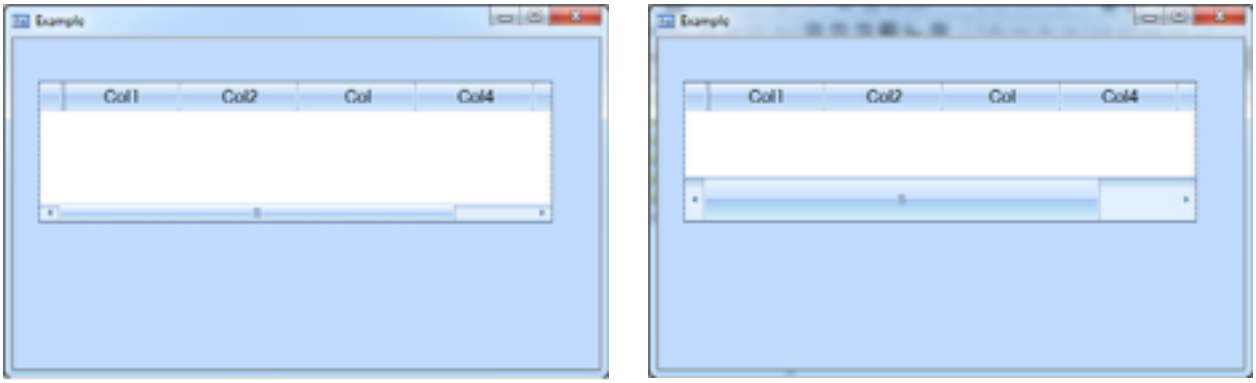
TechNotes

Aussehen des Scrollbars verändern



Die Abbildung zeigt die Breite (Höhe) des Scrollbars, wie er standardmäßig vom .Net Compiler umgesetzt wird. Rechts ist der über eine entsprechende XAML-Modifikation geänderte Scrollbar eines Grids.

Einleitung

Einer unserer Kunden hat mit SQLWindows eine **.Net-Anwendung** erstellt, die unter anderem auch auf Geräten mit „Touch"-Bedienung zum Einsatz kommt. Grundsätzlich, so wurde uns berichtet, funktioniere die Lösung auch wie geplant. Allerdings stellte sich bei der Fingerbedienung heraus, dass insbesondere die Standardbreite des (horizontalen und vertikalen) Rollbalkens eines Grids sehr schmal ist und daher bei der Fingerbedienung Schwierigkeiten beim Umgang mit diesem Bedienelement auftauchen können. Die Frage, die uns gestellt wurde, lautete daher: „Ist es möglich, den (automatisch eingeblendeten) Rollbalken in einem Grid in einer SQLWindows .Net-Anwendung zu verbreitern. Der Kunde formulierte auch seine Vermutung: „Ist doch eine WPF-Anwendung. Das müsste doch mit XAML gehen, oder?"

Lösungsansatz

Die Vermutung unseres Kunden war richtig, aber damit war selbstverständlich noch keine Lösung gefunden. Die Herausforderung bestand in der Umgestaltung des komplexesten Controls, dass Gupta Technologies in SQLWindows ausliefert: dem Grid. Der erste Gedanke war daher, das Grid umzugestalten. Dieser Gedanke wurde aber schnell verworfen, da der XAML-Code für das Grid während der Laufzeit und eben nicht beim Design erstellt wird. Zudem ist das Grid äußerst komplex und eine technische Dokumentation speziell der Umsetzung des Grids in XAML liegt uns leider nicht vor.

Allerdings, so die zweite Überlegung, wird vermutlich der Scrollbar, der im Grid angezeigt wird, wenn der Platz horizontal oder vertikal nicht ausreicht, um den Inhalt anzuzeigen, auf der Basis des Microsoft .Net Scrollbars aufgebaut sein. Sollte es nicht möglich sein, grundsätzlich das Aussehen des Scrollbars für die Touch-Bedienung für die ganze Anwendung durch Hinzufügung des entsprechenden XAML-Codes „umzugestalten“, dann wäre durch diese Vorgehensweise das Problem grundsätzlich für die gesamte Anwendung im Fall der Touch-Bedienung gelöst.

