Advanced Developers Conference
Development for Professionals!

Control Templates

Thomas Claudius Huber

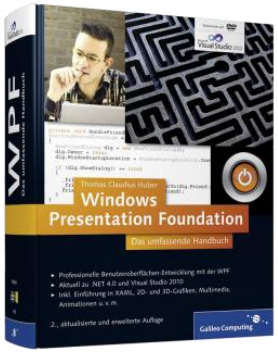
thomas.huber@trivadis.com

www.thomasclaudiushuber.com

ppedv

Thomas Claudius Huber

- .NET Senior Consultant @Trivadis Basel
- Spezialisiert auf Silverlight und WPF
  - Autor der umfassenden Handbücher zu Silverlight und WPF

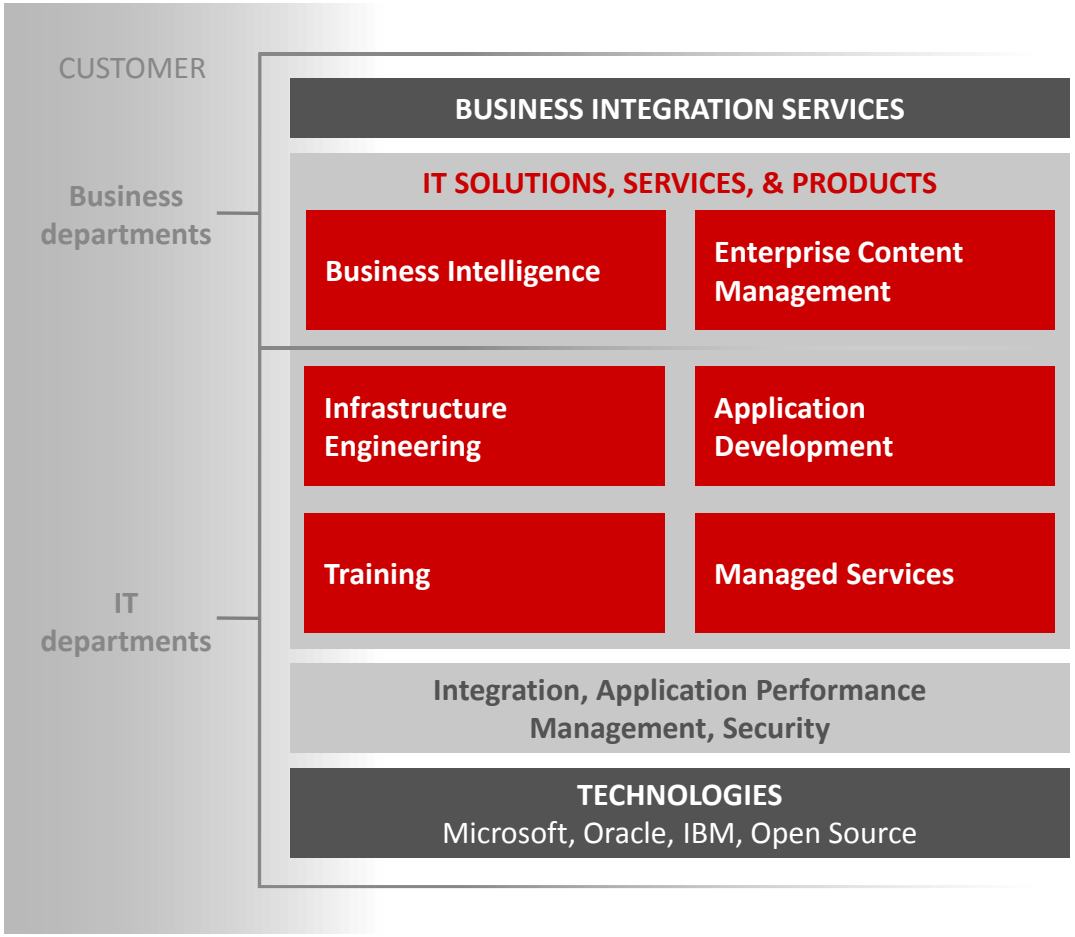


- Entwickelt im Business-Umfeld hauptsächlich WPF- und Silverlight-Anwendungen

Advanced Developers Conference
Development for Professionals!

