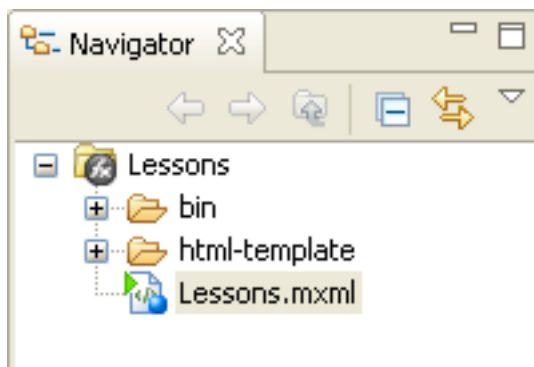


# Einstieg in Flex 2

## Ihre erste Anwendung

### Das Projekt **Lessons** anlegen

1. Starten Sie Flex Builder und wählen Sie im Menü **File > New > Flex Project**.
2. Im sich öffnenden Dialog akzeptieren Sie die erste Option **Basic** und klicken auf **Next**.  
Der nächste Dialog fragt nach dem Projektnamen und dem Speicherort für die Dateien.
3. Geben Sie in das Eingabefeld **Project Name** als Name **Lessons** ein.  
Wenn Sie ein Projekt erzeugen, generiert Flex Builder eine MXML Applikation basierend auf dem Projektnamen. Weil auch der Dateiname der Applikation denselben Namen verwendet, sind Leer- und Sonderzeichen hier nicht erlaubt.
4. Im Bereich **Project Contents** stellen Sie sicher, dass die Option **Use Default Location** angehakt ist. Der Standardspeicherort für Ihre Projektdateien lautet:  
Windows: **C:\Dokumente und Einstellungen\Benutzername\Meine Dateien\Flex Builder 2**  
Mac: **\Benutzer\Benutzername\Dokumente\Flex Builder 2**
5. Klicken Sie auf **Finish**.  
Flex Builder erzeugt ein neues Projekt und zeigt es im **Navigator** (oben links) an.



Der Projektassistent generiert automatisch die Projekt-Konfigurationsdateien, den Ausgabeordner (bin), in den Ihre kompilierten SWF Dateien abgelegt werden und die zentrale Anwendungsdatei Lessons.mxml.

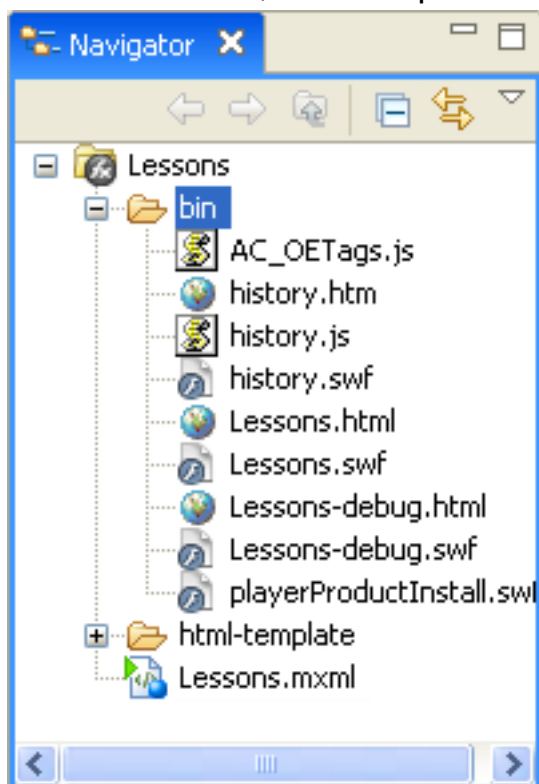
6. Versichern Sie sich, dass die Option **Build Automatically** (im Menü **Project**) aktiviert ist. Dies ist zwar die Standardeinstellung von Flex Builder, jedoch nicht in der Plug-In-Konfiguration.

## Wie man in Flex Builder eine Anwendung kompiliert

Bevor Sie in Flex Builder eine Anwendung kompilieren und ausführen, ist es hilfreich, das Basiskonzept zu kennen.

