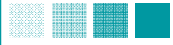


Flex States

Roman Bartoli



Gliederung

- Was sind States?
- So nicht!
- State Pattern
- States in MXML
- Modifikatoren
- States ableiten
- States gruppieren
- DesignView als Hilfe



Was sind States?

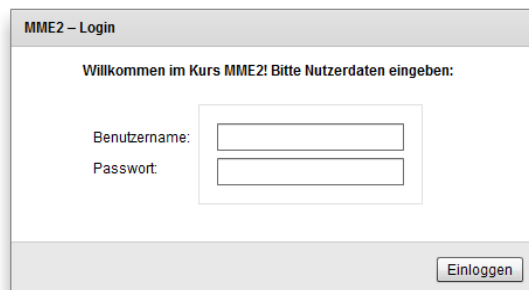
- States (meist View States) sind Zustände einer Komponente oder der ganzen Anwendung

z.B. Button Komponente:



Was sind States?

- Flex ist nicht seitenbasiert
- Einsatz von States als eleganter Weg zum Umschalten von Fenstern

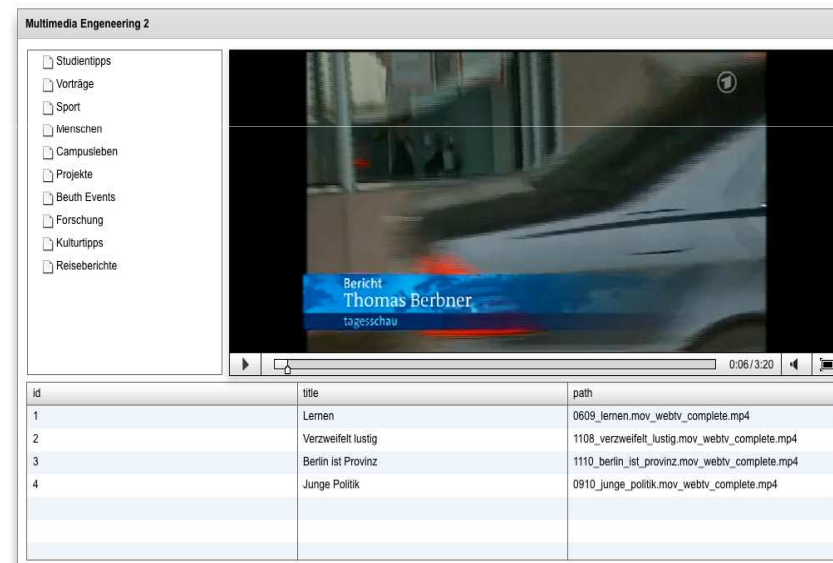


MME2 – Login

Willkommen im Kurs MME2! Bitte Nutzerdaten eingeben:

Benutzername:

Passwort:



Multimedia Engineering 2

Navigation:

- Studientipps
- Vorträge
- Sport
- Menschen
- Campusleben
- Projekte
- Beuth Events
- Forschung
- Kulturtipps
- Reiseberichte

Video Player:

Bericht
Thomas Berbner
tagesschau

0:06 / 3:20

id	title	path
1	Lernen	0609_lernen.mov_webtv_complete.mp4
2	Verzweifelt lustig	1108_verzweifelt_lustig.mov_webtv_complete.mp4
3	Berlin ist Provinz	1110_berlin_ist_provinz.mov_webtv_complete.mp4
4	Junge Politik	0910_junge_politik.mov_webtv_complete.mp4

So nicht!

```
public static const LOGIN_STATE = "login";
public static const APPLICATION_STATE = "application";

private var _state:String = LOGIN_STATE;

public function setState(state:String) {
    _state = state;
}

private function showElements():void {
    switch(_state) {
        case LOGIN_STATE:
            loginPanel.visible = true;
            break;
        case APPLICATION_STATE:
            loginPanel.visible = false;
            application.visible = true;
            break;
    }
}
```