Trivadis Solutions Portfolio and competences

trivadis
makes IT easier.



Advanced Developers Conference
Development for Professionals!

Trivadis facts & figures

- 11 Trivadis locations with more than 550 employees
- Financially independent and sustainably profitable
- Key figures 2010
  - Revenue CHF 101 / EUR 73 mio.
  - Services for more than 700 clients in over 1‘800 projects
  - Over 170 Service Level Agreements
  - More than 5'000 training participants
  - Research and development budget: CHF 5.0 / EUR 3.6 mio.

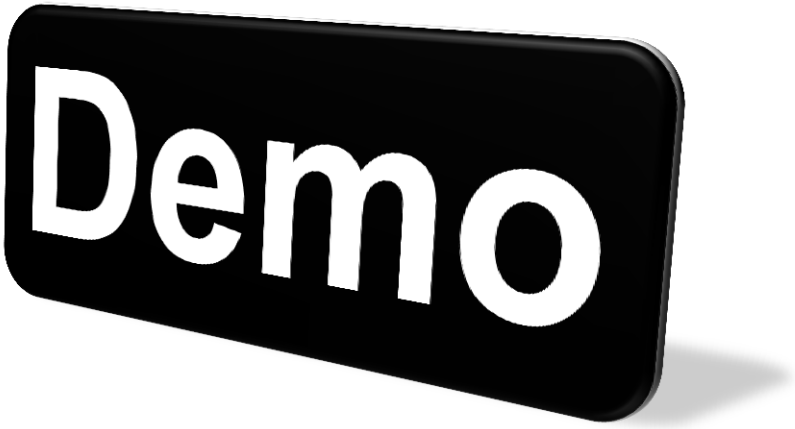
Agenda

- Das Aussehen anpassen
- Logik und Aussehen Verbinden
  - TemplateBinding
  - PART-Elemente
- Visuelles Feedback
  - Trigger
  - Visual States
- Eigene Custom Controls entwickeln

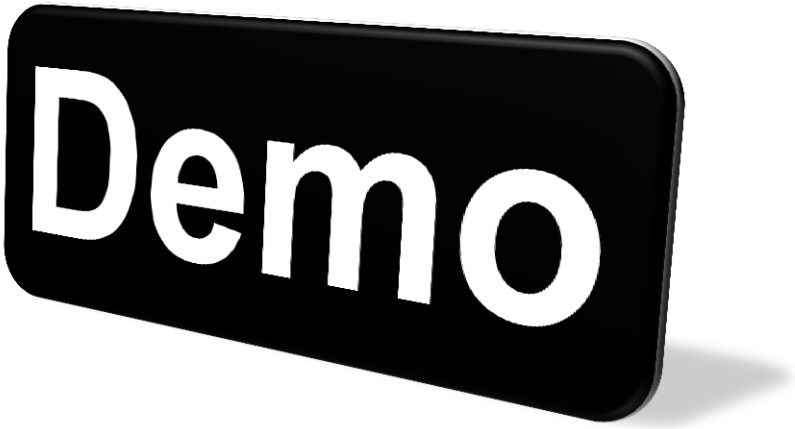
WPF/Silverlight-Controls sind „lookless“

- das Aussehen ist von der Logik getrennt
- Das Aussehen eines Controls wird über ein ControlTemplate (XAML) definiert
- Die Logik wird klassisch in C# oder VB.NET implementiert

Aussehen in Visual Studio XAML anpassen



Aussehen in Blend anpassen



Agenda

- Das Aussehen anpassen
- Logik und Aussehen Verbinden
  - TemplateBinding
  - PART-Elemente
- Visuelles Feedback
  - Trigger
  - Visual States
- Eigene Custom Controls entwickeln

