

# Multimedia Engineering II - Übung 2

---

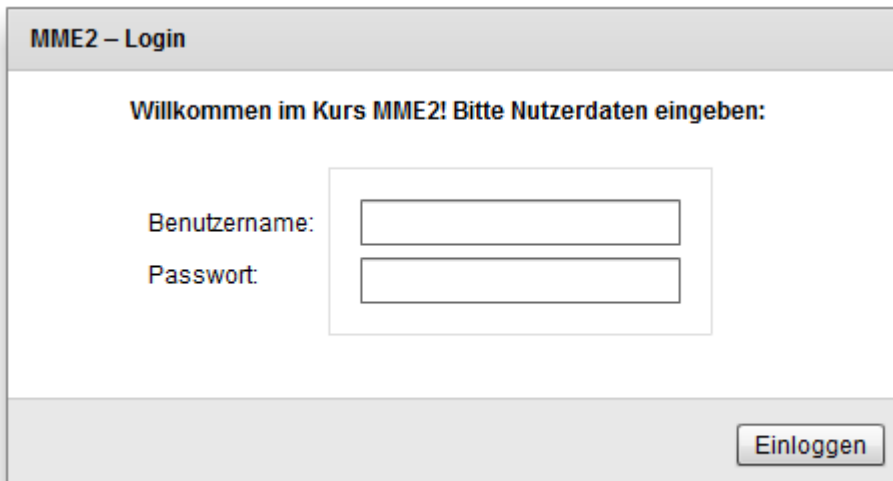
## Zielstellung der Übungsaufgabe

Das Login-Panel der ersten Übung erhält nun die Funktion, auf eine zweite „View“ zu wechseln. Auf dieser werden Sie nun das erste Mal einen Datenabruf einer XML für Ihre Anwendung realisieren.

## Bisheriger Stand

In der ersten Übung haben sie ein Login-Panel erstellt, welches bei Erfolg eine Meldung anzeigt.

Das Panel sollte ungefähr so aussehen:



The image shows a web-based login interface. At the top, there is a header bar with the text "MME2 – Login". Below this, a message reads "Willkommen im Kurs MME2! Bitte Nutzerdaten eingeben:". Underneath the message, there are two input fields: one for "Benutzername:" and one for "Passwort:". Both fields are contained within a single light gray box. At the bottom right of the interface, there is a button labeled "Einloggen".

## Informationen

### Event-Handling

#### Inline Script

Direkte Zuweisung eines AS Code als String, welches ausgeführt wird, sobald das Ereignis eintritt.

```
<s:Button click="Alert.show('clicked')">
```

#### AS Function aufrufen

```
<fx:Script>
    <![CDATA[
        protected function onClick(e:MouseEvent):void
        {
            Alert.show('Button geklickt: ' + e.target);
        }
    ]]>
</fx:Script>

<s:Button click="onClick(event)"/>
```

#### Eigene Events

Mit Hilfe der *dispatchEvent()* Methode können Events gefeuert werden. Diese Methode steht sämtlichen Komponenten zur Verfügung.

z.B. click Event eines Buttons manuell feuern:

```
var customEvent:MouseEvent = new MouseEvent(MouseEvent.CLICK);
loginButton.dispatchEvent(customEvent);
```

Natürlich kann auch eine eigene Event Klasse benutzt werden. In unserem Beispiel wird erst mal nichts anders als ein Event Name definiert.

```
package de.beuth.events
{
    import flash.events.Event;

    public class CustomEvent extends Event
    {
        public static const LOGIN:String = "login";

        public function CustomEvent(type:String, bubbles:Boolean=false,
cancelable:Boolean=false)
        {
            super(type, bubbles, cancelable);
        }
    }
}
```

Die Komponente, die diesen Event feuern will, macht den Event über ein sogenanntes Metatag bekannt:

```
<fx:Metadata>
    [Event (name="login", type="de.beuth.events.CustomEvent")]
</fx:Metadata>
```

Die Handler Funktion kann im MXML gesetzt werden:

```
<view:LoginPanel id="loginPanel" login="onLogin(event)"/>
```

oder auch mit Hilfe der *addEventListener* Funktion:

```
loginPanel.addEventListener(CustomEvent.LOGIN, onLogin);
```

