bevorzugt. Aufgrund von Verbesserungen der Stylesheets ist die jedoch nicht mehr länger der Fall. Nutzen Sie was immer Sie möchten.

Anhänge

Sie können einen oder mehrere `appendix`-Elemente nach dem letzten Kapitel eines Buches (book) oder nach dem letzten Abschnitt (section) eines Artikels (article) einfügen. Anhänge können alles beinhalten, was auch `section` beinhalten darf, somit auch weitere Abschnitte.

Beispielstruktur

Das folgende Beispiel soll Ihnen einen Anhaltspunkt geben, wie Ihr Dokument aufgebaut sein sollte:

```
<?xml version="1.0" encoding="UTF-8"?>

<book id="usersguide">
```

```
  <bookinfo>
    <title>Firebird Users Guide</title>
    <author>
      <firstname>William</firstname>
      <surname>Shakespeare</surname>
    </author>
    <edition>25 January 2006 – Document version 1.2</edition>
  </bookinfo>

  <chapter id="usersguide-intro">
    <title>Introduction</title>
    <para>Hello! This is the introductory text to the Firebird
      Users Guide.</para>
  </chapter>

  <chapter id="usersguide-download-install">
    <title>Downloading and installing Firebird</title>
    <para>In this chapter we'll demonstrate how to download and
      install Firebird.</para>
    <section id="usersguide-download">
      <title>Downloading Firebird</title>
      <para>To download Firebird from the Internet, first go to the
        following URL: etc. etc. etc.</para>
      ...more paragraphs, possibly subsections...
    </section>
    <section id="usersguide-install">
      <title>Installing Firebird</title>
      <para>Installing Firebird on your system goes like this:
        etc. etc.</para>
      ...more paragraphs, possibly subsections...
    </section>
  </chapter>

  ...more chapters...

  <appendix id="usersguide-dochist">
    <title>Document history</title>
    ...to be discussed later!

  <appendix id="usersguide-license">
    <title>License notice</title>
    ...to be discussed later!
</book>
```

Zu beachtende Punkte

- Beachten Sie zunächst wieder, dass Attribute in Anführungszeichen stehen. (Wenn Sie diese mit einem Attributeditor einsetzen, werden diese vom Programm selbst gesetzt.)

- Wie Sie im Beispiel sehen, können `chapters` und `sections` direkt mit einem oder mehreren `para`-Elementen starten. Sobald Sie aber Abschnitte (section) in einem Kapitel einbinden, oder Unterabschnitte in einem Abschnitt, können Sie keine weiteren `para`-Elemente danach verwenden - nur noch innerhalb dieser. Gute DocBook-XML-Editoren werden Sie dies auch nicht tun lassen. Wenn Sie Ihr DocBook-XML jedoch händisch erstellen, sollten Sie hierauf achten.

- Wenn Sie einen XML-Editor einsetzen, werden Sie vermutlich selten `para`-Elemente selbst erstellen. Füge ich beispielsweise ein `chapter` oder ein `section` im XMLMind XML Editor ein, wird automatisch ein –

zunächst leerer – `para` erstellt. Und wenn ich dann Text im Absatz eingebe und **ENTER** drücke, wird dieser Absatz automatisch mit einem `</para>` geschloseen und der nächste wird erstellt.

Überspringen Sie den Rest des Elemente-Unterabschnitts , sollten Sie schon alles über DocBook-Elemente wissen.

Listen

DocBook bietet verschiedene Listenelemente. Dies sind die gebräuchlichsten:

`itemizedlist`
Ein `itemizedlist` wird verwendet, um eine ungeordnete Aufzählungen zu erstellen:

```
<itemizedlist spacing="compact">
  <listitem><para>Oranges are juicy</para></listitem>
  <listitem><para>Apples are supposed to be healthy</para></listitem>
  <listitem><para>Most people find lemons way too sour</para>
    </listitem>
</itemizedlist>
```

Listeneinträgen werden allgemein mit Punkten vorangestellt angezeigt:

- Oranges are juicy
- Apples are supposed to be healthy
- Most people find lemons way too sour

Wenn Sie das `spacing`-Attribut weglassen, wird der Standard `normal` genutzt, was bedeutet, das vertikale Leerräume (typischerweise eine Zeilenhöhe) zwischen den Listeneinträgen eingefügt wird.

`orderedlist`
Ein `orderedlist` erzeugt eine geordnete Aufzählung, auch Nummerierung genannt:

```
<orderedlist spacing="compact" numeration="loweralpha">
  <listitem><para>Sumerians 3300 BC – 1900 BC</para></listitem>
  <listitem><para>Assyrian Empire 1350 BC – 612 BC</para></listitem>
  <listitem><para>Persian Empire 6th century BC – 330 BC</para>
  </listitem>
</orderedlist>
```

Standardmäßig werden Zahlen (1, 2, 3, ...) verwendet, die vor den Listeneinträgen erscheinen, aber diese können Sie mit dem Attribut `numeration` ersetzen lassen. Ausgabe:

a. Sumerians 3300 BC – 1900 BC
b. Assyrian Empire 1350 BC – 612 BC
c. Persian Empire 6th century BC – 330 BC

`procedure`
Ein `procedure` wird häufig wie ein `orderedlist` dargestellt, ist semantisch aber etwas anderes: ein procedure beschreibt eine bestimmte *Schrittfolge*:

```
<procedure>
  <step><para>Pick the lock</para></step>
  <step><para>Rob the house</para></step>
  <step><para>Get arrested</para></step>
</orderedlist>
```

So sieht die Ausgabe aus:

1. Pick the lock
2. Rob the house
3. Get arrested

Innerhalb eines `step`, können Sie einen Unterschritt `substeps` erstellen, welcher wiederum weitere `step`-Elemente enthalten kann.

`variablelist`

Ein `variablelist` besteht aus Einträgen der Art `varlistentry`, welche jeweils ein `term` beinhalten, gefolgt von einem `listitem`:

```
<variablelist>
  <varlistentry>
    <term>Tag</term>
    <listitem>
      <para>A piece of text enclosed in angle brackets</para>
    </listitem>
  </varlistentry>
  <varlistentry>
    <term>Element</term>
    <listitem>
      <para>A start tag, a matching end tag, and everything in
        between</para>
    </listitem>
  </varlistentry>
  <varlistentry>
    <term>Content of an element</term>
    <listitem>
      <para>Everything between the matching tags</para>
    </listitem>
  </varlistentry>
</variablelist>
```

Die Liste, die Sie gerade lesen, besteht aus verschiedenen Listentypen und ist selbst ein `variablelist` mit den Elementnamen (`itemizedlist`, `orderedlist`, etc.) als Einträge. Der nächste Abschnitt – *Links* – besteht ebenfalls aus einem Einleitungssatz gefolgt von einem `variablelist`.

Links

Sie können Hyperlinks zu Zielen innerhalb Ihres Dokuments, zu einem anderen Dokument in ihrem Dokumentensatz (set) oder im Internet.

`link`

`link` ist das allgemeine Element, das auf einem anderen Ort im Dokument oder Dokumentensatz (set) verweist. Das Attribut `linkend` muss immer vorhanden sein; sein Wert sollte die `id` des zu verlinkenden Elements (*link target*) sein.

```
Click <link linkend="docwritehowto-introduction">here</link> to jump
to the introduction.
```

Im gerenderten Dokument, ist „`here`“ ein *hot text*, das heißt: ein anklickbarer Link, der zur Einleitung verweist:

> Click here to jump to the introduction.

> **Achtung**
>
> Obwohl Sie `link` nutzen können, um auf ein beliebiges Element im gesamten Dokumentensatz (set) zu verlinken, sollten Sie dies jedoch nur für tun, wenn das Ziel auch im gleichen PDF exisitert. Die HTML-Version ist voll verlinkbar, die Ausgabe des PDF-Rendering hingegen, funktioniert nicht Dokumentübergreifen. Unsere PDFs enthalten typischerweise ein Buch (`book`) oder Artikel (`article`); wenn sich das Ziel außerhalb des aktuellen Dokuments befindet, nutzen Sie stattdessen `ulink` (siehe unten).

`ulink`

Verwenden Sie `ulink` um eine Internetresource zu verlinken. Das Attribut `url` ist selbsterklärend:

```
Click <ulink url="http://docbook.org/tdg/en/">this link</ulink> to
read The Definitive Guide on DocBook.
```

Die Wörter „`dieser Link`“ werden als Hyperlink zu `http://docbook.org/tdg/en/` dargestellt, in etwa:

> Click this link to read The Definitive Guide on DocBook.

`email`

Sie können einen E-Mail-Link mit `ulink` erstellen, es ist jedoch leichter das Element `email` zu benutzen. Dieses wird die E-Mail-Adresse als klickbaren Link in der Ausgabe anzeigen. Dieser XML-Code:

```
Send mail to
<email>firebird-docs-request@lists.sourceforge.net</email> to
subscribe.
```

wird in der Ausgabe zu:

> Send mail to <firebird-docs-request@lists.sourceforge.net> to subscribe.

Wenn Sie möchten, dass sich der *hot text* von der E-Mail-Adresse selbst unterscheidet, verwenden Sie `ulink` mit einer `mailto:`-URL.

> **Warnung**
>
> Wenn Sie Links zu E-Mail-Adressen einbinden, entweder mittels `email` oder `ulink` – oder wenn Sie dies nur *andenken*, und Ihr Dokument im Internet veröffentlicht wird, werden diese Adressen schnell von Spamern genutzt. Dies wird den Anteil von Spam für diese Adressen deutlich erhöhen. Stellen Sie also vorher sicher, dass der Eigentümer der Adresse sich auch einverstanden mit der Veröffentlichung zeigt!

`anchor`

Ein `anchor` (Ankerelement) ist ein leeres Element, dass eine genaue Position im Dokument auszeichnet. Es zeigt sich im lesbaren Text, aber es kann für ein Linkziel verwendet werden. Das ist nützlich, wenn Sie eine Stelle inmitten eines langen Absatzes verlinken wollen:

```
<para id="lost-at-sea">
  Blah blah blah...
  and some more...
```

```
  and then some...
  Now here's an interesting place in the paragraph I want to be able
  to link to:
  <anchor id="captain-haddock"/>There it is!
  Paragraph drones on...
  and on...
  and on...
</para>
```

Sobald der Anker platziert ist, können Sie hierhin verlinken:

```
<link linkend="captain-haddock">Go to the interesting spot</link> in
that long, long paragraph.
```

Wenn Ihr Link auf ein kurzes Element oder den Anfang eines Elements zeigt, ist es einfacher, dem Zielelement ein `id`-Attribut zu geben und dies als Verlinkung zu verwenden.