Standardmäßig kompiliert – oder baut – Flex Builder (in der Standalone-Version) die Anwendung, wenn Sie eine Datei zum Projekt hinzufügen oder ein Projekt speichern, nachdem Sie es in Flex Builder editiert haben. In der Plug-In-Version müssen Sie diese Option im Menü **Project > Build Automatically** aktivieren.

Nach dem Kompilieren erstellt Flex Builder die resultierende Flash-Datei (SWF) im Ordner **bin**, dem Standardordner, in den kompilierte Dateien abgelegt werden.



Flex Builder erstellt außerdem eine HTML-Wrapper-Datei für die SWF-Datei, in unserem Beispiel Lessons.html, für den Fall, dass die SWF-Datei in einem Browser aufgerufen werden soll.

**Hinweis:** Eine Flex 2 Anwendung läuft in einem Browser nur ab dem **Flash Player 9**.

Mit dem Projekt erzeugt Flex Builder die Hauptanwendungsdatei als MXML-Datei mit dem Parent Tag `<mx: Application>`. Ihr Projekt kann theoretisch mehrere Applikationsdateien enthalten, aber immer nur eine kann als Hauptapplikationsdatei festgelegt werden, empfohlen wird jedoch, immer nur eine Applikationsdatei je Projekt zu verwenden.

Soviel zum Basiskonzept des Kompilierens, nun können Sie eine kleine Anwendung in Flex Builder erzeugen und ausführen.

## Eine Anwendung erstellen und ausführen

Die Schritte in diesem Abschnitt setzen voraus, dass Sie das zuvor beschriebene Projekt **Lessons** bereits erstellt haben.

1. Falls die Datei Lessons.mxml noch nicht geöffnet ist, öffnen Sie sie im Navigator mit einem Doppelklick.
2. Wechseln Sie ggf. in die Ansicht **Source** mit Klick auf die gleichnamige Schaltfläche in der Dokumenttoolbar.



Flex Builder hat bei der Dateigenerierung zugleich den folgenden Code erzeugt:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute">
</mx:Application>
```

3. Fügen Sie einen Panel-Container ein, indem Sie den folgenden Code nach dem öffnenden Tag `<mx:Application>` eingeben:

```
<mx:Panel title="My Application" width="200" height="300">
</mx:Panel>
```

Panel-Container sind die grundlegenden Bausteine vieler Flex-Layouts.

4. Fügen Sie ein Label-Control hinzu, indem Sie den folgenden Code zwischen die öffnenden und schließenden `<mx:Panel>`-Tags setzen:

```
<mx:Label text="Welcome to Flex!" mouseDownEffect="WipeRight"/>
```

**Tipp:** Sie können Ihr Layout vorab ansehen, wenn Sie die Schaltfläche **Design** in der Dokumenttoolbar anklicken (s. Abb. zu Ziff. 2).

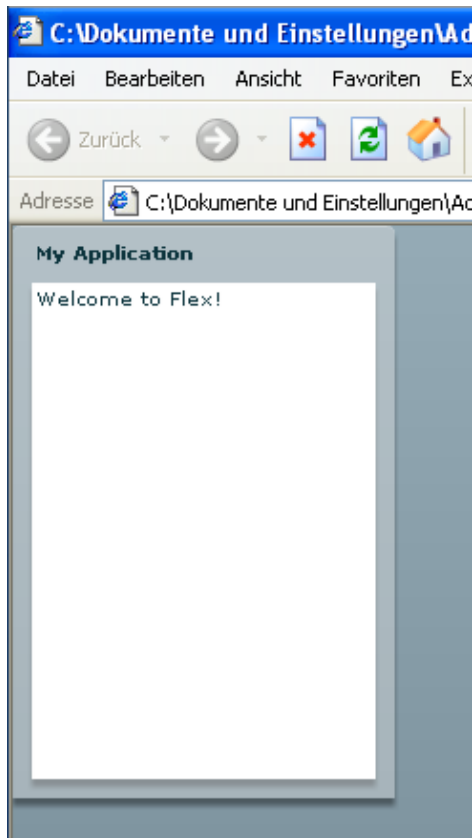
5. Speichern Sie die Datei.

Flex Builder erstellt beim Speichern automatisch die Anwendung. Sie können den Erstellungsprozess an der Anzeige in der unteren rechten Fensterecke ablesen. Während dieses Vorgangs können Sie weiterarbeiten.

6. Nachdem der Erstellungsprozess abgeschlossen ist, klicken Sie auf die Schaltfläche **Run** in der Toolbar, um die Anwendung zu starten.



Es öffnet sich Ihr Browser und startet die Anwendung.



**Hinweis:** Der Browser muss Flash Player 9 installiert haben, um die Anwendung ausführen zu können. Sie konnten bei der Installation von Flex Builder festlegen, zu welchen Browsern Flash Player 9 installiert werden sollte. Einen Browser mit Flash Player 9 wählen Sie in Flex Builder im Menü [Window > Preferences > General > Web Browser](#) aus.

7. Klicken Sie auf den Text „Welcome to Flex!“, um den WipeRight-Effekt zu sehen.

Sie können Ihre Anwendung veröffentlichen, indem Sie die von Flex Builder erzeugte SWF-Datei auf einen Webserver hochladen. Sie können auch die HTML-Datei hochladen (Lessons.html), um die Anwendung in einem Browser auszuführen. Die Dateien sind im Ordner [bin](#) innerhalb Ihres Projektordners abgelegt.

In dieser Lektion haben Sie gelernt, wie man in Flex Builder ein Projekt anlegt, kompiliert und ausführt. Mehr zu diesem Thema erfahren Sie in der Hilfe zu Flex Builder unter:

[Using Flex Builder 2 > Building Projects](#).

# Programmierbeispiele

## Tree und XML

1. Erstellen Sie ein neues Flex-Projekt (Menü **File > New > Flex Project**) namens XMLTree .
2. Wechseln Sie in die Ansicht **Design** und erstellen Sie – in dieser Reihenfolge – folgendes Layout:

a) **Panel** Title=Tree und XML Layout=absolute Width=400 Height=300  
paddingTop=10 paddingLeft=10 paddingRight=10 paddingBottom=10

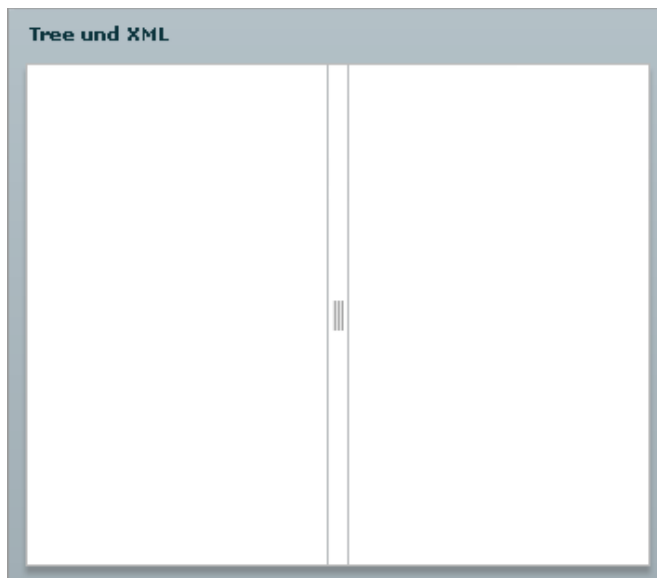
b) **Label** Text=Klicken Sie auf einen Eintrag in der Baumansicht.  
Width=100% Color=blue

c) **HDividedBox** Width=100% Height=100%

d) **Tree** ID=myTree Width=50% Height=100% On change=treeChanged(event)

e) **TextArea** Width=50% Height=100%

Das Layout sollte nun etwa so aussehen:



**Tipp:** Alle vorgenannten Eigenschaften lassen sich über den Bereich **Flex Properties** vornehmen. Neben der **Standard View** mit den am häufigsten verwendeten Eigenschaften gibt es auch noch die **Category View** und die **Alphabetical View**.

### 3. Wechseln Sie in die Ansicht **Source**. Der Quellcode sollte nun etwa so aussehen:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute">
  <mx:Panel layout="absolute" title="Tree und XML" width="75%" height="75%"
paddingTop="10" paddingLeft="10" paddingRight="10" paddingBottom="10">
  <mx:HDividedBox width="100%" height="100%">
    <mx:Tree width="50%" height="100%"></mx:Tree>
    <mx:TextArea width="50%" height="100%">
  </mx:HDividedBox>
</mx:Panel>
</mx:Application>
```