Lösung

Die Lösung des vom Kunden beschriebenen Problems bestand aus zwei Teilschritten:

1. Entwicklung und Test des entsprechenden XAML-Codes für die Umgestaltung des Scrollbars und
2. Das korrekte Deployment dieser Datei in der Entwicklungsumgebung

Umgestaltung („Perestroika“)

Die Vorgehensweise bei der Umgestaltung des Scrollbars bestand darin, den dem Gupta-Control zugrundeliegenden Microsoft .Net Control „Scrollbar“ umzugestalten. Damit ist gemeint, dass kein benannter Style — also kein Key-Attribut — definiert wurde, sondern ein Style, der sich auf das Microsoft Scrollbar bezieht (TargetType: {x:Type ScrollBar}). Der Vorteil dieser Vorgehensweise besteht darin, dass die Style-Definition grundsätzlich gilt und der Entwickler daher keine Zuordnung bei den einzelnen Controls im Attribut XAML-Style vornehmen muss.

```
<Application
 xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
  <Application.Resources>
   <Style x:Key="{x:Type ScrollBar}" TargetType="{x:Type ScrollBar}">
      <Style.Triggers>
        <Trigger Property="Orientation" Value="Horizontal">
          <Setter Property="Width" Value="Auto"/>
          <Setter Property="Height" Value="40" />
        </Trigger>
        <Trigger Property="Orientation" Value="Vertical">
          <Setter Property="Width" Value="40"/>
          <Setter Property="Height" Value="Auto" />
        </Trigger>
      </Style.Triggers>
    </Style>
  </Application.Resources>
</Application>
```

Das Aussehen des Scrollbars des Grids wird in der ganzen Anwendung durch eine Neudefinition des Styles des (Microsoft .Net) Scrollbars erreicht.

Durch diesen Style wird das Aussehen des Scrollbars — in diesem Beispiel lediglich in Bezug auf seine Breite — umdefiniert.

Deployment

Der zweite Schritt der Lösung besteht darin, die Datei unter dem richtigen Namen abzuspeichern und an der Stelle abzulegen, wo der .Net Compiler von SQLWindows derartige XAML-Dateien erwartet. Dieser Schritt ist zunächst ein bisschen knifflig, wenn man sich bisher noch nicht mit der XAML-Funktionalität und deren Umsetzung in SQLWindows beschäftigt hat[1]. Zunächst aber das Einfache: Die Style-Definition des Scrollbars wird unter dem Namen App.Xaml abgespeichert. Die Datei App-xaml[2] muss jetzt in ein spezielles Unterverzeichnis der Source Code-Datei gelegt werden.

Das Verzeichnis, in das die Datei App.xaml gelegt werden muss, existiert standardmäßig nicht. Der einfachste Weg, die notwendige Verzeichnisstruktur für die Quellcode-Datei anzulegen, besteht darin, im Hauptcode der SQLWindows-Anwendung ein Top-Level-Window in der Source-Code-Sicht zu markieren und aus dem Kontextmenü unter XAML die Option Edit auszuwählen. Der jetzt erschienene KAXAML-Editor kann sofort beendet werden. Durch den Aufruf von KAXAML im Edit-Modus wurden bereits die notwendigen (Unter-)Verzeichnisse erstellt.

Wenn jetzt beispielsweise mit dem Windows-Explorer in das Verzeichnis gewechselt wird, in der die SQLWindows Quellcode-Datei liegt, dann fällt dort das neue Verzeichnis .xaml auf. Im Verzeichnis unter .xaml liegt wiederum ein Verzeichnis, dessen Name der Name der Quellcode-Datei ist. In diesem Verzeichnis liegt eine Datei, die durch unser Arbeiten mit KAXAML angelegt wurde. Diese Datei sollte sofort gelöscht werden. Danach kann die App.xaml, die die Umgestaltung des Scrollbars enthält, in dieses Verzeichnis kopiert werden.

Das war’s …

Fazit

Innerhalb von kurzer Zeit — um genau zu sein, waren es drei Stunden für Planung, Umsetzung, Test und Dokumentation — konnte der Scrollbar des Grids in einer SQLWindows .Net-Anwendung nach den Anforderungen des Kunden umgestaltet werden. Der entwickelte Lösungsvorschlag beantwortet die Frage des Kunden. Tatsächlich ergeben sich aber in der Praxis weitere Fragen, wie beispielsweise „ich will nur eine .Net-Anwendung haben. Sie soll sich aber unterscheiden, wenn sie auf einem Touch-PC abläuft oder auf einem „normalen“ PC. Geht das auch?“ Selbstverständlich geht das auch, aber es sprengt den Rahmen dieses Papiers. In einem solchen Fall würde mit dynamischen Ressourcen gearbeitet werden, die nur in dem definierten Fall während der Laufzeit dazu gebunden würden. Aber das ist eine andere Frage …

---

1 Die Vorgehensweise wird in dem Manuskript SQLWindows & XAML ausführlich ab Seite 42 beschrieben.

2 In einem produktiven Projekt würde man vermutlich die einzelnen Style-Definitionen in eigenen XAML-Dateien ablegen und diese durch Verwendung von MergeDictionaries in der App.xaml-Datei für die Anwendung zusammen führen.

Es könnte sich auch ergeben, dass die Form und die Größe von Pushbuttons in einer Touch-Anwendung angepasst werden müssen. Dieses Problem lässt sich analog zur beschriebenen Vorgehensweise lösen.