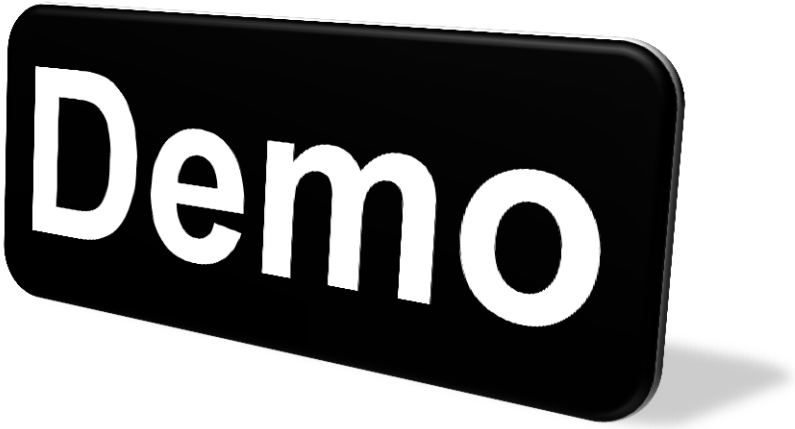
Logik und Aussehen verbinden

- Controls nutzen dazu zwei Möglichkeiten
  - TemplateBinding
  - PART-Elements

TemplateBinding

- Markup-Extension TemplateBinding
- Bindet eine Dependency Property aus einem Element des ControlTemplates an eine Property des Controls, auf welches das Template angewendet wird
- Falls keine Dependency Property vorliegt, wird ein normales Binding verwendet
  - Mit {Binding RelativeSource={RelativeSource TemplatedParent},Path=…}

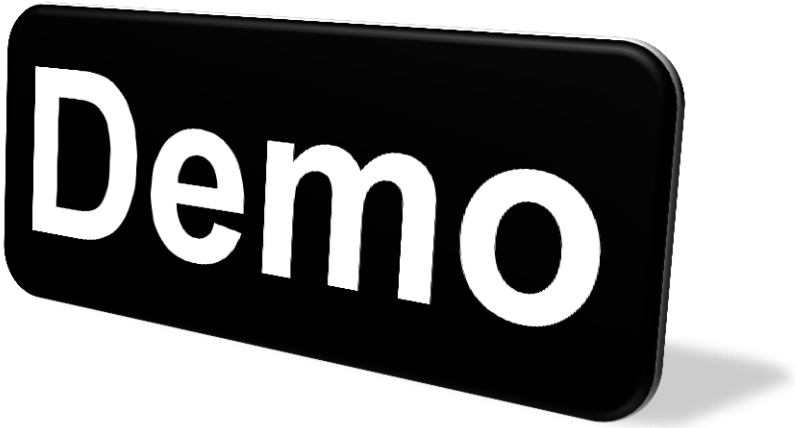
TemplateBinding



PART-Elemente

- Einige Controls verlangen im ControlTemplate bestimmte Elemente mit bestimmten Namen
- Die erwarteten Elemente gibt das entsprechende Control mit TemplatePart-Attributen auf Klassenebene an

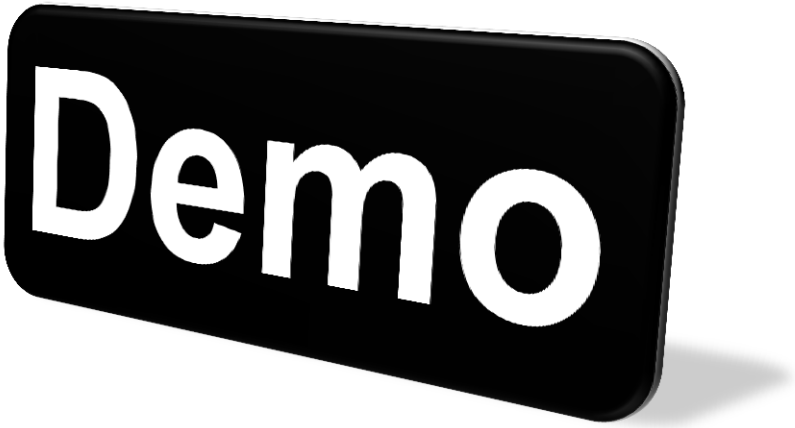
PART-Elemente



Expression Blend

- In Visual Studio werden ControlTemplates von Hand in XAML geschrieben
- Blend unterstützt im Gegensatz zu Visual Studio das erstellen von ControlTemplates im Designer
  - Blend unterstützt dabei das TemplateBinding und PART-Elemente