Programm-Listings, Bildschirmausgaben, literales Layout und Beispiele

`programlisting (Programm-Listings)`

Wenn Sie Code-Fragmente in Ihrem Dokument unterbringen wollen, packen Sie diese in ein `programlisting`-Element. Alles, was Sie innerhalb eines Programm-Listings schreiben, wird wörtlich, inklusive Zeilenumbrüche, Leerzeichen, etc., in die Ausgabe übernommen. Des weiteren wird eine Schriftart mit fester Zeichenbreite verwendet. Der Begriff „Programm-Listing“ ist im weiteren Sinne zu verstehen: Sie sollten das Element auch für SQL- und DocBook XML-Anweisungen und -Beispiele nutzen. Dieser Leitfaden - und insbesondere der Abschnitt über Elemente, welchen Sie gerade lesen - ist zugemüllt mit `programlistings`, also wissen Sie bereits wie diese aussehen:

```
Programlistings are rendered like this.
```

> **Wichtig**
>
> In Programm-Listings sollten Sie die Zeilenweite auf 70 Zeichen beschränken. Andernfalls wird der Text in PDFs rechts über den Rand laufen. Das gleiche gilt für andere Elemente wie `screen`, `literallayout`, etc.

`screen (Bildschirmausgaben)`

Verwenden Sie ein Element `screen`, um zu zeigen, was ein Anwender sieht oder sehen sollte, wenn sich im Textmodus oder Terminalmodus befindet. Hier wird das Layout ebenfalls durch eine Festbreitenschrift dargestellt, aber die Semantic ist eine andere. In der Ausgabe unterscheided sie sich nicht vom Programm-Listing. Hier ein kurzes Beispiel, was passiert, wenn Sie versuchen ein nichtexistierendes Ziel im *manual*-Baum zu erstellen:

```
<screen>
D:\Firebird\manual_incl_howto\src\build>build ugh
java version "1.4.2_01"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.2_01-b06)
Java HotSpot(TM) Client VM (build 1.4.2_01-b06, mixed mode)

Buildfile: build.xml

BUILD FAILED
Target `ugh' does not exist in this project.
```

```
</screen>
```

Und so sieht die Ausgabe aus:

```
D:\Firebird\manual_incl_howto\src\build>build ugh
java version "1.4.2_01"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.2_01-b06)
Java HotSpot(TM) Client VM (build 1.4.2_01-b06, mixed mode)

Buildfile: build.xml

BUILD FAILED
Target `ugh' does not exist in this project.
```

literallayout (wörtliches Layout)

`literallayout`, wie `screen` und `programlisting` hält Ihr Layout intakt und ändert die Schriftart normalerweise nicht - es sei denn Sie setzen das Attribut `class` auf `monospaced`. Es ist ein allgemeineres Element, als die zwei vorhergehenden. Seinem Inhalt wird keine Bedeutung zugemessen: Sie können irgendeine Art Text hier platzieren, wenn Sie das Layout erhalten wollen.

Beispiel (Quelltext):

```
<literallayout>
The Sick Rose

Oh Rose, thou art sick!
The invisible worm
That flies in the night,
In the howling storm,

Has found out thy bed
Of crimson joy,
And his dark secret love
Doth thy life destroy.

  — William Blake
</literallayout>
```

Beispiel Ausgabe:

The Sick Rose

Oh Rose, thou art sick!
The invisible worm
That flies in the night,
In the howling storm,

Has found out thy bed
Of crimson joy,
And his dark secret love
Doth thy life destroy.

— William Blake

example (Beispiel)

Ein `example` stellt ein formales Beispiel samt Titel dar. Es wird normalerweise mit einem `id`-Attribut verwendet, womit es an anderen Dokumentstellen referenziert werden kann. Ein Beispielverzeichnis wird

automatisch erstellt, wenn das Dokument gerendert wird. Sie werden häufig `programlisting`s ein einem `example` finden, aber es kann auch `screen`s, `para`s, lists, etc. enthalten.

Hier ist ein Beispiel für die Verwendung von `example`:

```
<example id="docwritehowto-sql-example">
  <title>Ein SQL-Beispiel</title>
  <para>Mit diesem Befehl können Sie alle Datensätze der Tabelle COUNTRY auflisten:</para>
  <programlisting>SELECT * FROM COUNTRY;</programlisting>
</example>
```

In der Ausgabe sieht dies so aus:

Beispiel 1. Ein SQL-Beispiel

Mit diesem Befehl können Sie alle Datensätze der Tabelle COUNTRY auflisten:

```
SELECT * FROM COUNTRY;
```

Wenn Sie ein Beispiel ohne zwingenden Titel erstellen möchten, verwenden Sie `informalexample`. Informelle Beispiele sind nicht Bestandteil des Beispielverzeichnisses.

Tabellen

Wenn Sie schonmal eine Tabelle in HTML erstellt haben, werden Sie auch keine großen Schwierigkeiten bei der Erstellung von Tabellen mittels DocBook haben. Es gibt ein paar Unterschiede, und DocBook-Tabellen sind weit mächtiger.

Ein Element `table` besteht aus einem `title` und einer oder mehreren `tgroup`s – übrlicherweise eine. Das `tgroup`-Element besitzt ein notwendiges Attribut: `cols`. Hierfür sind die Anzahl der Spalten innerhalb des `tgroup` anzugeben. Innerhalb eines `tgroup` können Sie die Elemente `thead`, `tfoot` und `tbody` platzieren. Jedes dieser Elemente hat ein oder mehrere Zeilen (engl. `row`s), welche wiederum so viele Zellen (engl. `entry`s) beinhalten, wie Sie zuvor im Attribut `cols` angegeben haben. (Sie können Zellen verbinden, aber das wird hier nicht behandelt.)

Soviel zu Basisstruktur. Jetzt werden wir Ihnen ein Beispiel zeigen; Erst als DocBook XML-Quelltext, und dann die resultierende Tabelle in der gerenderten Ausgabe. Machen Sie sich keine Gedanken über die `<colspec>`s; Dies sind keine Pflicht-Unterlemente. Sie sind nur zum Feintuning da.

```
<table id="docwritehowto-table-dboftheyear">
  <title>LinuxQuestions.org poll: Database of the year 2003</title>

  <tgroup cols="3">
    <colspec align="left" colname="col-dbname" colwidth="2*"/>
    <colspec align="right" colname="col-votes" colwidth="1*"/>
    <colspec align="right" colname="col-perc" colwidth="1*"/>

    <thead>
      <row>
        <entry align="center">Database</entry>
        <entry align="center">Votes</entry>
        <entry align="center">Percentage</entry>
      </row>
    </thead>
```

```
    <tfoot>
      <row>
        <entry>Total</entry>
        <entry>1111</entry>
        <entry>99.99</entry>
      </row>
    </tfoot>

    <tbody>
      <row>
        <entry>MySQL</entry>
        <entry>405</entry>
        <entry>36.45</entry>
      </row>
      <row>
        <entry>Firebird</entry>
        <entry>403</entry>
        <entry>36.27</entry>
      </row>

      ... 5 more rows not shown here ....

    </tbody>
  </tgroup>
</table>
```

Und hier ist die resultierende Tabelle:

Tabelle 2. LinuxQuestions.org poll: Database of the year 2003

Database	Votes	Percentage
MySQL	405	36.45
Firebird	403	36.27
Postgres	269	24.21
Oracle	25	2.25
Berkeley DB	4	0.36
Sybase	3	0.27
DB2	2	0.18
Total	**1111**	**99.99**

Nebenbei ist dies ein reeller Auszug von LinuxQuestions.org. Wie Sie sehen können, fehlen lediglich drei Leute, die zum Sieg für Firebird hätten stimmen müssen. Wenn Sie diese drei Personen kennen, kontaktieren Sie bitte unseren Chef-Inquisitor. Er würde gern ein kleines, ähm... *Gespräch* mit ihnen führen :–)

Tabellen werden automatisch inidziert. Ein `informaltable` hat die gleiche Struktur wie ein `table`, benötigt aber keinen Titel und wird nicht im Tabellenverzeichnis geführt. Wenn Sie Tabellen verschachteln wollen, nutzen Sie entweder `table`/`informaltable` *innerhalb* eines `entry` oder ein `entrytbl` *anstelle* eines `entry`.

Tabellen haben deutlich mehr Eigenschaften als hier gezeigt, aber dies sei nun an Ihnen herauszufinden.

HTML-Tabellen

DocBook in den Versionenen 4.3 und höher erlaubt es Ihnen, Tabellen im HTML-Stil zu erstellen. Somit ist die Verwendung von `trs` anstelle von `rows` und `td`/`th` anstelle von `entry` erlaubt. Warum sollten Sie das tun? Es gibt zwei Situationen, in denen die Verwendung der HTML-Tabellen vorteilhaft ist:

- Sie haben bereits eine vorhandene HTML-Tabelle und möchten nicht die Zeit investieren, diese zu konvertieren;
- Sie möchten verschiedene Hintergrundfarben in der Tabelle verwenden. Dies kann zwar auch in der DocBook-Tabelle getan werden, aber nur mit Verarbeitungsanweisungen, also den *processing instructions* – eine für jedes Ziel jedes Elementes, das eine eigene Farbe erhalten soll. In einer HTML-Tabelle können Sie das Attribut `bgcolor` des Kindelements verwenden.

Eine HTML-Tabelle darf keine `tgroups` enthalten; verwenden Sie `trs` entweder direkt in der Tabelle oder im Tabellenkopf `thead` / Tabellenkörper `tfoot` / Tabellenfuß `tbody`. Diese sind direkte Kindelemente der Tabelle selbst. Außerdem hat die HTML-Tabelle das Element `caption` anstelle von `title`. (Ein `informaltable`-Element hat weder caption noch title.)

Hier ist der Quelltext einer HTML-Tabelle:

```
<table bgcolor="blue" border="1">
  <caption align="bottom">An HTML-style table</caption>

  <tr bgcolor="#FFE080">
    <th>First column</th>
    <th bgcolor="#FFFF00">Second column</th>
  </tr>
  <tr align="center">
    <td bgcolor="orange" colspan="2">Table cell spanning two
      columns</td>
  </tr>
  <tr>
    <td bgcolor="#00FFC0">Yes, here I am</td>
    <td align="right" bgcolor="#E0E0E0" rowspan="2" valign="bottom">And
      there I go!</td>
  </tr>
  <tr>
    <td bgcolor="#FFA0FF">Another row...</td>
  </tr>
</table>
```

Und hier ist das Ergebnis:

First column	Second column
Table cell spanning two columns	
Yes, here I am	And there I go!
Another row...	

Tabelle 3. An HTML-style table

Nicht alle HTML-Tabellenelemente und -attribute werden von den Stylesheets unterstützt. So werden beispielsweise Eigenschaften, die in `col` und `colgroup` definiert wurden, nicht berücksichtigt. Definieren Sie diese stattdessen in `td`-/`th`-Elementen - oder erweitern Sie die Stylesheets!

