***Dashboard & Charts***

*Kompakthandbuch*

ckg-logo.png

*Matthias Jenzen, CKG Entwicklung*

Inhalt

1 Einführung 3

1.1 Ziel 3

1.2 Basis Funktionen 4

1.3 Erweiterte Funktionen 4

2 Dashboard Homepage User Interface 5

2.1 Homepage, Grundaufbau 5

2.2 Einblenden versteckter Charts über Dropdown Menü 5

2.3 Drag & Drop - Anordnen der sichtbaren Charts 6

2.4 Ausblenden sichtbarer Charts 6

2.5 Report eines Charts aufrufen 7

3 Beispiele für Kunden individuelle Dashboard Homepages 8

3.1 Kunde „Lueg“ 8

3.2 Kunde „WKDA“ 9

3.3 Kunde „Cardocu“ 10

4 Chart Customizing 11

4.1 Verknüpfung zwischen Chart und MVC Report 11

4.2 JSon-Templates 12

4.3 Chart Customizing Javascript Funktionen 13

5 Chart Programmierung 14

5.1 Vorbereitung der ViewModels 14

5.2 Programmierung der Chart Funktionen 14

6 Dashboard Administration 15

6.1 Kunden bezogene Dashboard Aktivierung 15

6.2 User bezogene Chart Aktivierung 15

7 Praktischer Einsatz 16

7.1 Kroschke „On“ Portal 16

# Einführung

## Ziel

Ziel ist es, für eine personalisierbare und kundenbezogene Dashboard Homepage den architektonischen .NET Unterbau zu entwickeln, mit dem Zweck der schnellen Integration neuer, sowie bestehender Web-Reports als Chart-Komponenten bzw. „Widgets“ .

Verschiedene Chart-Typen, wie z. B. Balken- und Kreisdiagramme sollen dabei ebenso unterstützt werden, wie die benutzerfreundliche Anordnungsmöglichkeit der einzelnen Dashboard Komponenten (Reihenfolge ändern, sowie das Ein- und Ausblenden einzelner Charts, ggfls. eine Kommentarfunktion für den User, etc).

Dabei wird jeder Chart (z. B. über eine SQL Konfigurationstabelle) mit einer vorhandenen Web-Report Anwendung verknüpft, sowie entsprechend parametrisiert   
(Art des Charts, Titel, Kurzbeschreibung, etc).

Durch diese Verbindung zu einer bestehenden Web-Applikation ist bereits die Berechtigungsebene geregelt: Die User-Berechtigungen (welcher User darf welche Charts sehen?) werden voll-automatisch von den Applikations-Berechtigungen geerbt.   
Somit sind webseitig keine weitere Administration und kein zusätzliches Customizing notwendig.

*Der .NET Unterbau ist bereits fertig gestellt und stellt ab August 2015 seine Funktionalität bereits für eine kundenbezogene Anwendung (Kunde VWL, ITA 8120 „Durchlaufzeiten“) zur Verfügung.*

Die Herausforderung dieses Jahreszieles ist die eigentliche Bereitstellung einer für „Chart API-fremde“ Entwickler simplen + aufwandarmen Programmier-Anbindung.

Es soll damit für jeden Web-Entwickler ohne Einarbeitungsaufwand möglich sein, einen neuen Chart mit nur 1 Stunde Aufwand zu entwickeln!

Aufgrund dieser Software-architektonischen Herausforderung sollte dieses Jahresziel mit mindestens 60% gewichtet werden.

Dieses Jahresziel ist eher „programmier-lastig“.

Es bedarf aufgrund seines Anspruchs der Bereitstellung einer einfachen Programmieranbindung nur einer kompakten Dokumentation.