TemplateBinding und PART-Element in Expression Blend



Agenda

- Das Aussehen anpassen
- Logik und Aussehen Verbinden
  - TemplateBinding
  - PART-Elemente
- Visuelles Feedback
  - Trigger
  - Visual States
- Eigene Custom Controls entwickeln

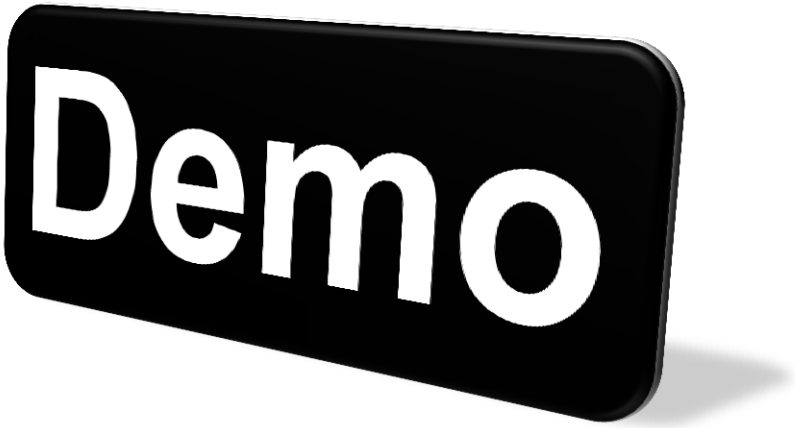
Visuelles Feedback

- Visuelles Feedback, wie MouseOver-Effekte, werden mit Triggern oder Visual States eingebunden
- Trigger werden nur von WPF, nicht jedoch von Silverlight unterstützt
- Silverlight verwendet für das visuelle Feedback sogenannte Visual States. Diese sind auch in WPF 4.0 möglich
  - In WPF 4.0 nutzen die Default-Templates der Controls allerdings Trigger.

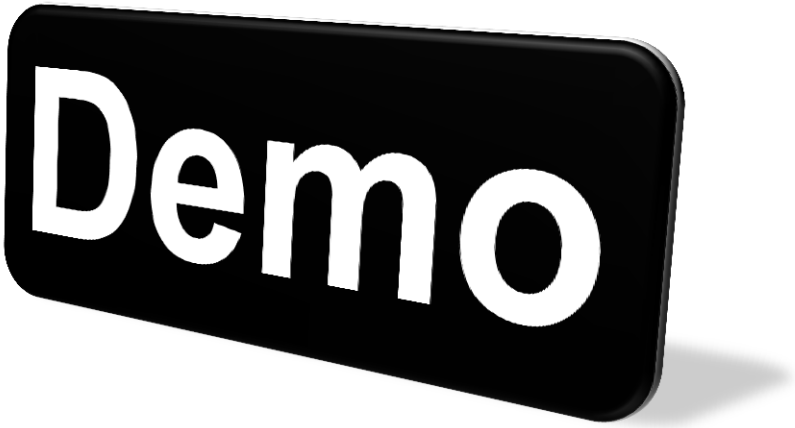
Trigger

- Trigger lassen sich sehr einfach in XAML schreiben
- Die ControlTemplate-Klasse besitzt eine Triggers-Property, zu welcher die Trigger hinzugefügt werden
- Ein Trigger kann
  - eine Property eines Elements im Template setzen
  - eine Animation auslösen (welche eine Property eines Elements im Template animiert)