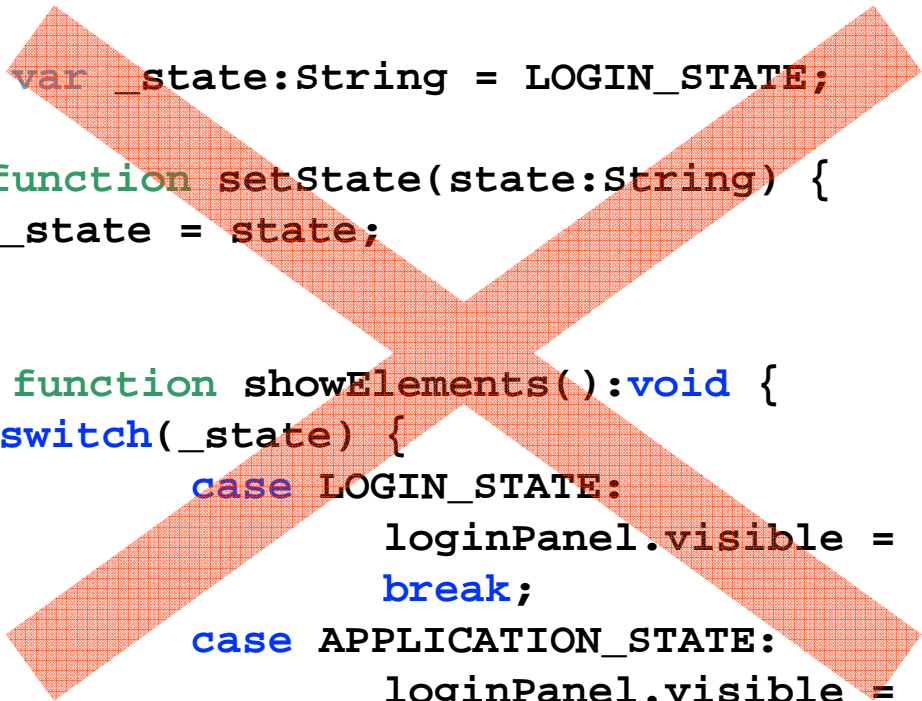
So nicht!

```
public static const LOGIN_STATE = "login";
public static const APPLICATION_STATE = "application";

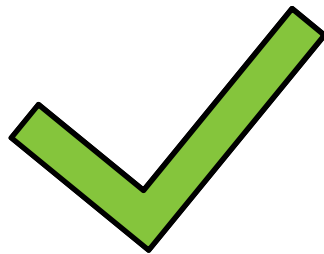
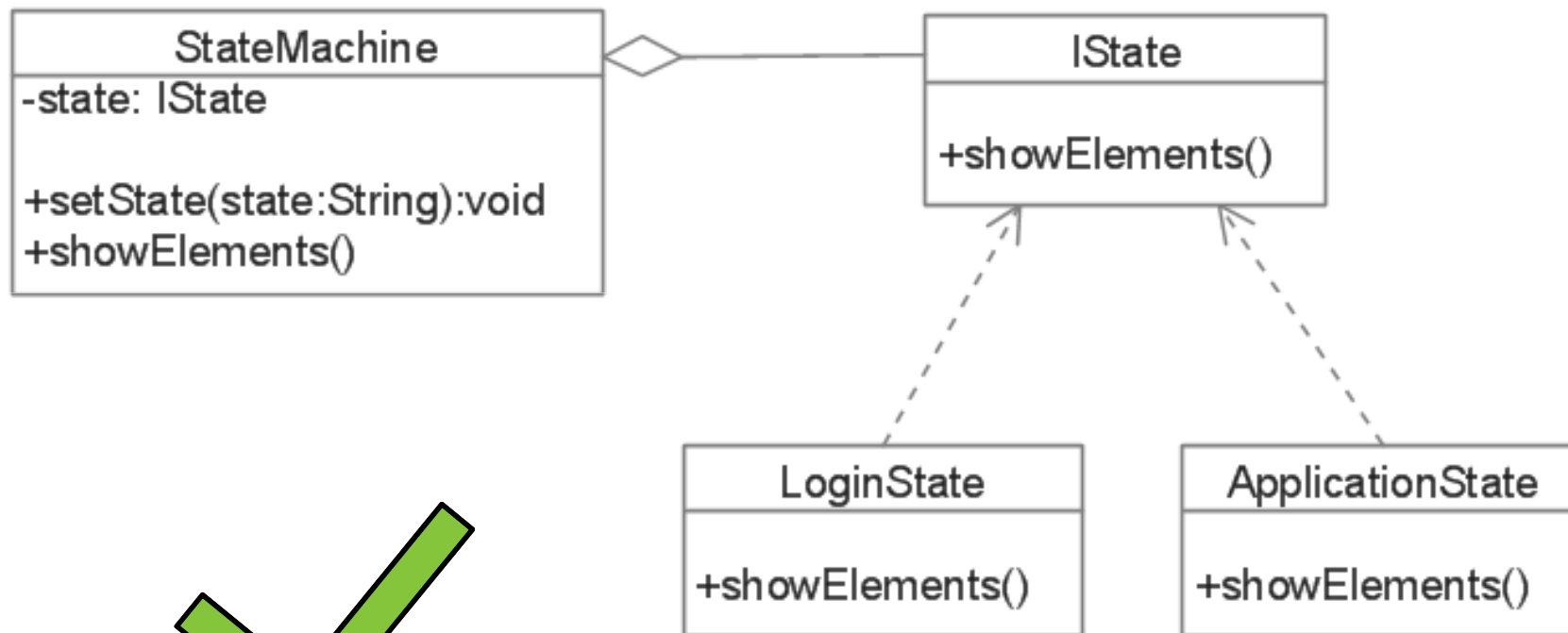
private var _state:String = LOGIN_STATE;

public function setState(state:String) {
    _state = state;
}

private function showElements():void {
    switch(_state) {
        case LOGIN_STATE:
            loginPanel.visible = true;
            break;
        case APPLICATION_STATE:
            loginPanel.visible = false;
            application.visible = true;
            break;
    }
}
```



