

Projekte per DOM bearbeiten

5.1	Bestehende Projekte bearbeiten	79
5.2	Neue Projekte erstellen	85

Bisher haben wir uns angesehen, wie List & Label mit Ihren Daten bekannt gemacht werden kann und wie Sie bzw. Ihre Anwender dann mit Hilfe des Designers Layouts für die Datenausgabe erstellen können. Natürlich kann es Situationen geben, in denen eine solche Bearbeitung durch den Benutzer gar nicht erwünscht ist. Angenommen, Ihre Anwendung bietet dem Benutzer lediglich die Möglichkeit, ein Farbschema und Schriftartoptionen auszuwählen, der eigentliche Bericht soll aber nicht durch den Benutzer bearbeitet werden können.

Der aufmerksame Leser wird zunächst an die Formeln für Objekteigenschaften denken – ja, das wäre eine Möglichkeit. Der Nachteil ist allerdings, dass Sie immer daran denken müssen, für die Schriftart eine Formel zu verwenden. Dies kann die Pflege von Projekten etwas mühsam machen. Zudem gibt es Situationen, in denen sich z.B. die gewünschten Spalten einer Tabelle erst dynamisch aus den Daten ergeben. Spätestens dann wird es schwierig, die Projekte so zu gestalten, dass sie allen Anforderungen gewachsen sind.

Eine bequeme und mächtige Antwort auf diese Anforderungen stellt das DOM („Document Object Model“) von List & Label dar. Damit können Sie per Code auf alle Objekte und deren Eigenschaften zugreifen und auch völlig neue Projekte dynamisch erstellen. In diesem Kapitel wollen wir einen genaueren Blick auf diese Möglichkeiten werfen – Sie werden anhand einiger Beispiele sehen, was Sie mit Hilfe der DOM-Klassen alles erreichen können.

Diese Klassen finden Sie im Namensraum `combit.ListLabel?.?.Dom`. Das Klassensystem ist sehr umfangreich – auch hier würde eine komplette Referenz den Rahmen dieses Buchs sprengen. Die Online-Hilfe für die .NET-Komponente enthält aber eine ausführliche Dokumentation aller Klassen und Eigenschaften, so dass Sie hier bei Fragen stets schnell fündig werden (Abbildung 5.1).

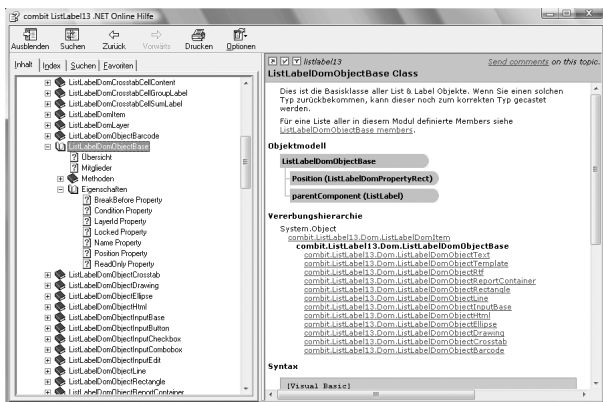


Abb. 5.1: Überblick über die DOM-Klassen in der Online-Hilfe

5.1 Bestehende Projekte bearbeiten

Beginnen wir dieses Kapitel mit einem neuen Beispiel. Erzeugen Sie also eine leere Anwendung und fügen Sie eine Referenz auf die List & Label Assembly hinzu. Als Nächstes benötigen Sie Zugriff auf den Namensraum für die DOM-Klassen, fügen Sie also (hier im Beispiel für Version 13)

```
using combit.ListLabel13.Dom;
```

ganz zu Beginn hinzu. Schließlich ziehen Sie noch eine Schaltfläche auf die Form, hinter deren Click-Ereignis Sie den Code in Listing 5.1 legen.