Visuelle Zustände via Trigger



Visuelle Zustände via Trigger in Expression Blend



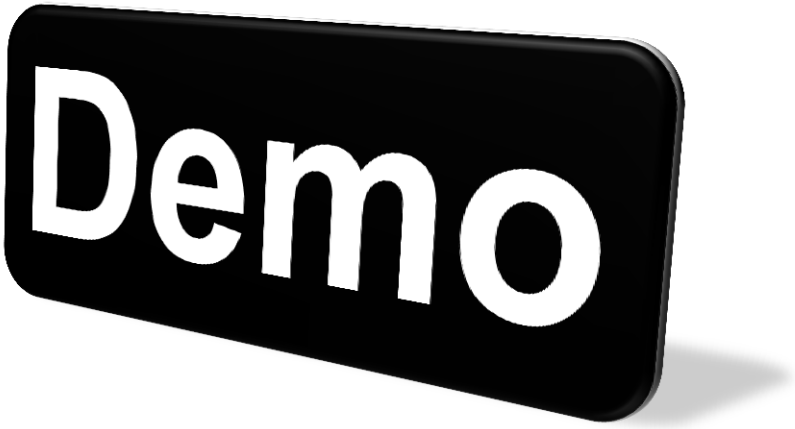
Visuelle Zustände via Visual State Manager

- Der Visual State Manager hat seinen Ursprung in Silverlight
  - wurde in .NET 4.0 auch in die WPF übernommen
- Ist gegenüber Triggern Designer-freundlicher
  - Anstatt mit Events und Properties wird mit tatsächlichen Zuständen gearbeitet
- Lässt sich aufgrund der Menge an XAML-Code nicht leicht „von Hand“ schreiben

Visuelle Zustände via Visual State Manager

- Ein Visual State beschreibt einen visuellen Zustand eines Controls
- Visual States lassen sich gruppieren
  - Aus jeder Gruppe ist genau ein Visual State aktiv
- Visual States werden auf Klassenebene mit dem TemplateVisualState-Attribute deklariert

Visual State Manager



Agenda

- Das Aussehen anpassen
- Logik und Aussehen Verbinden
  - TemplateBinding
  - PART-Elemente
- Visuelles Feedback
  - Trigger
  - Visual States
- Eigene Custom Controls entwickeln

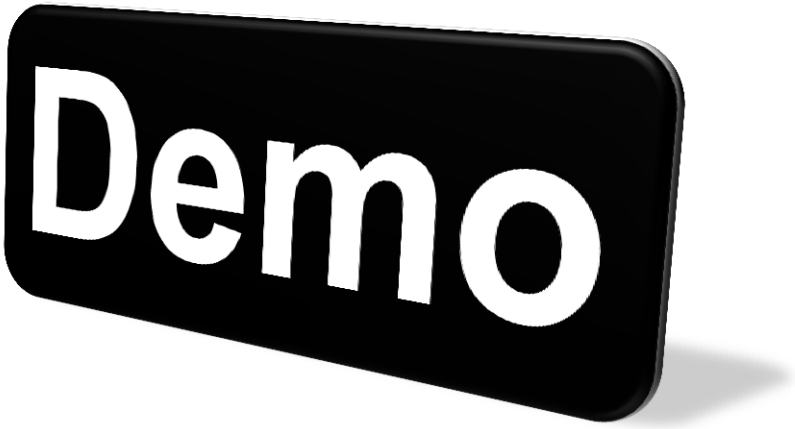
UserControl vs. Custom Control

- **User Control** unterstützt keine Themes und Templates, aber:
  - Ist einfach zu entwickeln. Besteht wie ein Window aus einer XAML- und einer Codebehind-Datei
  - Zum Gruppieren existierender Controls
- **Custom Control** unterstützt
  - ControlTemplates, um Logik vom UI zu trennen
  - Theme-spezifisches UI (nur bei WPF, nicht bei Silverlight)
    - Wird automatisch angewendet, wenn das Windows Theme sich ändert
- Custom Control wird meist dann erstellt, wenn das Control nicht nur von einer Anwendung benötigt wird

Struktur eines Custom Controls

- *Themes\generic.xaml* enthält das UI
  - Definiert in XAML
  - In einem Style wird die Template-Property gesetzt
  - Die TargetType-Property enthält den Wert typeof(YourCustomControl).
- *YourCustomControl.cs* enthält eine Subklasse von Control
  - Der Default-Wert für die DefaultStyleKey-Property wird im statischen Konstruktor definiert, um das Control mit dem Style in generic.xaml zu verbinden.

