

# **LIVE DEMO OF COMPUTED PROPERTIES LIBRARY FOR UI5**

Martin Käser @ eXXcellent solutions

<https://www.exxcellent.de>

# COMPUTED PROPERTIES??

- Used to store intermediate results
- Calculates a value from input properties
- Updates value when any input changes
- Behaves like any normal model property
- Concept from Vue.js and Aurelia
- Simple library for UI5

# EX. EDITOR (1): SHARED CALCULATION

Login:	<input type="text" value="johanna.hoffmann"/>
First name:	<input type="text" value="Johanna"/>
Family name:	<input type="text" value="Hoffmann"/>
Email:	<input type="text" value="Johanna.Hoffmann@e-corp.com"/>
<input type="button" value="Cancel"/>	

Login:	<input type="text" value="johanna.hoffmann"/>
First name:	<input type="text" value="Johanna"/>
Family name:	<input type="text" value="Hoffmann"/>
Email:	<input type="text" value="Johanna@e-corp.com"/>
<input type="button" value="Save"/>	
<input type="button" value="Reset"/>	
<p>Warning: You have unsaved changes.</p>	

- Show/hide controls if some properties change vs backup properties
- Multiple controls share parts of calculation: "changed" status

# EX. EDITOR (2): UI5 - OUT OF THE BOX

## 1) Expression Binding:

```
<Button visible="{= %{/login} !== %{{orig}}/login} || %{/surname} !== %{{orig}}/surname} ... }" ... />
```

## 2) Formatter:

```
<Button visible="{ parts: ['/login','orig>/login','/surname','orig>/surname', ...],  
  formatter: '.hasChanged' }" ... />
```

## Flaws:

- Verbose syntax
- Poor reusability
- Logic partially in view

# EX. EDITOR (3): LIB - COMPUTED PROPERTY IN VIEW

```
<Page>
  <dependents>
    <util:ComputedProperty
      var="{local>/changed}"
      value="{= %{/login} != %{orig>/login} || %{/surname} != %{orig>/surname} ... }"/>
    </dependents>
    ...
  <Button visible="{local>/changed}" ... />
  <Button visible="{= !%{local>/changed} }" ... />
</Page>
```

- Custom element "ComputedProperty" propagates "value" to "var"
- Takes advantage of (expression) binding mechanism
- Put it inside "dependents" aggregation
- Benefits: reusable

# EX. EDITOR (4): LIB - COMPUTED PROP BUILDER

```
new ComputedPropertyBuilder(this, "local>/changed")
  .withFunction(
    (login, loginOrig, surname, surnameOrig) => login!==loginOrig || surname!==surnameOrig,
    ["/login", "orig>/login", "/surname", "orig>/surname"])
  .add();
```

- Alternative: "ComputedPropertyBuilder" allows definition in controller
- Takes JavaScript (arrow) function plus parameters from model(s)
- Stores function in model and constructs expression binding:

```
{= %{$funcs>/isChanged_123}( %{/login},%{orig>/login},%{/surname},%{orig>/surname})
```

- Dynamically adds "ComputedProperty" element to view
- Benefits: reusable, readable & logic in controller

# EX. TABLE (1): ADVANCED USE CASES

Users 10 rows				
First name	Family name	Email	Pretty email	
Philipp	Werner	Philipp.Werner@e-corp.com	Philipp Werner <Philipp.Werner@e-corp.com>	Edit
Emily	Schmitt	Emily.Schmitt@e-corp.com	Emily Schmitt <Emily.Schmitt@e-corp.com>	Edit
Sofia	Becker	Sofia.Becker@e-corp.com	Sofia Becker <Sofia.Becker@e-corp.com>	Sofia.Becker@e-corp.... Save Cancel
Jonas	Walter	Jonas.Walter@e-corp.com	Jonas Walter <Jonas.Walter@e-corp.com>	Edit
Max	Wagner	Max.Wagner@e-corp.com	Max Wagner <Max.Wagner@e-corp.com>	Max.Wagner@e-corp.... Save Cancel
Emily	Herrmann	Emily.Herrmann@e-corp.com	Emily Herrmann <Emily.Herrmann@e-corp.com>	Edit
Anna	Meyer	Anna.Meyer@e-corp.com	Anna Meyer <Anna.Meyer@e-corp.com>	Anna.Meyer@e-corp.c... Save Cancel
Lilly	Hofmann	Lilly.Hofmann@e-corp.com	Lilly Hofmann <Lilly.Hofmann@e-corp.com>	Edit
Jakob	Müller	Jakob.Müller@e-corp.com	Jakob Müller <Jakob.Müller@e-corp.com>	Edit
Paul	Neumann	Paul.Neumann@e-corp.com	Paul Neumann <Paul.Neumann@e-corp.com>	Edit

- Computed Properties for lists

## EX. TABLE (2): WRAPPING LIST DATA

- Augment data in temporary model without polluting original model:

```
list: [  
  ...  
  {firstname: "Sofia", surname: "Becker", email: "BeckerS@e-corp.com"},  
  ...  
]
```



```
wrappedList: [  
  ...  
  {  
    obj: {firstname: "Sofia", surname: "Becker", email: "BeckerS@e-corp.com"},  
    temp: {editMode: true, prettyEmail: "Sofia Becker <BeckerS@e-corp.com>"}  
  },  
  ...  
]
```

- Updated only if list reference changes, so add elements with concat(), not push()



## EX. TABLE (3): COMPUTED PROP WITH REL. BINDING

- Bindings relative to row binding context
- ComputedProperty in row template is duplicated into every row

```
new ComputedPropertyBuilder(this, "temp/prettyEmail")
  .withFunction(
    (firstname, surname, email) => `${firstname} ${surname} <${email}>`)
    ["obj/firstname", "obj/surname", "obj/email"])
  .addById("rowTemplate");
```

# COMPUTED PROPERTY LIBRARY

- Library with demo on Github:  
<https://github.com/exxcellent/ui5-computed-properties>
- JavaScript ES5 for compatibility
- Property changes are logged on debug - filter for "computed-property"
- Just copy ComputedProperty.js and ComputedPropertyBuilder.js!

# SUMMARY

- Two layers: May be used in view or controller
- Avoids redundancy
- Provides reasonable abstraction
- Improves readability of view
- Calculation logic in controller
- Simplifies debugging

**THANK YOU FOR YOUR ATTENTION!**