> **Anmerkung**
>
> In XMLMind können Sie eine HTML-Tabelle über das Menü in der Werkzeugleiste erstellen. Über den Bearbeiten-Bereich können Sie lediglich DocBook-Tabellen erstellen.

PDF-Ausgabe großer Tabellen

DocBook `table`s gehören zu den sogenannten formalen Elementen (engl. *formal elements*). Formale Elemente werden automatisch in Verzeichnisse aufgenommen (Tabellenverzeichnis, Abbildungsverzeichnis, etc.); wenn ein formales Element kein Attribut `id` besitzt, weist ihm das Stylesheet eines zu. Die Vorlage, die die XSL-FO-Ausgabe generiert (den Zwischenschritt hin zum PDF), weist jedem formalen Objekt auch das Attribut `keep-together.within-page="always"` zu, was verhindert, dass Seitenumbrüche innerhalb eines Objekts stattfinden. Dies ist normalerweise gewünscht, aber was passiert, wenn das Objekt nicht auf eine Seite passt? Bis vor kurzem nutzten wir Apache FOP 0.20.5 um die XSL-FO-Ausgabe zu erstellen. Dieser Prozessor ignorierte das `keep-together`-Attribut, wenn das Objekt zu groß war. Aber die derzeitige Version (0.93 oder höher) "drückt" diese Eigenschaft *immer* durch. Das Ergebnis ist, dass das Obejekt in diesem Falle abgeschnitten (oder in irgendeiner anderen Art passend gemacht) wird, damit es auf die Seite passt. Dies ist ein Feature, kein Bug. Somit gibt es auch keinen Grund sich hierüber zu beschweren.

Es gibt zwei Workarounds, sollte eine Tabelle zu groß für eine Seite sein:

1. Wenn die Tabelle keinen Titel benötigt und es Ihnen nichts ausmacht, dass diese nicht im Tabellenverzeichnis aufgelistet wird, verwenden Sie stattdessen `informaltable`.

2. Fügen Sie eine Verarbeitungsanweisung (*processing instruction*) am Anfang der Tabell ein:

   ```
   <table frame="all" id="ufb-about-tbl-features">
     <?dbfo keep-together='auto'?>
     <title>Summary of features</title>
   ```

   In XMLMind wird dies folgendermaßen umgesetzt:

   1. Setzen Sie den Cursor in das Titelelement.

   2. Wählen Sie *Edit -> Processing Instruction -> Insert Processing Instruction Before* aus dem Menü. Eine grüne Zeile erscheint oberhalb des Titels.

   3. Geben Sie `keep-together='auto'` in diese Zeile ein.

   4. Lassen Sie den Cursor auf der grünen Zeile, wählen Sie dann *Edit -> Processing Instruction -> Change Processing Instruction Target* aus dem Menü. Nun erscheint eine Dialog-Box.

   5. Ändern Sie `target` in `dbfo` und klicken Sie auf OK.

   Natürlich können Sie das Gleiche für kleinere Tabellen tun, wenn Sie dort Zeilenumbrüche bevorzugen. Die gegenteilige Anweisung `<?dbfo keep-together='always'>` wird Seitenumbrüche in `informaltables` verhindern. Stellen Sie sicher, dass die Elemente auf eine Seite passen, bevor Sie dies verwenden!

Bilder

Um Bilder einzubinden, verwenden Sie `mediaobject` welches ein `imageobject` beinhaltet, das wiederum ein `imagedata` beinhaltet:

```
<mediaobject>
  <imageobject>
    <imagedata align="center" fileref="images/services.png"
      format="PNG"/>
  </imageobject>
</mediaobject>
```

Vielleicht wundern Sie sich, dass 3 verschachtelte Elemente benötigt werden, um ein simples Bild einzubinden. Hierfür gibt es einen guten Grund, den werde ich Ihnen aber nicht erzählen ;-) - der ist für uns nicht von Belang. Wir müssen nur wissen, dass es so gemacht wird.

Ungeachtet des Bildverzeichnisses, welches relativ zur DocBook-Quelle steht, sollte `fileref` *immer* in der Form `images/`*`filename.ext`* angegeben werden. Die geschieht, weil die HTML- und die FO-Ausgabe die Bilddateien aus ihren Quellverzeichnissen zu einem Unterverzeichnis namens `images` im Ausgabepfad kopieren werden. (Die FO-Ausgabe ist eine Zwischenform. Sobald dies zum PDF gewandelt wurde, ist das Bild in der Datei selbst enthalten.)

Wenn die Dateireferenz nicht „korrekt“ aus Sicht der Dateiquellen ist, werden Sie das Bild in XMLMind auch nicht sehen. Sollte Sie das stören, erstellen Sie einen Symlink auf den Bildordner (Linux) oder kopieren Sie diesen in das gleiche Verzeichnis wie die Quelldatei (Windows). Eine Verknüpfung scheint unter Windows nicht zu funktionieren. Nutzen Sie die Kopien nur in Ihrer lokalen Kopie - erstellen Sie keinen Commit mit doppelten Bilddaten ins CVS!

Ein `mediaobject` wird als separater Block formatiert. Wenn Sie das Bild mit Text eibinden wollen, nutzen Sie stattdessen `inlinemediaobject`; die verschachtelten Element bleiben wie sie sind.

> **Hinweis für Übersetzer**
>
> Übersetzer: Alle Bilder, die Sie nicht für Ihre Sprache überarbeiten oder ersetzen, sollten nicht in Ihren sprachabhängigen Dokumentensatz (set) übernommen werden. Seit Januar 2006 sehen die *build tools* erst in Ihrem Sprachverzeichnis (z.B. `manual/src/docs/firebirddocs-fr/images`), und danach in `manual/src/docs/firebirddocs/images`. Somit wird kein CVS-Speicher verschwendet, wenn Sie die Originalbilder verwenden.
>
> Das gleiche Verhalten ergibt sich für die anderen Sätze: Wenn ein Bild von, sagen wir mal spanischen Release Notes, referenziert wird, und dieses nicht in `rlsnotes-es/images` vorliegt, wird dieses aus `rlsnotes/images` verwendet. Es funktioniert nicht sprach-*übergreifend*.

Hinweise

DocBook besitzt mehrere Tags, um einen Textblock als Hinweis, Warnung, Tipp, etc. zu markieren. In der Ausgabe werden solche Blöcke eingerückt und mit einem Symbol oder Wort zur Kennzeichnung ihres Zwecks markiert. Die verwendbaren Tags in alphabetischer Reihenfolge:

`<caution>`, `<important>`, `<note>`, `<tip>`, und `<warning>`

Ich werde Ihnen einen Tipp (`<tip>`) als Beispiel zeigen; Die anderen werden genauso verwendet:

```
<tip>
  <para>If you insert a caution, important, note, tip, or warning
    element in your text, don't start it with the word caution,
    important, note, tip, or warning, because these words are usually
    automatically generated by the rendering engine.</para>
</tip>
```

Und dies ist die Ausgabe:

> **Tipp**
>
> If you insert a `<caution>`, `<important>`, `<note>`, `<tip>`, or `<warning>` element in your text, don't start it with the word `caution`, `important`, `note`, `tip`, or `warning`, because these words are usually automatically generated by the rendering engine.

Sie haben vielliecht bemerkt, dass die Wörter `caution`, `important` etc. sich vom Aussehen des restlichen Tipp-Textes unterscheiden. Wie kommt das? Um Ihnen die Wahrheit zu sagen, habe ich diese mit speziellen Tags umschlossen (erst mit `<sgmltag>`s, zweitens mit `<literal>`s) damit sie nun so aussehen wie sie hier erscheinen. Aber das ließ den XML-Quelltext sehr unsauber aussehen. Deshalb entschied ich mich diese Tags aus den Beispielquellen zu entfernen.

Optional können Sie den Hinweisen ein Element `title` mitgeben. Wenn Sie dies nicht tun, wird ein Standardkopf (in Dokumentsprache) für die Ausgabe erstellt.

Möchten Sie einen Textblock erstellen ohne diesen als Tipp oder anderes zu kennzeichnen, nutzen Sie `<blockquote>`.

Absatzköpfe

Wenn Sie einen Absatzkopf oder Titel ohne Unterabschnitt erstellen möchten, gibt es ein paar Möglichkeiten.

`bridgehead`
: Ein `bridgehead` ist ein flussfreier Titel zwischen Absätzen, der keinem Anfang eines Kapitels oder Abschnitts zugeordnet ist. Das Attribut `renderas` gibt an, wie es gerendert wird.

```
<para>You may remember that Mr. Hardy started with this firm as
  elevator boy and with grim determination worked his way up to
  the top. And after the wedding today he becomes General Manager
  of this vast organisation.</para>

<bridgehead renderas="sect5">Mr. Laurel's comments</bridgehead>

<para>We also spoke to his lifetime friend and companion Mr. Laurel.
  Mr. Laurel says that after viewing the situation from all sides,
  he is thoroughly reconciled to the fact that the moving picture
  industry is still in its infancy. Mr. Laurel also states that
  technology, whilst it may appear to be the center of all—</para>
```

Der oben angegebene Quelltext erscheint so in der Ausgabe:

> You may remember that Mr. Hardy started with this firm as elevator boy and with grim determination worked his way up to the top. And after the wedding today he becomes General Manager of this vast organisation.
>
> #### Mr. Laurel's comments
>
> We also spoke to his lifetime friend and companion Mr. Laurel. Mr. Laurel says that after viewing the situation from all sides, he is thoroughly reconciled to the fact that the moving picture industry is still in its infancy. Mr. Laurel also states that technology, whilst it may appear to be the center of all—

Sie können frei wählen, welchen Level Sie für `renderas` nutzen wollen, aber logischerweise wird dies meist die derzeitige Ebene plus (mindestens) eins sein.

formalpara
Ein formalpara ist ein Absatz mit Titel. Unsere Stylesheets rendern den Titel als mitlaufenden Kopf.

```
<formalpara>
  <title>Motherly love:</title>
  <para>This is the love your mother has for you, not to be
    confused with brotherly or otherly love.</para>
</formalpara>
```

In der Ausgabe erscheint damit:

> **Motherly love:** This is the love your mother has for you, not to be confused with brotherly or otherly love.

Eine Abgrenzung wird zum Titel hinzugefügt, sofern es nicht bereits auf ein Satzzeichen (Doppelpunkt) endet.