## States

Beschreiben einzelne Zustände einer Anwendung oder Komponente

States können für jede Komponenten und auch für die Anwendung (<s:Application>) selber angelegt werden

Beliebig viele States können mit Hilfe der *State*-Klasse definiert, allerdings kann immer nur ein State zur selben Zeit angezeigt werden

Jede Komponente bzw. Anwendung besitzt ein *Base State*

Zustände können sich laufend und zeitgleich ändern

- durch Nutzerinteraktion
- durch das Eintreffen von serverseitigen Daten

Jeder State ist in der Lage

- Komponenten hinzuzufügen/entfernen, sowie deren Styles und Eigenschaften zu verändern
- Event-Listener zu definieren (bei Aufruf bzw. Verlassen eines States)

States können abgeleitet bzw. vererbt werden, mit Hilfe der *basedOn* Eigenschaft:

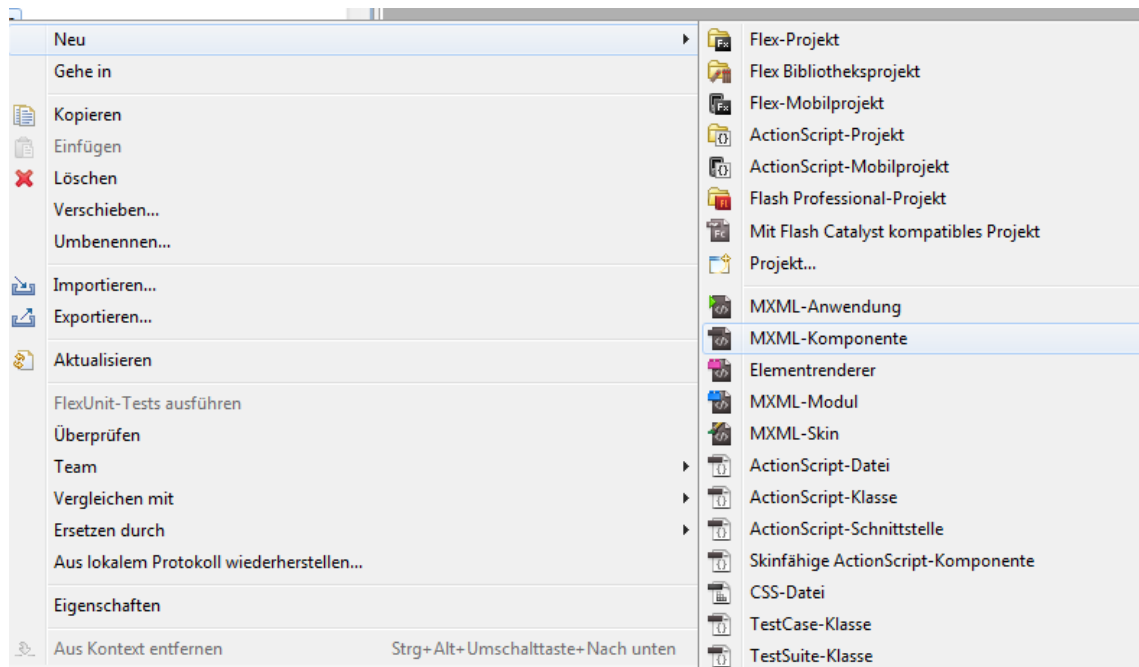
```
<s:states>
    <s:State name="State1"/>
    <s:State name="State2"/>
    <s:State name="State3" basedOn="State1"/>
</s:states>
```

Komponenten können mit Hilfe der *includeIn* Anweisung für einen oder mehrere States aktiviert werden. Das Gegenstück ist *excludeFrom*, um eine Komponenten explizit für bestimmte States auszuschließen.

```
<mx:Panel includeIn="State1"/>
<mx:Panel2 includeIn="State2"/>
```

## Teil 1: Anlegen von View-States

Erstellen sie eine eigene MXML Komponente LoginPanel und fügen sie diese der Hauptanwendung zu.



Erstellen sie eine weitere view Komponente (MainView), welche die eigentliche Oberfläche der Anwendung repräsentiert. Dieser Komponente fügen sie eine Standard *ComboBox*-Komponente( `s:ComboBox` ) hinzu, die später unsere xml Daten abbilden soll.