## Basis Funktionen

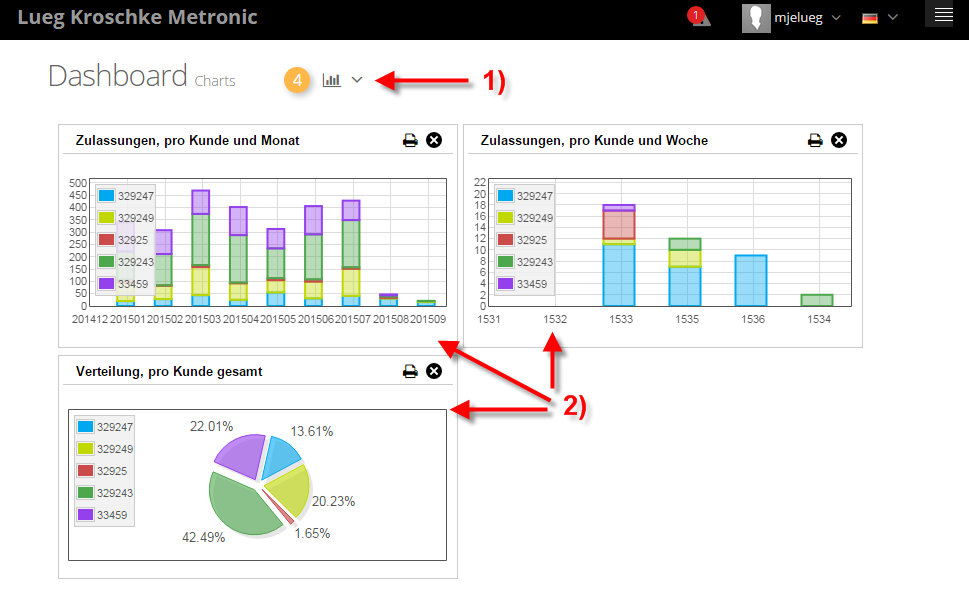
* Integration des Javascript Chart Frameworks “Flotr2” (<http://www.humblesoftware.com/flotr2/index> )
* Integration des Javascript Dashboard Frameworks  (<http://modeln.github.io/sDashboard>)
* Architektonischer .NET Unterbau mit Bereitstellung einer simplen, aufwandarmen Programmier-Anbindung für „API-fremde“ Entwickler   
  => mit dem Zweck der schnellen Integration neuer + bestehender Reports als Dashboard Chart-Komponente.  
  Es wird damit möglich sein, einen neuen Chart mit nur 1 h Aufwand zu entwickeln (dabei aber die bestehende Deployment Logistik berücksichtigend)
* Automatische Koppelung der Dashboard Charts an eine bestehende Report-Anwendung aus dem ServicesMvc Pool
* Report-Mehrfachkoppelung, d. h. ein Report kann mehrere Charts speisen  
  Beispiel: Ein Zulassungsreport kann einerseits eine Chart Komponente speisen „*Zulassungen der letzten 3 Monate / Kunde als Bar Chart*“ sowie eine zweite Komponente „*Prozentuale Verteilung der Zulassungen des letzten Jahres als Pie Chart*“
* Statisch konfigurierbarer Filter der Report-Daten als Datenquelle für den jeweiligen Dashboard Chart
* Lokalisierbarkeit, soweit es die Report-Daten zulassen.
* Vollautomatische Integration in die Berechtigungs-Struktur der zugehörigen Anwendung  („*Wer darf welchen Chart überhaupt sehen?*“)  
  => d. h. für jede Dashboard-Komponente benötigen wir für den Anfang *kein* eigenes Berechtigungskonzept, da wir die Berechtigungen aus der Report Anwendung „anzapfen“.
* Ein- und Ausblenden von einzelnen Dashboard Komponenten, persistierbar, User bezogen.
* Verschieben von einzelnen Dashboard Komponenten, persistierbar, User bezogen.

## Erweiterte Funktionen

Automatische Verlinkung von der Dashboard Chart-Komponente aus zum jeweiligen Report auf Basis des dem Chart zugrunden liegenden Report-Filters.  
Wir empfehlen eingehend, dieses Feature von Anfang mit einzuplanen, da Fragen der Integrität der Dashboard-Chart Daten so leicht beantwortet werden können.  
Z. B. Fragen wie   
*„Warum ist der Balken für die Zulassungen im Dezember so niedrig?? Wie setzen sich denn die Chart-Zahlen zusammen??“*  
=> können dann sehr leicht mit einem Button „*Zugrunde liegenden Report dieses Charts aufrufen*“ beantwortet werden.  
Direkt vom Chart aus würde dann der Report aufgerufen werden und dessen Pre-Selektionsfilter gleich mit analog zum Chart Filter voreingestellt werden.  
Beispiel:   
Ein Chart zeigt die Zulassungen jeweils der letzten 3 Monate an.  
Beim Aufruf des Reports würde der Report Pre-Selektionsfilter gleich auf diese 3 Monate synchron voreingestellt.