Verschiedene inline-Elemente

Zum Abschluss dieses Abschnitts zu DocBook-Elementen, werde ich noch eine Kurzbeschreibung zu einigen Inline-Elementen (*inline elements*) geben. Sie heißen „inline“, da sie den Fluss des Textes nicht beeinflussen. Wenn ich zum Beispiel das Element emphasis benutze:

```
Don't <emphasis>ever</emphasis> call me fat again!
```

ist das Ergebnis:

> Don't *ever* call me fat again!

Das Wort „ever“ wird betont, aber es behält seinen Platz im Satz. Wir haben bereits einige Inline-Elemente kennengelernt: die verschiedenen Linkarten. Andere Elemente - wie table, warning, blockquote und programlisting – werden immer als Block dargestellt, abgesetzt vom umlaufenden Text (selbst wenn Sie diese als „inline“ in Ihrem XML-Quelltext setzen). Nicht überraschend ist somit, dass diese Block-Elemente (*block elements*) genannt werden. Block-Elemente beinhalten häufig Inline-Elemente; andersherum ist es nicht möglich.

OK, starten wir mit diesen Inline-Elementen. Ich werde Beispiele - jeweils mit XML-Quelltext und als gerenderte Ausgabe - für die meisten zeigen:

filename – command – application – envar
Verwenden Sie das Tag filename um einen Dateinamen im weitesten Sinne zu markieren. Optionale Attribute können außerdem anzeigen, ob es sich hierbei um eine Headerdatei, ein Verzeichnis, etc. handelt.

```
Place your doc in the <filename
class="directory">src/docs/firebirddocs</filename> subdirectory.
```

Die Ausgabe:

> Place your doc in the src/docs/firebirddocs subdirectory.

command und application kennzeichnen beide ausführbare Programme. command wird üblicherweise für kleinere Programme und interne Befehle verwendet; sein Inhalt sollte der exakt einzugebende Befehl sein; application wird grundsätzlich für größere Programme verwendet. Der Name der ausführbaren Datei wird nicht benötigt. Beide können auf das gleiche Programm verweisen:

```
Type <command>netscape&amp;</command> in a terminal window to start
<application>Netscape Navigator</application>.
```

Dies ist die Ausgabe:

> Type **netscape&** in a terminal window to start Netscape Navigator.

`envar` kennzeichnet eine Umgebungsvariable.

`subscript` – *`superscript`*
Die zwei Elemente im Einsatz:

```
After inventing the formula e = mc<superscript>2</superscript>, I
really felt like a glass of liquid H<subscript>2</subscript>O !
```

Ausgabe: After inventing the formula e = mc^2, I really felt like a glass of liquid H_2O !

`varname` – *`constant`* – *`database`*
Die Verwendung von `varname` und `constant` sollte offensichtlich sein. Das Tag `<database>` kann für Datenbankobjekte verwendet werden, muss aber nicht:

```
The <database class="table">COUNTRY</database> table has two fields:
<database class="field">COUNTRY</database> and
<database class="field">CURRENCY</database>.
```

Ausgabe: The COUNTRY table has two fields: COUNTRY and CURRENCY.

`function` – *`parameter`* – *`returnvalue`*
Diese drei sprechen für sich selbst, glaube ich.

```
The <function>log</function> function takes parameters
<parameter>a</parameter> and <parameter>b</parameter>.
```

Ausgabe: The `log` function takes parameters *`a`* und *`b`*.

`prompt` – *`userinput`* – *`computeroutput`*
`prompt` wird für eine Zeichenkette verwendet, die anzeigt, dass der Benutzer eine Eingabe tätigen soll; `userinput` verweist auf den eingegebenen Text (nicht zwangsweise in der Befehlszeile!); `computeroutput` ist der Text, der vom Computer ausgegeben wurde:

```
Type <userinput>guest</userinput> at the <prompt>login:</prompt>
prompt and the server will greet you with a <computeroutput>Welcome,
guest user</computeroutput>.
```

Ausgabe: Type **`guest`** at the `login:` prompt and the server will greet you with a `Welcome, guest user`.

`keycap`
Der Text einer Taste, oder allgemeiner:

```
Hit the <keycap>Del</keycap> key to erase the message, or
<keycap>SPACE</keycap> to move on.
```

Ausgabe: Hit the **Del** key to erase the message, or **SPACE** to move on.

sgmltag
Dieses Element wird innerhalb dieses Leitfadens ausführlich genutzt: Es markiert marks SGML *und* XML-Tags, Elemente, Attribute, Einträge, etc.:

```
If it concerns a directory, set the
<sgmltag class="attribute">class</sgmltag> attribute of the
<sgmltag class="element">filename</sgmltag> element to
<sgmltag class="attvalue">directory</sgmltag>.
```

Ausgabe: If it concerns a directory, set the `class` attribute of the `filename` element to `directory`.

Andere mögliche Werte für `sgmltag.class` sind: `starttag`, `endtag`, `emptytag`, und `genentity` (für einen Eintrag).

emphasis – citetitle – firstterm
Verwenden Sie `emphasis` um Wörter zu betonen, `citetitle` für Buchtitel, etc., und `firstterm` wenn Sie Ihren Lesern ein neues Wort oder Konzept vorstellen:

```
We use <firstterm>DocBook XML</firstterm> for our Firebird
documentation. A short introduction follows;
<emphasis>please</emphasis> read it carefully! If you want to know
more about the subject, buy <citetitle>DocBook – The Definitive
Guide</citetitle>.
```

Ausgabe: We use *DocBook XML* for our Firebird documentation. A short introduction follows; *please* read it carefully! If you want to know more about the subject, buy *DocBook – The Definitive Guide*.

quote – literal
Verwenden Sie `quote` für einen Inline-Bereich, der in Anführungszeichen steht (im Gegensatz zu `blockquote`). Anführungszeichen werden automatisch eingefügt. Die Nutzung von `quote` anstelle der Eingabe von Anführungszeichen durch Sie selbst (was natürlich auch möglich ist), hat den Vorteil, dass wir die Art der Anführungszeichen jederzeit durch unsere Stylesheets anpassen können, wenn wir wollen. Außerdem unterscheiden sich die Anführungszeichen von Sprache zu Sprache:

```
<para>An <quote lang="en">English quote</quote>
  and a <quote lang="fr">French quote</quote>.</para>
```

Ausgabe: An “English quote” and a « French quote ».

Bitte beachten Sie, dass Sie das Attribut `lang` nicht zusammen mit `quote`s innerhalb Ihres eigenen Dokuments verwenden. Ihr Wurzelelements `lang`-Attribut wird sicherstellen, dass das korrekte Anführungszeichen verwendet wird. Wenn jemand Ihr Dokument übersetzt - und das Wurzel-`lang`-Attribut ändert – wird es mit den Anführungszeichen der Zielsprache gerendert. Natürlich hatte ich dieses Attribut hier zu verwenden, um die Unterschiede zu zeigen.

Ein `literal` ist ein Wort oder Textfragment, dass literal (buchstäblich) wiedergegeben wird. Es ist ein allgemeines Element, häufig verwendet, um Wörter typografisch auszuweisen:

```
At all costs avoid using the word <literal>humongous</literal> in
your documentation.
```

Ausgabe: At all costs avoid using the word `humongous` in your documentation.

Sollten Sie diese Elemente wann immer möglich verwenden? Nunja, wenn Sie es tun, werden Sie Ihr Dokument sicherlich reichhaltiger machen; Sie machen es z.B. leichter, nach Dateinamen zu scannen oder ein Verzeichnis aller genutzten Anwendungen zu erstellen. Auf der anderen Seite gibt es so viele dieser semantischen Elemente

(tatsächlich haben wir nur *einige* hier besprochen), dass Sie die Segel streichen werden, wenn Sie alle anwenden wollten. Das ist nicht unser Anliegen: Wenn Sie sich wirklich verrückt machen wollen, tun Sie dies bitte *nachdem* Sie Ihr Dokument comitted haben :-)

Somit gilt als allgemeiner Rat: gehen Sie behutsam mit diesen Elementen um; verwenden Sie sie dort, wo es nach Ihrem ermessen Sinn macht, aber übertreiben Sie es nicht.

Abschluss der Elemente

Sie haben sicherlich bemerkt, dass in gerenderten Dokumenten (Sie lesen gerade eins, sofern Sie nicht die XML-Version geöffnet haben) viele Elemente das gleiche Aussehen besitzen: Ein `filename`, ein `literal` und ein `application` haben die gleiche Typografie; das gleiche gilt für `emphasis`, `firstterm` und `citetitle`.

Was ist also der Sinn dieser verschiedenen Tags? Warum nicht nur ein paar, wie `emphasis` und `literal`, wenn sie sowieso gleich aussehen. Dafür gibt es zwei gute Gründe:

- Erstens, wenn wir die meisten unserer Inline-Elemente als `emphasis` und `literal` kennzeichnen würden, ginge auch die Semantik verloren. Rufen Sie sich in Erinnerung, dass es im DocBook-XML nur um die Struktur und Semantik geht. `firstterm` und `citetitle` können genauso *aussehen* wie `emphasis`, sobald es gerendert wurde, aber sie sind nicht das Gleiche. Die XML-Quellen kennen diesen Unterschied, auch wenn dies nicht immer offensichtlich ist. Diese Information ist nützlich, und wir wollen diese nicht verlieren.

- Des weitern, können wir unsere Stylesheets für jedes Element individuell anpassen. Sobald wir also entscheiden, dass ein `firstterm` anders als ein `citetitle` aussehen soll, können wir das tun - aber *nur* wenn sie tatsächlich als unterschiedliche Tags markiert wurden, nicht wenn beide `emphasis`'s im XML-Quelltext sind.

Damit schließen wir die Abschnitte über DocBook ab. Mit dem bis hierher gezeigten Wissen, sollten Sie nun in der Lage sein, DocBook XML-Dokumente für das Firebird-Projekt zu erstellen. Wenn Sie einen dedizierten XML-Editor - was sehr ratsam ist - sollten Sie außerdem dessen Dokumentation konsultieren, um den Umgang hiermit zu lernen; Dies können wir mit diesem Leitfaden nicht abbilden.

Sprache und Stil

Nach der Flut von DocBook-Informationen im vorigen Abschnitt, werden wir unsere Aufmerksamkeit einem anderen wichtigen Aspekt des Schreibens widmen: Sprache und Stil (in diesem Abschnitt) sowie Copyrights (im nächsten Abschnitt).