Erzeugen sie in der Application (die allererste MXML) zwei View-States *LoginState* und *ApplicationState*:

```
<s:states>
    <s:State name="LogInState" />
    <s:State name="LoggedInState" />
</s:states>
```

Die beiden zuvor erstellten Komponenten (LoginPanel und MainView) können Sie nun der Application hinzufügen. Nutzen Sie dafür den Komponenten-Browser im Design-Modus oder eine MXML-Anweisung. Um die Komponenten manuell hinzuzufügen importieren Sie den entsprechenden Namespace (z.B. `xmlns:view="de.beuth.view.*"`) und dann finden Sie unter `<view:MainView/>` ihre Komponenten.

Als nächstes erzeugen sie eine eigene Event Klasse und benutzen sie ein *login* Event, um der Anwendung mitzuteilen, dass der Login erfolgreich war.

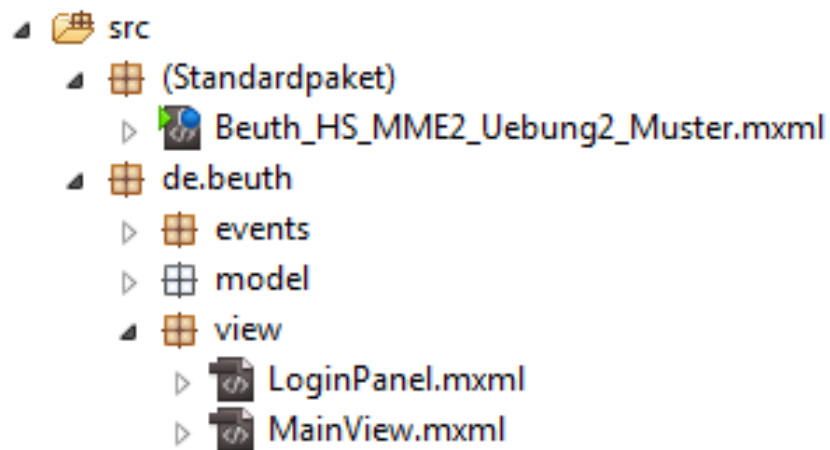
Nach erfolgreichem Login soll die Anwendung in den *LoggedInState* schalten.

### Tipps:

Mit Hilfe der *currentState* Eigenschaft der Komponenten können sie die States aktivieren.