### 4. Setzen Sie vor `<mx:Panel>` die folgende XML-Struktur:

```
<mx:XMLList id="treeVPS">
  <node label="VPS IT-ZVG">
    <node label="Leitung">
      <node label="Andreas Dormann"/>
      <node label="Reiner Lange"/>
    </node>
    <node label="Entwicklung">
      <node label="Rainer Sievers"/>
      <node label="Helge Spang"/>
      <node label="Thomas Bäsecke"/>
      <node label="Herr Bartsch"/>
      <node label="Gertrud Korb"/>
    </node>
    <node label="Dokumentation">
      <node label="Martina Hutmacher"/>
      <node label="Rainer Busch"/>
    </node>
    <node label="Qualitätssicherung">
      <node label="Ellen Rabl"/>
      <node label="Peter Müller"/>
    </node>
    <node label="2nd Level Support">
      <node label="Johannes Köhler"/>
    </node>
  </node>
</mx:XMLList>
```

**In einer echten Anwendung werden XML-Daten natürlich von externen Dateien oder aus Datenbanken gelesen. Aber das kommt später ;-)**

## 5. Ergänzen Sie `<mx:Tree>` um die fettgedruckten Angaben:

```
<mx:Tree id="myTree" width="50%" height="100%"
labelField="@label" dataProvider="{treeVPS}" />
```

Damit verbinden Sie das Tree-Control mit der XML-Liste (anhand der ID **treeVPS**).

Die Daten, die das Control anzeigen soll, weisen Sie über **labelField** zu.

**@** kennzeichnet die Daten als Attribut des jeweiligen XML-Knotens.

Damit beim Anklicken auch etwas passiert, nämlich der angeklickte Knoten im TextArea-Control angezeigt wird, müssen wir nun ein wenig *ActionScript* schreiben.

## 6. Setzen Sie nach `<mx:Application>` und vor `<mx:XMLList>` folgenden Code:

```
<mx:Script>
  <![CDATA[

    [Bindable]
    public var selectedNode:XML;

    // Ereignis-Behandler für das Tree-Control Change-Ereignis
    public function treeChanged(event:Event):void {
      selectedNode=Tree(event.target).selectedItem as XML;
    }

  ]]>
</mx:Script>
```

Sie haben nun Ihr erstes *ActionScript* geschrieben. Genauer gesagt: Sie haben die öffentliche, an Komponenten anbindbare `[Bindable]` Variable **selectedNode** als XML-Typ deklariert und die Funktion **treeChanged** geschrieben, die dieser Variablen den ausgewählten Knoten übergibt.