Sprache

Die Firebird Community ist sehr verschieden und besteht aus Menschen mit vielen verschiedenen Muttersprachen. Wenn Sie Ihre Dokumentation in einer anderen Sprache als der eigenen schreiben, werden Sie wahrscheinlich einige Fehler machen. Dies ist nicht katastrophal, aber man sollte zumindest versuchen, die Anzahl der Fehler zu reduzieren. Einige Strategien, die Ihnen dabei helfen, sind:

- Benutzen Sie ein Wörterbuch! Einfach, effektiv und erstaunlicherweise keine Hightech.
- Wenn Sie zwischen zwei Schreibweisen eines Wortes oder zwischen mehrere mögliche Versionen eines Ausdrucks schwanken, googeln Sie nach Alternativen und achten Sie auf deren Frequentierung. Folgen Sie Links, um zu sehen wie Muttersprachler das Wort oder den Ausdruck in ihren Texten verwenden.
- Lassen Sie einen Muttersprachler über Ihren Text sehen und ggf. korrigieren.

Stil

Erwarten Sie nicht, einen Style Guide hier zu finden - ich wüsste nicht, wie man einen schreibt. Nur ein paar Hinweise und Tipps:

- Versuchen Sie, in einfacher, alltäglicher Sprache, wo immer möglich zu schreiben. Vermeiden Sie schwierige Wörter, wenn es eine vertraute, einfache Alternative gibt.
- Vermeiden Sie lange Sätze (über 25 Wörter), wenn Sie können. Vermeiden Sie vor allem zwei oder mehr lange Sätze unmittelbar nacheinander.
- Seien Sie vorsichtig mit Konstrukten wie doppelter oder dreifacher Verneinung („Ich kann nicht leugnen, dass ich nicht unzufrieden bin“) und passiver Sprache („Vorsicht sollte geboten sein...“). Sie müssen diese nicht unter allen Umständen vermeiden, aber sie machen einen Satz schwer verständlich. Um dies zu verhindern, nutzen Sie positive („Ich bin zufrieden“) und aktive („Seien Sie vorsichtig...“) Sprache.
- Verwenden Sie Listen für Aufzählungen, z.B. für:

  - Eine Sammlung von Hinweisen und Tipps.
  - Eine Reihe Beispiele (wie dieses hier).
  - Schritte, die in einer Nummerierung erscheinen.
  - Alternative Lösungsansätze für ein Problem.

  Sind jedoch nur wenige Einträge zu nennen, verwenden Sie stattdessen reinen Text: „Meine Mutter liebt drei Männer: John, Dick und Dave.“
- Nutzen Sie nicht zu viele Ausrufezeichen. Verwenden Sie niemals mehrere Ausrufezeichen oder Fragezeichen. Das ist ärgerlich!! Glauben Sie nicht auch??

Docwriter-Block

Manchmal wissen Sie, was Sie schreiben wollen, Sie haben alle Wörter, aber Sie bekommen den Satz nicht zusammen - Ihnen gelingt einfach kein *Fluss*. Das ist sehr frustrierend und es kann manchmal den Fortschritt Ihres Textes für viele Minuten blockieren. Und es ist umso frustrierender, weil Sie ja *wissen* was Sie Ihren Leser sagen wollen, aber Sie scheinen nicht in der Lage zu sein, einen anständigen Satz zu produzieren. Nach vielen schmerzhaften Erfahrungen dieser Art, habe ich die folgende Strategie entwickelt (nicht, dass ich glaube, ich wäre der erste):

1. Notieren Sie, was Sie in losen Sätzen und Wortbrocken sagen möchten. Machen Sie sich keine Gedanken über den Stil. Schreiben Sie einfach, was Sie dem Leser sagen wollen, stellen Sie sicher, dass alles da ist und in der richtigen Reihenfolge. Wenn Sie währenddessen merken, dass Sie bei einer Sache unsicher sind, machen Sie sich eine Bemerkung an diesem Punkt. Heben Sie Ihre Bemerkungen vom restlichen Text ab z. B. <<wie diesen>> oder !WIE DIESEN!

Dies kann in folgendem Text enden:

> CVS bedeutet Concurrent Versions System (<<überprüfen!>>). Zweck: Verwaltung von Softwareversionen. Sie können es verwenden, allein oder in einer Gruppe. Sie müssen einen CVS-Client benutzen. Ein CVS-Client ist ein Programm, mit dem Sie auf ein CVS-Repository (<<Begriff erklären?>>) zugreifen können. Um herauszufinden, ob Sie einen CVS-Client auf Ihrem System installiert haben, geben Sie „ cvs “ auf der Kommandozeile ein. Wenn Sie keinen installiert haben, rufen Sie diese URL für den Download auf .... [etc., etc.]

2. Wenn Sie Bemerkungen eingefügt haben, kümmern Sie sich erst um diese. *Prüfen Sie,* ob CVS wirklich Concurrent Versions System bedeutet (tut es). *Entscheiden Sie,* ob Sie wirklich den Begriff „CVS repository“ an diesem Punkt erklären sollten (sollten Sie).

3. Gehen Sie nun nochmals über den Absatz und versuchen Sie, den Text fließender zu gestalten. Wahrscheinlich wird dies leichter sein als Sie glauben!

4. Wenn es noch etwas holprig aussiet, egal - besser holprig und klar als glatt-fließend und unpräzise. Vielleicht können Sie diese Passage bei einem späteren Besuch etwas schöner gestalten.

Dieser Ansatz funktioniert gut für mich. Wenn also auch mals auf diese Weise festhängen, probieren Sie es aus, hoffentlich wird es auch Ihnen helfen.

Fragen des Urheberrechts

Viele Leute finden, rechtliche Fragen langweilig, aber dies ist ein wichtiger Abschnitt. Bitte lesen Sie ihn gründlich.

Verwendung fremden Materials

Während wir unsere Handbücher schreiben, ziehen wir alle anderen Arten von Dokumentation zu Rate - und das sollten wir auch, da wir das beste Ergebnis erzielen wollen. Alle Informationen, die wir in öffentlich zugänglichen Drittanbieter-Handbücher, Benutzerhandbücher, Lernprogramme usw. finden, können in unseren eigenen Dokus frei verwendet werden. Aber aber es ist wichtig *Informationen* nicht mit *wörtlichen Text* zu verwechseln. Wir dürfen nicht einfach Text aus anderen Werken in unsere eigenen Dokumentation kopieren und einfügen, sofern es uns nicht seitens des Autors ausdrücklich erlaubt ist.

Wenn Sie einen Text von jemand anderem verwenden wollen, überprüfen Sie den Urheberrechtsschutz der Arbeit. Gibt es keinen, wird die Arbeit automatisch nach der Berner Konvention urheberrechtlich geschützt und Sie müssen davon ausgehen, dass es *illegal* ist diese zu kopieren - auch teilweise. Dies gilt auch, wenn die Arbeit frei verfügbar ist! Nicht für ein Dokument zu bezahlen bedeutet nicht, dass Sie Texte hieraus frei kopieren und in einer eigenen Arbeit veröffentlichen dürfen.

Borland InterBase-Handbücher

Die Borland InterBase 6 Beta docs - obwohl kostenlos - sind nicht Bestandteil des InterBase-Pakets, das im Juli 2000 Open-Source wurde. Wir haben mehrmals bei Borland angefragt, ob wir diese Dokumente nutzen könnten„ da sie unter die InterBase Public License fielen“, aber sie haben sich nicht mal die Mühe gemacht

zu antworten. Fühlen Sie sich frei, diese Dokumentation als Informationsquellen zu verwenden, aber kopieren Sie keine Texte hieraus.

PostgreSQL-Dokumente

PostgreSQL ist eine weitere wichtige Open-Source-Datenbank, mit (nicht überraschend) vielen Ähnlichkeiten zu Firebird, es gibt aber auch viele Unterschiede. Abhängig von der Art der Dokumentation, die Sie schreiben wollen, kann es vorteilhaft sein, sich auf bestehende PostgreSQL-Dokumente zu stützen. Beachten Sie aber, dass, wenn Sie PostgreSQL Material verwenden, MÜSSEN Sie deren Copyright in Ihrem Dokument verwenden!

Die Homepage der PostgreSQL-Dokumentationen ist hier:

http://www.postgresql.org/docs/

Die aktuelle PostgreSQL-Lizenz finden Sie hier:

http://www.postgresql.org/about/licence

Eine nette Sache der PostgreSQL-Dokus ist, dass sie ebenfalls in DocBook verfasst sind, genau wie unsere. Allerdings verwenden sie DocBook SGML anstelle von XML, somit werden bestimmte Einstellungen erforderlich sein. Die DocBook SGML Quellen finden Sie hier:

http://developer.postgresql.org/cvsweb.cgi/pgsql-server/doc/src/sgml/

Oder schauen Sie sich den gesamten CVS-Baum, Dokus und alles weitere an. Für weitere Anweisungen schauen Sie unter:

http://developer.postgresql.org/docs/postgres/cvs.html

Ihr Copyright und die PDL

Wenn Sie etwas zum Firebird Dokumention-Teilprojekt beitragen, wird Ihre Arbeit im Open-Source-Repository bei SourceForge einbezogen. Im Januar 2005 beschloss das Firebird doc-Team, die Dokumentationen unter der *Public Documentation License* zu veröffentlichen. Die Lizenzierung der Arbeiten unter der PDL bedeutet, dass Sie das Urheberrecht behalten, anderen Benutzern jedoch bestimmte Rechte gewähren:

- *Kostenlose Nutzung*: Jeder kann Ihre Arbeit verwenden und verteilen, kostenlos oder gegen Geld, solange die Lizenzhinweise intakt gehalten werden.
- *Recht auf Änderung*: Jeder kann Ihre Arbeit modifizieren und verteilen, solange alle modifizierten Versionen ebenfalls der PDL unterliegen, der ursprüngliche Lizenzhinweis beibehalten wird, und die Änderungen dokumentiert werden.
- *Größere Werke*: jeder darf Ihre Dokumentation in einem größeren Werk (modifiziert oder nicht) integrieren. Die größeren Arbeiten als Ganzes müssen nicht unter der PDL veröffentlicht werden, aber die Lizenzvoraussetzungen müssen für die PDL-lizenzierten Teile erfüllt sein.

Das Schöne an der PDL ist, dass sie die gleichen Rechte und Beschränkungen für die Nutzung unserer docs bietet wie es die IPL und IDPL (Firebird Code-Lizenzen) für den Firebird-Quellcode tun. Den vollständigen Text unserer Lizenz finden Sie über die Links in den Lizenzhinweisen weiter unten; der DocBook-Quellcode liegt unter `src/docs/firebirddocs/licenses.xml`

Wie wenden Sie die PDL auf Ihr Werk an?

Um Ihre Arbeit unter der PDL freizugeben, fügen Sie ein `appendix` mit dem Titel *License Notice* und folgendem Text an:

> The contents of this Documentation are subject to the Public Documentation License Version 1.0 (the „License“); you may only use this Documentation if you comply with the terms of this License. Copies of the License are available at http://www.firebirdsql.org/pdfmanual/pdl.pdf (PDF) and http://www.firebirdsql.org/manual/pdl.html (HTML).
>
> The Original Documentation is _*TITLE OF THE WORK*_.
>
> The Initial Writer of the Original Documentation is _INITIAL AUTHOR'S NAME_.
>
> Copyright (C) _YEAR(S)_. All Rights Reserved. Initial Writer contact(s): _EMAIL OR OTHER CONTACT ADDRESS(ES)_.