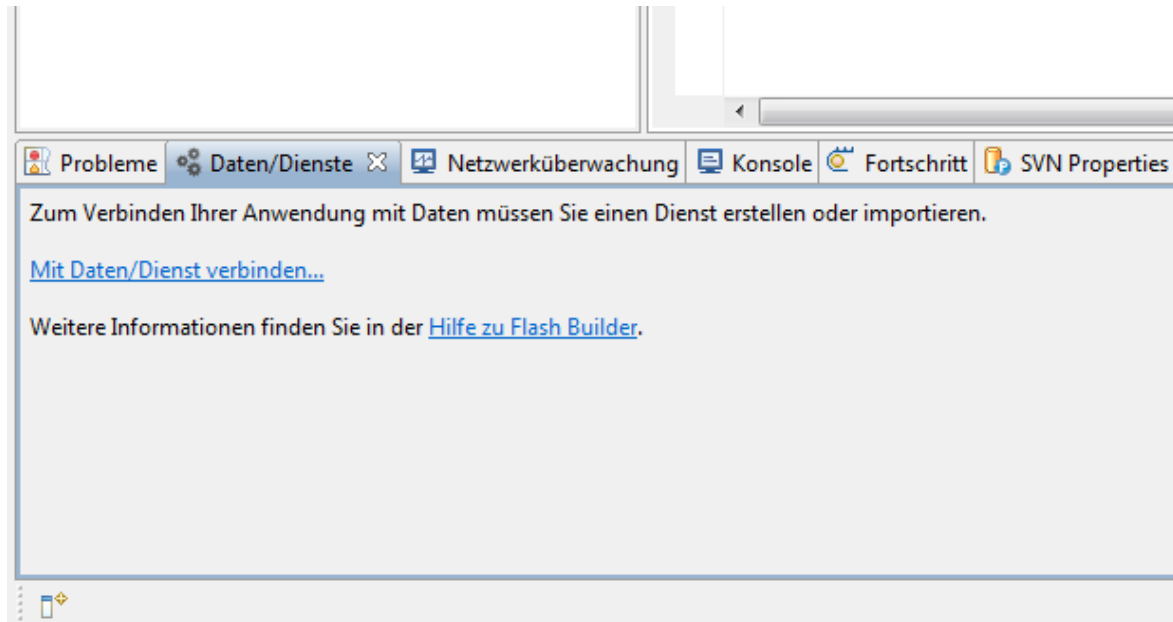
```
currentState='LoginState';
```

Bitte erzeugen sie eine übersichtliche Package Struktur, die einer MVC Struktur entsprechen soll:

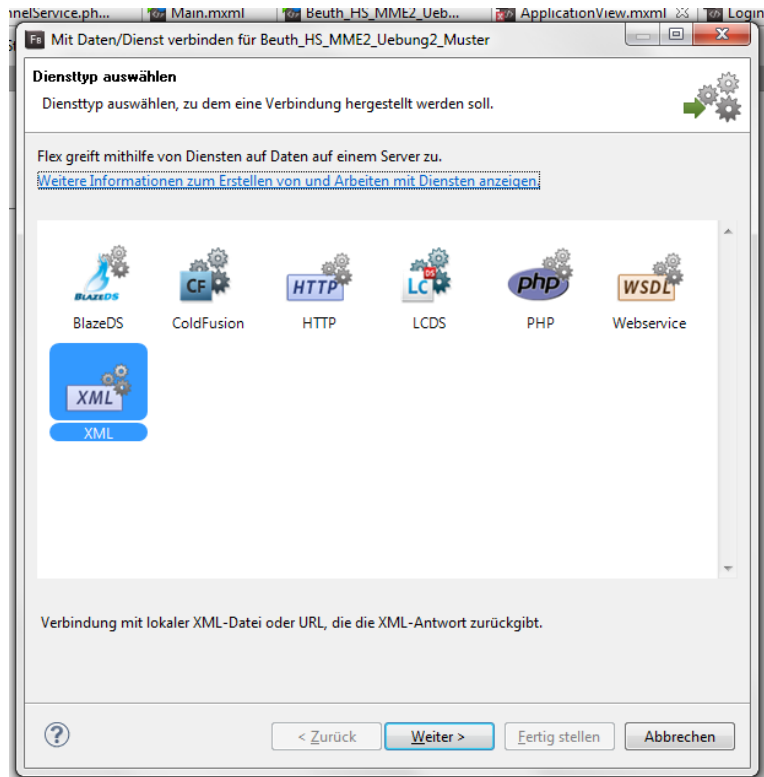


## Teil 2: Abruf von XML-Daten

Um nun der im vorherigen Schritt erstellten ComboBox-Komponente den XML-Inhalt zuzuweisen, benutzen wir den automatischen Dienst-Generator von Flash Builder 4.



Klicken Sie hierfür auf "Mit Daten/Dienst verbinden..." und wählen Sie "XML" im folgenden Fenster als Quelle aus.



Anschließend werden Sie nach dem Pfad zur XML gefragt. Hier kann entweder der Pfad zu einer lokal-gespeicherten XML angegeben werden oder eine XML im Internet. (Natürlich kann auch eine aus einem Skript generierten XML angegeben werden.)

Verweisen Sie auf folgende URL:

[http://141.64.64.105/beuthbox\\_reloaded/bb\\_amfphpsvn/services/1\\_categories.xml](http://141.64.64.105/beuthbox_reloaded/bb_amfphpsvn/services/1_categories.xml)

Nach einem Klick auf "Aufrufen" wird die Datei automatisch eingelesen und Sie haben die Möglichkeit, den passenden Knoten für Ihren Dienst auszuwählen.

In diesem Fall ist es der Knoten "cat", welcher automatisch als Array erkannt wird. Als letztes müssen Sie noch einen passenden Namen für Ihren Dienst definieren, die Pakete für die generierten Dateien angeben und "Fertig stellen" klicken.

**XML-Dienst konfigurieren**

ⓘ Eine Methode namens „getData“ mit dem Rückgabebetyp „cat[]“ wird erstellt.