Struktur eines Custom Controls



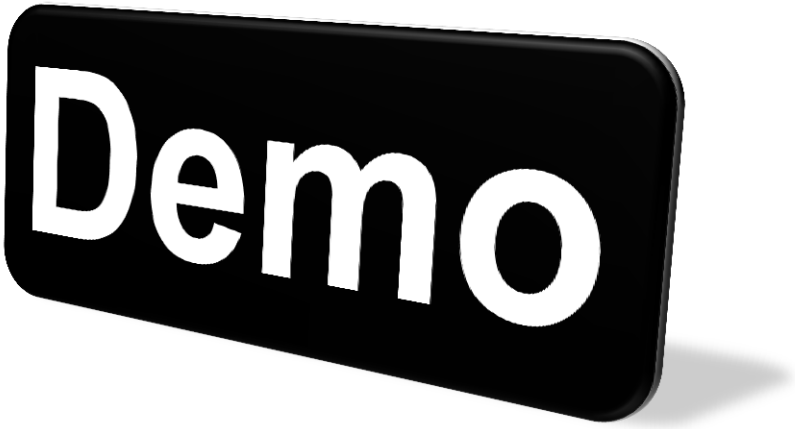
UI mit dem Code verbinden

- Template Binding
- PART-Elemente

PART-Elemente

- Elemente im Template mit dem *x:Name*-Attribut benennen
- OnApplyTemplate-Methode überschreiben
  - Darin mit this.Template.FindName(...) die benannten Elemente auslesen und in Klassenvariablen speichern.
- Mit TemplatePartAttributen auf der Klasse über die erwarteten Elemente «informieren»
  - Für Entwickler, die das Template neu definieren
  - Für Tools wie Expression Blend

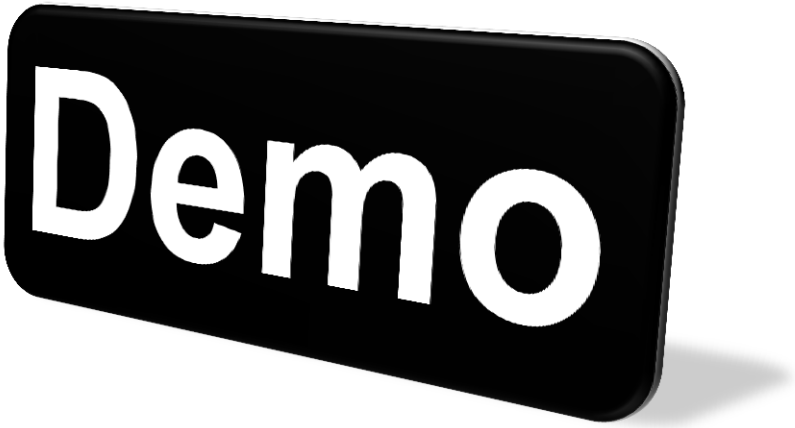
TemplateBinding & Part-Elemente



Visuelle Zustände unterstützen

- Zentrale Methode für einen Statusübergang erstellen
  - Darin VisualStateManager.GoToState aufrufen
- Klasse mit TemplateVisualState-Attributen ausstatten
  - Zur Information für Tools wie Expression Blend
- Anfangsstatus bereits in der OnApplyTemplate-Methode setzen

Visuelle Zustände unterstützen



Agenda

- Das Aussehen anpassen
- Logik und Aussehen Verbinden
  - TemplateBinding
  - PART-Elemente
- Visuelles Feedback
  - Trigger
  - Visual States
- Eigene Custom Controls entwickeln

Fazit

- ControlTemplates werden über PART-Elemente, TemplateBinding und VisualStates verbunden
- Trigger werden nur von der WPF unterstützt
- Lässt sich das Control in Expression Blend designen, hat der Entwickler des Controls die Attribute richtig gesetzt

Infos

- www.thomasclaudiushuber.com
- www.trivadis.com

- thomas.huber@trivadis.com
- www.twitter.com/thomasclaudiush

FRAGEN?