Alles, was wie _LIKE THIS_ aussieht, muss natürlich ersetzt werden. Wenn Sie nicht der ursprüngliche Autor sind, lassen Sie seine oder ihre Bekanntmachung intakt und fügen Sie den folgenden Text ein:

> Contributor(s): _NAME(S) + SHORT DESCRIPTION (COUPLE OF WORDS) OF CONTRIBUTION_.
>
> Portions created by _CONTRIBUTOR'S NAME_ are Copyright (C) _YEAR(S)_. All Rights Reserved. Contributor contact(s): _EMAIL OR OTHER CONTACT ADDRESS(ES)_.

Es kann diverse *Contributor's sections* (Hinweise auf Beitragende Autoren) in den Lizenzhinweisen geben.

Einfügen einer Dokumenthistorie

Wenn Ihr Beitrag aus mehr als einer einfachen Änderung oder Addition an einer Stelle besteht, fügen Sie einen Anhang (appendix) mit Namen *Dokumenthistorie* vor oder nach der Lizenz und Datenschutz ein. Wenn ein solcher Anhang bereits vorhanden ist, geben Sie eine immer eine Beschreibung Ihrer Modifikation(en) darin an. Bitte beachten Sie, dass selbst wenn es bereits eine Dokumenthistorie gibt, Sie noch einen Mitwirkendenabschnitt in den Lizenzhinweisen hinzufügen müssen - aber Sie können dann auch einen Hinweis „siehe Dokumenthistorie“ anstelle der Kurzbeschreibung angeben.

Wenn Sie der ursprüngliche Autor sind, dann ist es auch völlig in Ordnung, eine Dokumenthistorie in der ersten Version eines Dokuments für spätere Revisionen einzufügen. Vergleichen Sie das erste Revision-Element im Beispiel unten.

Gegenstand der Dokumenthistorie ist das Element `revhistory` mit seinen Unterelementen:

```
<revhistory>
  <revision>
    <revnumber>1.0</revnumber>
    <date>12 Sep 2005</date>
    <authorinitials>PV</authorinitials>
    <revdescription>
      <para>First version</para>
    </revdescription>
  </revision>
  <revision>
```

```
    <revnumber>1.1</revnumber>
    <date>5 Dec 2005</date>
    <authorinitials>PV</authorinitials>
    <revdescription>
      <para>Added information on COALESCE</para>
      <para>Corrected some spelling errors</para>
    </revdescription>
  </revision>
</revhistory>
```

Bitte kürzen Sie die Monatsnamen im Date-Element, da die Datumsspalte im PDF recht schmal ist.

Unten ist ein Beispiel zur Dokumenthistorie (Ausgabeansicht, nicht Quelle!), welches ein `revhistory`-Element darstellt. Beachten Sie die Verweise im CVS-Baum: Wir sind gesetzlich verpflichtet, alle Änderungen zu identifizieren und zu datieren. Aber da CVS dies bereits tut, können wir einfach den Benutzer hierüber informieren und eine weniger umfangreiche, aber schöner zu lesende Historie im Dokument selbst vorzeigen.

> Die exakte Dateihistorie wird im CVS-baum des `manual Modul` festgehalten; siehe http://sourceforge.net/cvs/?group_id=9028
>
> **Versionsgeschichte**
>
1.0	2003	IBP	First publication of the free Quick Start Guide.
1.x	June 2004	IBP	Donated to Firebird Project by IBPhoenix.
2.0	2004	PV	Downgraded to Firebird 1.0 Added Classic vs. Superserver section. Reorganised and corrected Disk Locations Table. Added (new) screenshots. Updated and completed information on Control Panel applets. Added extra examples to „Expressions involving `NULL`“. Various other corrections and additions.

Wenn Sie den DocBook-Quellcode dieses Leitfadens in Ihrem XML-Editor öffnen (`src/docs/firebird-docs/docwriting-howto.xml`), können Sie ganz einfach die Dokumenthistorie und die Lizenzhinweise kopieren und in ihr eigenes Dokument einfügen. Kopieren Sie nicht die oben angeführten Beispiele; kopieren Sie die realen Anlagen am Ende der Dokumente, und bearbeiten Sie diese.

Ein Copyright zum Beginn

Unsere Lizenz und Documenthistorie, erscheinen beide am Ende des Dokument. Wenn Ihre Urheberrechte offensichtlich direkt am Beginn ausweisen möchten, können Sie auch einen kurzen Copyright-Hinweis in der `xxxinfo` des Dokuments einfügen, so z.B.:

```
<bookinfo>
  <title...
  <author...
  <edition...
  <copyright>
    <year>2003</year>
    <year>2004</year>
    <holder>Tootsie Griff</holder>
  </copyright>
</bookinfo>
```

Solch ein Hinweis ersetzt nicht die Lizenzhinweise und / oder Dokumenthistorie - es ist ein Extra.

Anhängen der gesamten Pubic Documentation License

Anstelle der URL, können Sie auch die gesamte PDL in Ihrem Dokument anhängen. Dies kann besonders nützlich sein, wenn Ihre Arbeit ein Buch (`book`) oder langer Artikel (article) ist und Sie erwarten (oder hoffen), dass die Menschen es drucken und in Papierform verteilen werden. Auf einem kurzen Dokument wird die gesamte PDL ein wenig übertrieben wirken, aber es ist Ihre Entscheidung.

Sie können die PDL DocBook-Quellen unter `src/docs/firebirddocs/licenses.xml` erhalten. Bitte beachten Sie, dass nur der Abschnitt mit dem Lizenztext selbst (einschließlich der allgemeine Lizenzhinweis) zur PDL gehört. Die Einführung ist nicht Bestandteil der Lizenz.

Wenn Sie die PDL in Ihrem Dokument einbinden, können Sie die Leerstellen in Abschnitt 5.2 der Lizenz ausfüllen. Sie können sie aber auch lassen, wie sie sind (vorausgesetzt, Ihr Name steht im Lizenzhinweis) oder geben Sie einfach den „Initial Writer“ oder „den Urheberrechtshalter“ an.

Hinweise für Übersetzer

Die Übersetzung eines Dokuments ist eine Art der Bearbeitung. Wenn Sie also ein Übersetzer sind, sollten Sie folgende Dinge tun:

- Listen Sie sich selbst als Mitwirkender im Lizenzhinweis auf, mit einer Beitragsbeschreibung wie z. B. "Übersetzung ins Russische". Sie können den Lizenzhinweis in die Zielsprache übersetzen, wenn Sie wollen, aber Sie können es auch in Englisch lassen oder in beiden Sprachen führen.

- Fügen Sie ein Element `Revision` - in der Zielsprache - zur revhistory der Dokumenthistorie ein. Für die Revisionsnummer (`revnumber`), verwenden Sie die Zahl der Revision, die Sie übersetzt haben, gefolgt von einer Bindestrich und Ihrem Sprachcode, z. B. „`2.0-ES`“ oder „`1.1-fr`“:

```
<revhistory>
  ...previous revisions...
  <revision>
    <revnumber>1.1</revnumber>
    <date>5 Dec 2005</date>
    <authorinitials>PV</authorinitials>
    <revdescription>
      <para>Added information on COALESCE</para>
      <para>Corrected some spelling errors</para>
    </revdescription>
  </revision>
  <revision>
    <revnumber>1.1-fr</revnumber>
    <date>13 Déc 2005</date>
    <authorinitials>AM</authorinitials>
    <revdescription>
      <para>Traduction en français</para>
    </revdescription>
  </revision>
</revhistory>
```

- Fügen Sie ein `othercredit`-Element zur *xxx*`info` am Anfang des Dokuments ein, z.B.:

```
<articleinfo>
  <title>Guía de NULL en Firebird</title>
  <author>
```

```
    <firstname>Paul</firstname>
    <surname>Vinkenoog</surname>
  </author>
  <othercredit>
    <firstname>Víctor</firstname>
    <surname>Zaragoza</surname>
    <contrib>Traducción al castellano</contrib>
  </othercredit>
  <edition>22 de julio de 2005 - Versión de documento 2.0-es</edition>
</articleinfo>
```

Das `contrib`-Element für Mitwirkende beinhaltet die gleiche Information wie die Mitwirkende-Beschreibung in den Lizenzhinweisen, sollte aber immer in der Zielsprache verfasst sein.

Kennzeichnen Sie die Dokumentversion im `edition`-Element – stellen Sie sicher, dass es die gleiche wie in der Dokumenthistorie ist.

Übersetzung der PDL

Sie müssen die PDL nicht üebersetzen. Wenn Sie dies jedoch tun:

- Fügen Sie ein unabhängiges Dokument zum Dokumentensatz (set) Ihrer Sprache in einem Buch mit Namen *Licenses* hinzu. (Aber übersetzen Sie „Licenses“ in *Ihre* Sprache).
- In der übersetzten Einleitung der PDL, erklären Sie, dass nur die englische Version rechtlich bindend ist und fügen Sie einen Link zur englischen Version ein.
- In allen Lizenzhinweisen, die zur übersetzten Version der PDL führen, ist auch ein Link zur englischen Version zu hinterlegen und stellen Sie klar, dass nur die englische Version rechtlich bindend ist.

Optional können Sie die übersetzte PDL an Ihr Dokument anhängen, wenn Sie der Extraaufwand nicht stört.

Fügen Sie Ihr Dokument dem manual Modul hinzu

Wenn Ihr Dokument abgeschlossen ist, und Sie überprüft haben, dass es korrekt erstellt wird, möchten Sie es sicherlich in *manual Modul* hinzufügen. Wenn dies Ihr erster Beitrag zum Dokumentationsprojekt ist, werden Sie sicherlich verstehen, dass Sie Ihr Werk zuerst bei den Koordinatoren für die Überprüfung einreichen. Alternativ können Sie die HTML-Version auch auf einer Webseite zugänglich machen, sodass das Ergebnis auf der Mailingliste diskutiert werden kann. Danach - und vielleicht nach ein paar Korrekturen - kann das Dokument im Modul eingebunden werden. Wenn Sie die erforderlichen Rechte besitzen, können Sie dies selbst tun, wenn nicht, übernimmt das einer der Koordinatoren für Sie.