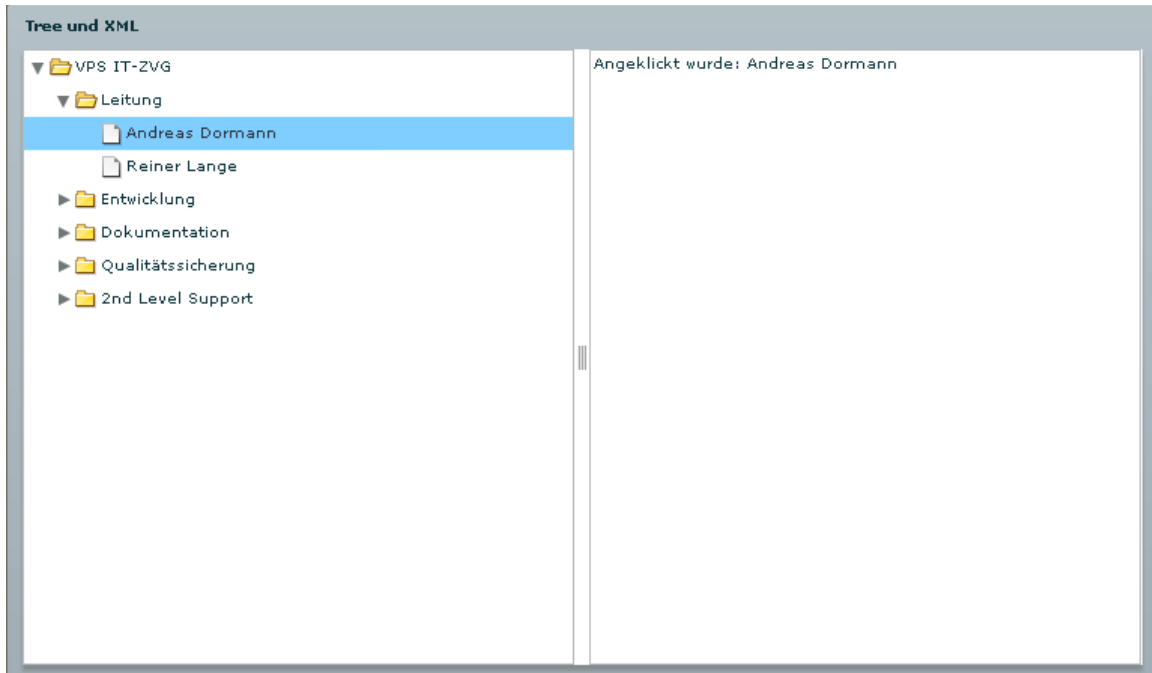
Damit alles zusammen funktioniert, müssen wir die Funktion noch mit dem Tree-Control verbinden und das TextArea-Control anweisen, das Ergebnis anzuzeigen.

## 7. Ergänzen Sie `<mx:Tree>` und `<mx:TextArea>` um die fettgedruckten Angaben:

```
<mx:Tree id="myTree" width="50%" height="100%" labelField="@label"
  dataProvider="{treeVPS}" change="treeChanged(event)" />
<mx:TextArea width="50%" height="100%"
  text="Angeklickt wurde: {selectedNode.@label}" />
```

## 8. Speichern Sie das Projekt und führen Sie die Anwendung aus.

Das Ergebnis sollte – nach Anklicken der Tree-Elemente – etwa so aussehen:



Herzlichen Glückwunsch! Sie haben erfolgreich ein Tree-Control mit XML-Daten verbunden.

Wir wollen zum Abschluss dieser Übung noch eine kleine Ergänzung vornehmen. Dazu machen wir einen Ausflug in die Hilfe von Flex Builder.

9. Wählen Sie im Menü [Help > Help Contents](#).

Wie Sie sehen, bietet Ihnen Flex Builder eine Fülle von Dokumentationen an.

10. Klicken Sie auf [Adobe Flex 2 Language Reference > All Classes > Tree](#).

Hier können Sie nun alle Informationen zum Tree-Control nachlesen.

11. Suchen Sie über [Properties](#) die Eigenschaft `editable`.

Die Dokumentation verrät uns, dass das Tree-Control seine Elemente editierbar machen kann.

Dazu müssen wir die standardmäßig auf **false** gesetzte Eigenschaft auf **true** setzen.

12. Ergänzen Sie das Tree-Control um folgenden Code:

```
<mx:Tree id="myTree" width="50%" height="100%" labelField="@label"
  dataProvider="{treeVPS}" change="treeChanged(event)" editable="true"/> >
```

13. Speichern Sie das Projekt und starten Sie die Anwendung. Sie können nun jedes Tree-Element bearbeiten.