Listing 5.1: Erste Gehversuche mit dem DOM

```
// Bestehendes Projekt öffnen
ListLabelDomProjectList project = new
    ListLabelDomProjectList(new ListLabel());
project.Open(@"c:\programme\combit\1113\
    Beispielanwendung\artikel.lst",
    L1DomFileMode.Open,
    L1DomAccessMode.ReadOnly, true);

// Iteration über alle Objekte
foreach (ListLabelDomObjectBase obj in
    project.Objects)
{
    // Ausgabe des Objektnamens
    MessageBox.Show(obj.Name);
}
// Projekt schliessen
project.Close();
```

Sehen wir uns die Aufrufe der Reihe nach an. Zunächst wird die Projektklasse erzeugt. Diese benötigt eine Komponente vom Typ `ListLabel` als Elternkomponente.

Interessanter wird es beim Öffnen des bestehenden Projekts. Die `Open`-Methode bietet verschiedene Überladungen an. Die im Listing 5.1 gezeigte Variante ist die mächtigste – Sie können über die Parameter genau steuern, was passieren soll. Die möglichen Werte der `L1DomFileMode`-Aufzählung für den zweiten Parameter finden Sie in Tabelle 5.1.

Tabelle 5.1: Die verschiedenen Dateimodi der `Open`-Methode

Wert	Bedeutung
Create	Das Projekt wird neu erzeugt. Wenn schon eine Datei mit dem angegebenen Namen existiert, wird diese überschrieben.
CreateNew	Das Projekt wird neu erzeugt. Wenn schon eine Datei mit dem angegebenen Namen existiert, wird eine Exception geworfen.
Open	Das Projekt wird geöffnet. Wenn keine Datei mit dem angegebenen Namen existiert, wird eine Exception geworfen.
OpenOrCreate	Das Projekt wird neu erzeugt oder geöffnet. Wenn schon eine Datei mit dem angegebenen Namen existiert, so wird diese geöffnet, ansonsten wird die Datei angelegt.

Ihre Auswahl kombinieren Sie im dritten Parameter mit einem der Werte aus der `L1DomAccessMode`-Aufzählung (Tabelle 5.2).

Tabelle 5.2: Die verschiedenen Zugriffsmodi der Open-Methode

Wert	Bedeutung
ReadOnly	Das Projekt wird nur mit Leserechten geöffnet.
ReadWrite	Das Projekt wird mit Lese- und Schreibrechten geöffnet.

Der letzte Parameter schließlich erlaubt es, mögliche Syntaxfehler im Projekt zu ignorieren. Dies ist insbesondere dann praktisch, wenn Sie mehrere Projekte bearbeiten wollen, aber die benötigte Datenstruktur nicht kennen oder kennen müssen. Um lediglich eine Schriftart zu ändern, ist es ja nicht unbedingt erforderlich, vorher eine komplexe SQL-Abfrage abzusetzen.

Nach Aufruf der `Open()`-Methode steht Ihnen im Projektobjekt dann der Zugriff auf alle Möglichkeiten offen. In unserem Beispiel iterieren wir mit einer `foreach`-Schleife durch alle verfügbaren Objekte und geben deren Namen aus.

Die wichtigsten Eigenschaften der Projektklasse finden Sie in Tabelle 5.3.

Tabelle 5.3: Wichtige Eigenschaften der Projektklasse

Eigenschaft	Zweck
Layers	Bietet den Zugriff auf die Ebenen des Projekts.
Layout	Ermöglicht den Zugriff auf die Hilfslinien, das Ausrichtungsgitter etc.
Objects	Wichtigste Eigenschaft des Projektobjekts, hierüber können Sie auf die einzelnen Objekte des Projekts zugreifen.
Printers	Erlaubt es, die Drucker- und Seitenlayoutoptionen zu bearbeiten (Papiergöße und Ausrichtung, Druckernamen etc.).

Die Namen der Objekte angezeigt zu bekommen, ist ja schon schön und gut, in der Praxis interessieren Sie sich aber vermutlich eher für die Änderung einzelner Objekteigenschaften. Wir haben durch das letzte Beispiel gelernt, dass das Objekt mit der Überschrift „Überschrift“ heißt. Dieses Wissen wenden wir so gleich in der Praxis in Form von Listing 5.2 an.