Anfordern der Commit-Berechtigungen

Um die Commit-Rechte zu erhalten benötigen Sie zunächst einen Account für SourceForge. Wenn Sie bisher keinen besitzen, registrieren Sie sich unter http://sourceforge.net/account/register.php. Schicken Sie dann eine Mitteilung an die firebird-docs Mailingliste mit Ihrem SF-Benutzernamen und der Bitte diesen zum Firebird-Projekt hinzuzufügen. Der Leiter des manual-Unterprojekt und diverse Firebird-Projektadmins verfolgen die Liste; Sie werden sich Ihrer Anfrage annehmen. Generell sollten Sie die Berechtigungen *nach* Ihrem ersten Beitrag anfordern, da die Entscheidungsträger etwas in der Hand haben wollen.

Die folgenden Phrasen sind gleichbedeutend:

- Projektmitglied sein.
- Commit-Rechte besitzen.
- Schreibenden und lesenden Zugriff für das Repository zu besitzen.

Was sollten Sie tun und was nicht, sobald Sie Commit-Rechte besitzen?

Sobald Sie als Projekt-Mitglied angenommen werden, haben Sie Schreibzugriff auf die gesamte Firebird-Repository, nicht nur auf das manual Modul. Es ist kein technisches Hindernis, Änderungen für andere Module zu committen - das `firebird2`-Kernmodul zum Beispiel, oder sogar das `CVSROOT` -Modul, in dem wichtige Projektinformationen gespeichert sind.

Sie haben sich sicherlich schon gedacht, dass dies *NICHT* Sinn der Sache ist. Halten Sie sich an folgende Regeln:

- Tätigen Sie *niemals* Eingaben zu anderen Modulen, es sei denn die Verantwortlichen fordern Sie ausdrücklich dazu auf.
- Committen Sie nur Arbeiten zum manual Modul, wenn es eine Ihnen zugewiesenen Aufgabe betrifft. Selbst dann ist es eine gute Praxis, Änderungen und Ergänzungen auf der Mailing-Liste anzukündigen, so dass die anderen Schreiber eine Chance haben, sich dazu zu äußern. Immerhin ist dies eine kollektive Anstrengung.
- Wenn Sie denken, dass ein neues Dokument oder ein Verzeichnis aufgenommen werden sollte, erstellen und committen Sie es nicht einfach, schlagen Sie es vor auf der Liste.

In der Praxis sind die Dinge ein bisschen entspannter als hier angegeben, insbesondere dann, wenn es Ihre eigenen Aufgaben betrifft. Wir wollen nicht, dass Sie sich beschränkt fühlen und sicherlich soll Ihnen nicht das Gefühl gegeben werden, dass man für jede kleine Änderung um Erlaubnis fragen muss. Aber wir möchten, dass Sie verantwortungsvoll handeln, und wir wollen von einander wissen, was wir tun. Außerdem ist einander zu kontakieren oft inspirierend. Gemeinsam können wir die Dinge am Laufen halten!

Committen Sie Ihre Arbeit

Selbst wenn Sie Projektmitglied sind, können Sie Änderungen Ihrer lokalen Kopie nur abgeben, wenn diese auch mit Ihrem SF-Login ausgecheckt wurde. Wenn Sie immer noch mit einer Kopie arbeiten, die anonym ausgecheckt wurde, müsen Sie erst einen frischen SSH-Checkout durchführen und die Änderungen erneut zuweisen. Danach führen Sie den Commit aus. Bitte vgl. Sie auch Docbuilding Howto, wenn Sie nicht mehr wissen, wie der SSH-Checkout durchzuführen ist.

Wenn zwischen Ihrem letzten Checkout oder Update einige Zeit vergangen ist, führen Sie erst ein Update vor dem Commit aus. Dies wird Ihre lokale Kopie mit dem Repository synchonisieren und die Anzahl möglicher Konflikte reduzieren.

Sobald Sie für einen Commit bereit sind, wechseln Sie in das manual-Verzeichnis. Wenn Sie command-line CVS verwenden, tippen Sie folgendes ein:

cvs update -d [nur wenn Sie erst ein Update durchführen wollen]

cvs add *`path/to/mydocument.xml`* [nur wenn ein neues Dokument vorhanden ist, dass noch nicht im CVS liegt]

cvs commit -m "*Kurze Informationsbeschreibung, die hier einzugeben ist*"

Nach dem `-m` und innerhalb der Anführungszeichen, ist eine kurze Information über diesen Commit anzugeben, z.B. `"Neue Funktionen zur API-Referenz hinzugefügt"` oder `"Fehler im isql-Handbuch gefixt"`.

Geben Sie Ihr SF-Kennwort ein, wenn es abgefragt wird, damit alle Ihre durchgeführten Änderungen - inlusive der Unterverzeichnisese - committed werden. Ihr CVS-Client weiß welcher Server zu kontaktieren ist; diese und andere Informationen sind in den `CVS`-Unterverzeichnissen gespeichert, die während des Checkout erstellt wurden.

Wenn Sie einen anderen CVS-Client verwenden, konsultieren Sie hierzu bitte die Dokumentation.

> **Wichtig**
>
> Nachdem ein neues Dokument hinzugefügt wurde, müssen Sie einen separaten Commit durchführen. Dies gilt für command-line CVS und die meisten (wenn nicht für alle) anderen CVS-Clients.

Veröffentlichung Ihres Dokuments auf der Firebird Website

Möchten Sie Ihr Dokument veröffentlichen, müssen erst die HTML- und PDF-Ausgaben generiert werden. Dieser Vorgang wird im Firebird Docbuilding Howto besprochen. Wir gehen in diesem Abschnitt davon aus, dass dies erfoglreich durchgeführt wurde.

Benennung der PDF-Datei

Die Build-Tools benennen die Dateien automatisch nach ihrer ID im obersten DocBook-Element. Wir ändern die Namen der Mehrseiten-HTML-Ausgabe nicht - diese Seiten werden primär für die Onlinebetrachtung genutzt, und das Ändern eines einzigen Dateinamens würde sofort diverse Links zerstören. PDFs hingegen werden häufig heruntergeladen und Dateinamen wie `qsg2.pdf` oder `ubusetup.pdf` in einem Downloadverzeichnis oder auf einem Desktop sind *wirklich* nicht hilfreich bei der Identifizierung als Firebird-Handbücher. Deshalb sind hier ein paar Richtlinien für die Dateinamen:

- Stellen Sie sicher, dass der Name das Wort `Firebird`, bevorzugt am Anfang, verwendet;
- Versuchen Sie den Dokumenttitel zu nutzen, aber halten Sie es kurz;
- Nutzen Sie Bindestriche („-“) um Wörter zu separieren;
- Wenn der Titel sehr lang ist, vermeiden Sie Teile wie „Handbuch“, „Leitfaden“, „Howto“ etc., es sei denn das Weglassen würde zu Konfusionen führen;
- Nutzen Sie die Dokumentsprache, verwenden Sie aber nur ASCII-Zeichen (keine Umlaute, etc.)
- Falls (und *nur* falls) die Anwendung der o.a. Regeln zu einem bereits vorhandenen Dateinamen einer anderen Sprache führen, fügen Sie die Dokumentsprache (oder ihre Abkürzung) hinzu.

Um die genannten Richtlinien zu verdeutlichen, werden einige existierende Beispiele unten aufgelistet:

- `Firebird-2.0-QuickStart.pdf`
- `Firebird-Security.pdf`

- `MSSQL-to-Firebird.pdf`
- `Firebird-Generator-Guide.pdf`
- `Firebird-nbackup.pdf`
- `Firebird-2.0-Schnellanleitung.pdf`
- `Firebird-1.5-Arranque.pdf`
- `Firebird-et-Null.pdf`
- `Firebird-nbackup-fr.pdf`
- `Firebird-su-Ubuntu.pdf`
- `Firebird-nbackup-nl.pdf`
- `Guia-Escrita-Firebird.pdf`
- `Firebird-1.5-BystryjStart.pdf`
- `Firebird-Perehod-s-MSSQL.pdf`

Einseitiges HTML-Dateien

Falls und wenn wir mit der Veröffentlichung von einseitigen HTML-Dateien auf der Website beginnen - mittels **build monohtml** erstellt - sollten wir die gleichen Namen wie die entsprechenden PDFs verwenden, aber natürlich mit der Erweiterung `.html`.

Hochladen der PDF

Wenn Sie Schreibrechte auf den Firebird Webserver besitzen, stellen Sie eine SFTP-Verbindung zu `web.firebirdsql.org` und laden Sie die Dateien hoch:

- `/srv/www/htdocs/pdfmanual` (English docs)
- `/srv/www/htdocs/pdfmanual/fr` (French docs)
- `/srv/www/htdocs/pdfmanual/ja` (Japanese docs)
- etc.

Release Notes werden in den entsprechenden Verzeichnissen abgelegt:

- `/srv/www/htdocs/rlsnotes`
- `/srv/www/htdocs/rlsnotes/fr`
- etc.

Wenn Sie keine Schreibrechte für den Server besitzen, fragen Sie jemanden, der diese besitzt. Falls Sie ein Projektmitglied sind, können Sie auch nach einem Benutzer und Kennwort für den Server fragen.

Hochladen von mehrseitigen HTML-Dateien

Stellen Sie sicher, dass Sie alle notwendigen Dateien hochladen: die HTML-Dateien, die zusammengenommen Ihr Handbuch (oder Handbücher) darstellen, das Stylesheet `firebirddocs.css` (wenn es seit dem letzten Upload geändert wurde), und das Unterverzeichnis `images` mit allen geänderten oder hinzugefügten Inhalten. Um alle Links funktionsfähig zu halten, kann es notwendig sein, das übergeordnete `book` ebenfalls neuzuerstellen und hochzuladen (oder gar das gesamte `set`). Laden Sie das gesamte Drumherum nach:

- `/srv/www/htdocs/manual` (English docs)
- `/srv/www/htdocs/manual/fr` (French docs)
- etc.

Warnung

Wenn in Frage kommende Seiten zu einem anderen Basis-Set gehören als die standardmäßigen `firebird-docs` (z. B. `papers` oder `rlsnotes`), dann platzieren Sie diese nicht in den erwähnten Verzeichnissen. Wir haben hierfür noch keine klaren Regeln definiert, aber mehrseitige HTML-Builds sollten nicht mit anderen Sets gemischt werden. Wenn diese Situation eintritt, melden Sie es bitte auf der firebird-docs Liste.

Aktualisieren des Firebird Dokumentationsverzeichnisses

Das Firebird Dokumentationsverzeichniss unter http://www.firebirdsql.org/index.php?op=doc ist ein PHP-Skript, dass die meisten Inhalte aus den auf dem Server liegenden Dateien holt. Wenn Sie existierende Dokumente aktualisiert haben, die bereits im Verzeichnis aufgeführt sind, ist nichts weiter zu tun, sofern Sie den Dateinamen nicht geändert haben. Wenn Sie jedoch ein neues Dokument oder eine neue Übersetzung erstellt haben, müssen sie dies dem Verzeichnis bekannt geben. Hier steht wie:

Wenn Sie ein komplett neues Dokument erstellt haben

1. Schauen Sie sich das Dokumentationsverzeichnis an und entscheiden Sie welche Kategorie am besten zu Ihrem Dokument passt. (Kategorien werden durch orange Köpfe gekennzeichnet.)