State Pattern



States in MXML

```
<s:states>
    <s:State name="LoginState"/>
    <s:State name="ApplicationState"/>
</s:states>

<fx:Script>
    <![CDATA[
        ...
        // Aenderung über currentState
        private function setLoginState():void {
            this.currentState = "loginState";
        }
    ]]>
</fx:Script>
```

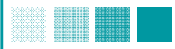



States in MXML

- State Änderungen über Punktnotationen in MXML
- State Änderungen können direkt in die Komponenten geschrieben werden
- Kann für alle Attribute außer *id* angewendet werden
- Der erste State ist immer der baseState, gleichbedeutend mit `currentState= ' '`

```
<s:Label text.firstState="Ich bin im ersten State"  
        text.secondState="Ich bin im zweiten State" />
```





Modifikatoren

- Styles verändern

```
<s:Label color.firstState="#FF0000" />
```

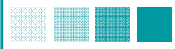
- Events setzen

```
<s:Button  
    click.firstState="onFirstStateClick()"  
    click.secondState="onSecondStateClick()" />
```

- Komponenten hinzufügen/entfernen

```
<s:Panel includeIn="firstState"  
        excludeFrom="secondState" />
```





States ableiten

- States können voneinander ableiten
- Im `SecondState` wird auch das erste Label angezeigt.

```
<s:states>  
    <s:State name="FirstState" />  
    <s:State name="SecondState" basedOn="FirstState" />  
    <s:State name="ThirdState" />  
</s:states>
```

```
<s:Label text.firstState="Ich bin im ersten State" />  
<s:Label text.secondState="Ich bin im zweiten State" />
```



States gruppieren

- Das Panel wird im SecondState **und** FourthState angezeigt.

```
<s:states>
  <s:State name="FirstState" stateGroups="group1"/>
  <s:State name="SecondState" stateGroups="group2"/>
  <s:State name="ThirdState" stateGroups="group1"/>
  <s:State name="FourthState" stateGroups="group1,group2"/>
</s:states>
```

```
<s:Panel includeIn="group2"/>
```



Design View als Hilfe