Listing 5.2: Ändern von Objekteigenschaften

```
private void domButton2_Click(object sender, EventArgs e)
{
    // Projektklasse erzeugen und
    // bestehendes Projekt öffnen
    ListLabelDomProjectList project = new
        ListLabelDomProjectList(new ListLabel());
    project.Open(@"c:\programme\combit\
        1113\Beispielanwendung\artikel.lst",
        L1DomFileMode.Open,
        L1DomAccessMode.ReadWrite, true);

    // Textfont anpassen
    ListLabelDomObjectText text =
        (project.Objects["Überschrift"] as
        ListLabelDomObjectText);
    text.Paragraphs[0].Font.FaceName =
        "Comic Sans MS";

    // Projekt speichern
    project.Save();

    // Projekt schliessen
    project.Close();
}
```

Nachdem wir das Projekt diesmal mit Schreibrechten geöffnet haben, greifen wir über die Objektliste direkt auf das Objekt „Überschrift“ zurück. Da wir ja schon wissen, dass es sich um ein Textobjekt handelt, können wir das Objekt direkt in eine `ListLabelDomObjectText`-Instanz überführen. Zu guter Letzt greifen wir auf den ersten Absatz im Textobjekt zu und setzen die Schriftart auf 'Comic Sans MS'.

Profitipp

Ganz wichtig: Fast jeder Wert in List & Label ist eine Formel – dies ist ja für Sie nichts Neues mehr. Bei Verwendung der DOM-Klassen tut sich dadurch aber eine mögliche Fehlerquelle auf. Der Name der Schriftart im Beispiel muss unbedingt in Anführungszeichen gesetzt werden (auch einfache Anführungszeichen, wie im Listing gezeigt, sind erlaubt), um gültig zu sein! Ansonsten erhalten Sie einen Syntaxfehler beim Öffnen des Projekts. Nur in Anführungszeichen gesetzter Text stellt eine korrekte List & Label-Formel dar.

Um zu prüfen, ob der Code erfolgreich ausgeführt wurde, starten Sie die Beispielanwendung aus der Startmenügruppe. Wählen Sie `DESIGN | ARTIKELLISTEN` und wählen Sie die Datei „artikel.lst“ aus, die wir in obigem Beispiel verwendet haben. Wenn alles geklappt hat, sehen Sie im Designer die Früchte Ihrer Arbeit (Abbildung 5.2).

Analog zum Textobjekt können Sie auch die anderen Objekte bearbeiten. Für jeden Objekttyp steht eine spezialisierte, von `ListLabelDomObjectBase` abgeleitete Klasse zur Verfügung. Einen schnellen Überblick über die Objekte und deren Eigenschaften verschaffen Sie sich jederzeit mit Hilfe des DOM-Viewer, den Sie ebenfalls im Startmenü finden. Dieser Viewer zeigt das gesamte Objektmodell mit allen Unterobjekten und Eigenschaften an – eine wahre Fundgrube für eigene Experimente (Abbildung 5.3).

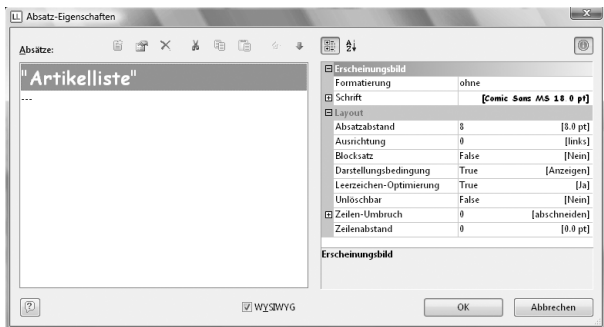


Abb. 5.2: Das Ergebnis unserer Bemühungen im Designer

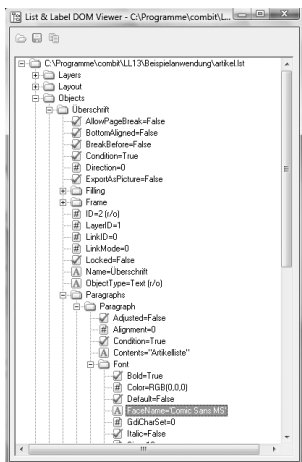


Abb. 5.3: Der List & Label DOM-Viewer