2. Verbinden Sie sich zum Server, gehen Sie in Verzeichnis`/srv/www/htdocs/doc` und werfen Sie einen Blick auf die Dateien, die mit `Cat_` beginnen. Öffnen Sie die Datei, die zu Ihrer gewählten Kategorie gehört.

3. Lesen Sie die Anweisungen am Anfang der Datei.

4. Erstellen Sie einen neuen Abschnitt, beginnend mit dem Titel des Dokuments in Englisch, durch eine speziell formatierte Zeile für jede verfügbare Version. Bei einem neuen Dokument könnte ein solcher Abschnitt wie folgt aussehen:

   ```
   Firebird Uninstallation Howto
   en:/manual/fb-uninstall.html
   en:/pdfmanual/Firebird-Uninstall.pdf
   ```

   Jede Versionszeile beginnt mit dem Sprachcode, gefolgt von einem Doppelpunkt, gefolgt von einer URL. Für Dokumente, die auf unserem eigenen Server liegen, ist diese URL einfach der „absolute“ Weg beginnend am Server-Root. Die Abschnitte werden durch Leerzeilen getrennt. Die Reihenfolge der Abschnitte in der Datei bestimmt die Reihenfolge der Dokumente innerhalb ihrer Kategorie im Dokumentationsverzeichnis auf der Webseite. Die Reihenfolge der Versionszeilen innerhalb eines Abschnitts ist irrelevant.

5. Speichern Sie die Datei. Wenn Sie diese auf Ihrem eigenen Computer bearbeiten, laden Sie sie zurück auf den Server. Aktualisieren Sie jetzt die Firebird Dokumentationsverzeichnis-Seite in Ihrem Web-Browser und überprüfen Sie, ob das Dokument dort aufgeführt ist, wo es sein sollte, und ob die Links funktionieren. Stellen Sie außerdem sicher, dass die Links in der korrekten Spalte sind (HTML in der mittleren Spalte, PDF etc. in der Spalte ganz rechts).

6. Das PHP-Skript macht einen ziemlich guten Job bei der automatischen Bestimmung des Dokumenttyps, aber es gibt Fälle, in denen es diesen falsch ermittelt. Wenn dies der Fall ist, fügen Sie den Dateityp - zwischen geschweiften Klammern - unmittelbar nach der URL in der Versionszeile ein

   ```
   en:http://www.ibphoenix.com/main.nfs?a=ibphoenix&page=ibp_60_sqlref{html}
   ```

7. Sobald alles funktioniert, committen Sie die aktualisierte Kategorie-Datei zum CVS. Wenn Sie das Firebird `web`-Modul vom SourceForge-Server ausgecheckt haben, finden Sie die Kategorie-Dateien (und mehr) im Ordner `web/website/doc`. Verwenden Sie Ihren SF-Benutzernamen und das zugehörige Kennwort für den Checkout, andernfalls werden Sie nicht in der Lage sein Ihre Änderungen zu committen. Die Arbeit mit CVS ist beschrieben im Firebird Docbuilding Howto.

Wenn Sie ein existierendes Dokument in eine neue Sprache übersetzt haben oder ein neues Dokument hierfür erstellt haben

1. Schauen Sie sich das Dokumentationsverzeichnis an und entscheiden Sie welche Kategorie am besten zu Ihrem Dokument passt. (Kategorien werden durch orange Köpfe gekennzeichnet.)

2. Verbinden Sie sich zum Server, gehen Sie in Verzeichnis`/srv/www/htdocs/doc` und werfen Sie einen Blick auf die Dateien, die mit `Cat_` beginnen. Öffnen Sie die Datei, die zu Ihrer gewählten Kategorie gehört.

3. Lesen Sie die Anweisungen am Anfang der Datei.

4. Suchen Sie den Abschnitt für das betreffende Dokument und fügen Sie die Versionszeile(n) für Ihre Anpassungen hinzu, z.B.:

```
Firebird Uninstallation Howto
en:/manual/fb-uninstall.html
en:/pdfmanual/Firebird-Uninstall.pdf
fr:/manual/fr/fb-uninstall-fr.html
fr:/pdfmanual/fr/Deinstaller-Firebird.pdf
```

   Die Reihenfolge der Versionszeile ist irrelevant, aber der Titel muss ganz oben bleiben.

5. Schritte 5, 6 und 7 sind die gleichen wie die für neue Dokumente.

Anhang A: Dokumenthistorie

Die exakte Dateihistorie kann über das manual Modul im CVS Baum abgerufen werden; http://sourceforge.net/cvs/?group_id=9028

Versionsgeschichte

0.1	17 Jan 2004	PV	Erster unvollständiger Entwurf, veröffentlicht unter dem Titel *Writing Documentation for Firebird* (aka *Firebird Docwriting Howto*).
0.2	27 Jan 2004	PV	Erste vollständige Version. (Erhielt Eingang in CVS am 31 Jan 2004)
1.0	8 Mär 2004	PV	Erstes offizielle Release auf der Firebird-Website.
1.1	26 Feb 2005	PV	*Die folgenden Änderungen wurden zwischen März 2004 and Februar 2005 durchgeführt:* Änderung des Titels zu *Firebird Docwriting Guide*. Abschnitt zu PostgreSQL docs hinzugefügt. Hinweis für nicht-DocBook-Mitwirkende hinzugefügt. Erklärung des Begriffs wohlgeformtes XML. DocBook-Vorteile klarer herausgestellt. Empfehlung von section-Elemente gegen sectN-Elementen getauscht. `xref` und anderen wenig verwendetes Zeug aus der Elementreferenz entfernt; `procedure` hinzugefügt. Info über nichtproportionales Literallayout aktualisiert. Abschnitt über PDL und wie die Lizenhinweise und Dokumenthistorien zu verwenden sind hinzugefügt. Einige kleinere Verbesserungen. Dokumenthistorie und Revisionsnummer hinzugefügt.. Lizenzierung dieser Arbeit unter der Public Documentation License.
1.1.1	8 April 2005	PV	Absatz über `titleabbrev`-Elemente hinzugefügt.
1.2	10 Feb 2006	PV	Alle <sectN>-Elemente in der Quellcodestruktur in <section> abgeändert. Änderung der docbuildhowto `links` in `ulinks`, da die Artikel absofort in separaten PDFs landen. *DocBook XML Characteristics*: Bemerkung über „plaintext“ entfernt. Hinweis zu XSLT hinzugefügt. *DocBook XML authoring tools*: in zwei Unterabschnitte aufgeteilt; Warnung bzgl. ConText UTF-8-Problem; Info zu SciTE hinzugefügt; Warnung bzgl. Speichern als 8-bit hinzugefügt; ersten Absatz über dedizierte XML-Editoren angepasst; Oxygen hinzugefügt; Altova Authentic entfernt; Aktualisierung/Anpassung der Informationen über Altova XMLSpy. *Writing your DocBook doc*: umbenannt in *Setting up your DocBook doc*; Änderung des zweiten Absatzes; dritten Absatz verschoben („Please read the subsection...“) nach *Elements we use frequently*;

			Anpassung der Links von „subsection on hierarchical elements“ zum neuen Absatz. *Creating the Document*: Set/Book Einleitung geändert; Master-Dok-Beispiel geändert; UTF-16-Hinweis hinzugefügt; Informationen über Platzierung von Dateien, die zu anderen Basis-Sets gehören. *Typing text*: Kleinere Änderungen des ersten und letzten Absatzes. *Elements we use frequently*: zum Hauptabschnitt befördert, eingegliedert nach *Setting up your DocBook doc*; Tip vor dem ersten Unterabschnitt zu normalem Absatz geändert, Änderung des ersten Satzes; Trennung von *Hierarchical elements* in Unterabschnitte, und Bearbeitung / Ergänzung von vielem Zeug; Unterabschnitt über HTML-Tabellen hinzugefügt; Starke Änderungen des Abschnitts „quote - literal“; Unterabschnitte über Bilder und Absätze hinzugefügt. *Non-DocBook aspects of the writing process*: verschwunden, alle Unterabschnitte wurden zu Hauptabschnitten befördert; ihr erster Absatz ist nun in *Language and style* zu finden. Bearbeitungen. *Copyrights*: Umbenannt in *Copyright issues* und Einleitungsabsatz eingefügt. *Respecting others' copyrights*: renamed *Using material written by others*. Der erste Absatz wurde in zwei geteilt und bearbeitet. Der Absatz über Borland docs wurde in einen Unterabschnitt ausgegliedert, die ersten Wörter entfernt. *Using PostgreSQL docs* ist nun ein Unterabschnitt in *Using material written by others*, und umbeannt in *PostgreSQL docs*. *Your copyright and the PDL*: starke Bearbeitung, Reorganisation der Unterabschnitte, und Ergänzungen. *Committing your work*: Ergänzung von **cvs add** command line und „Important“-Hinweis zum Committen nach dem Hinzufügen.
1.2.1	11 Mai 2006	PV	Start tag in bridgehead-Beispiel korrigiert (/ entfernt).
1.2.2	25 Jan 2007	PV	*Elements we use frequently*: Erwähnung der `title`-Option für Warnungen und Hinweise. Einleitungen für Übersetzer in einen `Hinweis` verschoben .
1.3	5 Mai 2007	PV	*Topics discussed in this guide*: Neuen Eintrag zur letzten Liste hinzugefügt. *Links*: Hinweis über das Ausgleichen von "hot zones" (behoben in FOP 0.93). *Program listings, screens, literal layout, and examples*: Hinweis über nicht-proportionale Schriften entfernt `literallayout`. Änderung der Ausgabe von `example` in ein Zitat. *HTML tables*: id zugewiesen. Änderung von *processing instructions* in ein `firstterm`. *PDF rendering of large tables*: Neuer Abschnitt. *Style*: Kleine Neuanordnung der Wörter im 3. Listeneintrag. *Publishing your document on the Firebird website*: Neuer Abschnitt. *License notice*: (C) 2004–2006 -> 2004–2007.
1.3-de	21. Jul 2013	MK	*Deutsche Übersetzung basierend auf der englischen Dokumentenversion 1.3.*

Anhang B: Lizenzhinweis

Die Original-Dokumentation trägt den Titel *Firebird Docwriting Guide*.

Der ursprünglich Autor der Original-Dokumentation ist: Paul Vinkenoog.

Übersetzung ins Deutsche: Martin Köditz.