XML-Quelle: ☐ Lokale Datei ☒ URL

URL:

Knoten auswählen:  ☒ Ist Array?

**Dienstdetails**

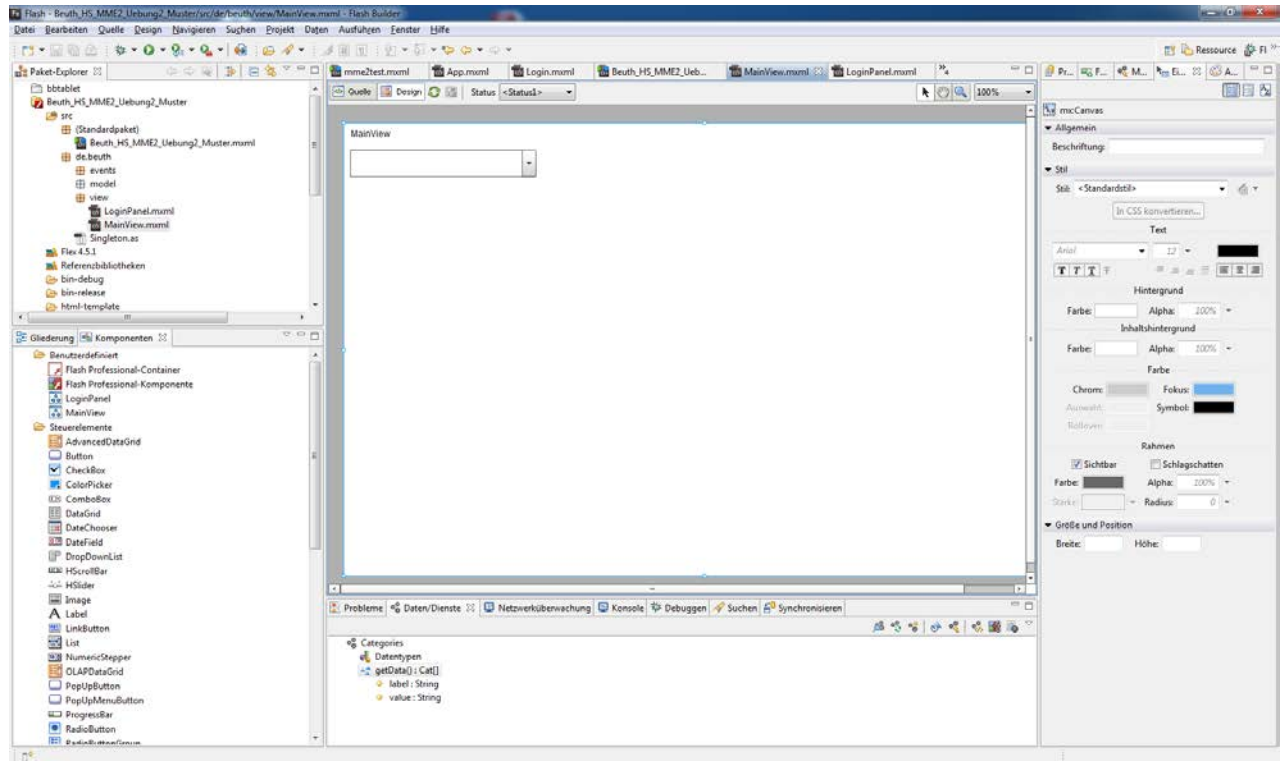
Dienstname:

Dienstpaket:

Datentyppaket:

Hinweis: Für in anderen Domänen gehostete Dienste ist eine [Cross-Domain-Datei](#) erforderlich.

Flash Builder 4 erstellt Ihnen nun automatisch in den angegebenen Paketen das ValueObject und den Service für den Datenaufruf.



Als nächstes werden Sie den Datenaufruf mit Ihrer ComboBox-Komponente verknüpfen. Hierfür müssen Sie nichts weiter tun, als die nun im "Daten/Dienste" angelegte Funktion "getData() : Cat[]" direkt auf die ComboBox-Komponente mit der Maus zu ziehen und das folgende Fenster mit "OK" zu bestätigen.

Wenn Sie nun die Anwendung starten, sehen Sie, dass die Kategorien nun erfolgreich geladen wurden.