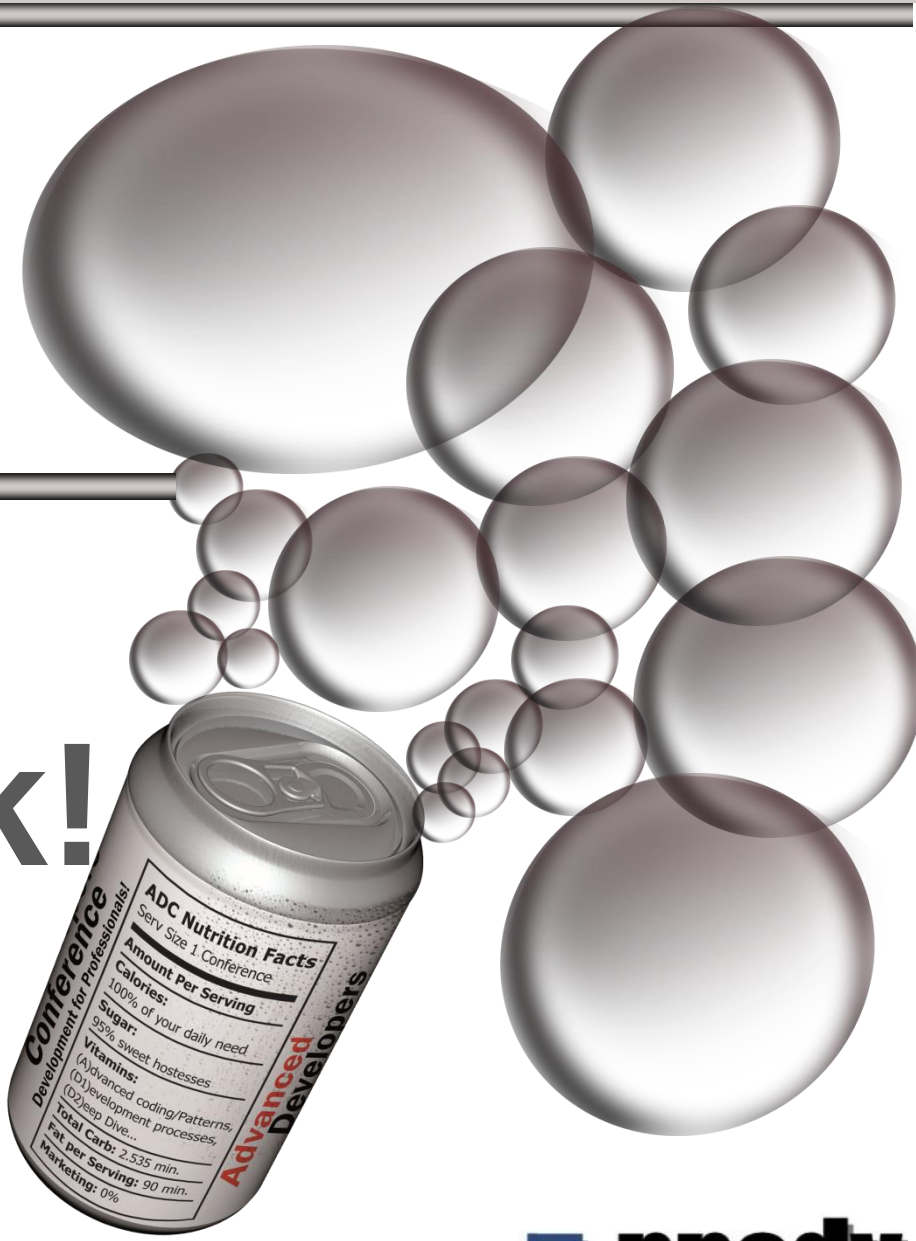
Advanced Developers Conference
Development for Professionals!

Ihr Feedback ist uns wichtig

Advanced Developers Conference
Development for Professionals!

Advanced Developers Conference
Development for Professionals!

Vielen Dank!



ppedv