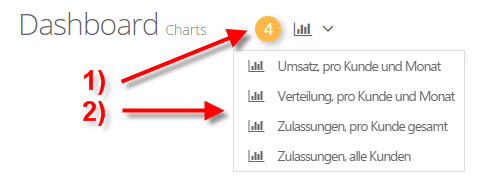
# Dashboard Homepage User Interface

## Homepage, Grundaufbau



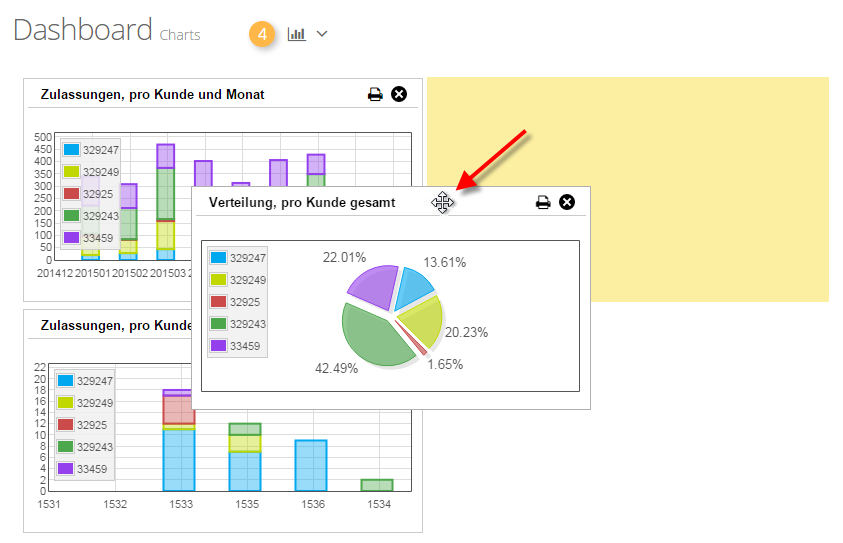
1. Versteckte Charts, per Dropdown Menü aufklappbar (User bezogen konfigurierbar)
2. Sichtbare Charts (User bezogen konfigurierbar)

## Einblenden versteckter Charts über Dropdown Menü



1. Anzeige *Anzahl* der versteckten Chart Komponenten.
2. Per Dropdown Menü werden die versteckten Charts aufgelistet.  
   Mit Klick auf einen Chart im Menü wird dieser Chart eingeblendet.  
     
   Die neue Chart Anordnung wird automatisch User bezogen persistiert, sodass diese beim nächsten Seitenaufruf erhalten bleibt.

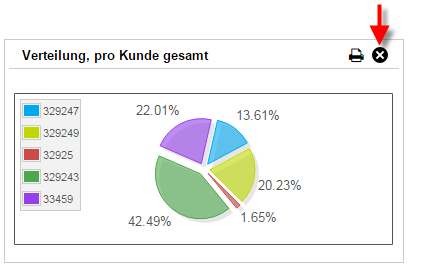
## Drag & Drop - Anordnen der sichtbaren Charts



Die sichtbaren Charts können User bezogen auf dem Dashboard angeordnet bzw. verschoben werden.  
Die geschieht ganz einfach per Maus über „Drag & Drop“.

Die neue Chart Anordnung wird automatisch User bezogen persistiert, sodass diese beim nächsten Seitenaufruf erhalten bleibt.

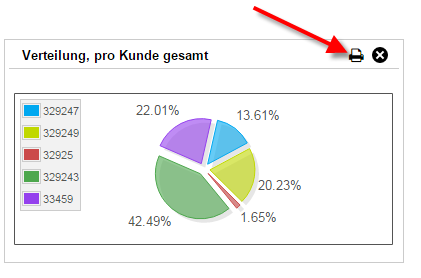
## Ausblenden sichtbarer Charts



Ein sichtbarer Chart wird über den „Close“ Button in der Chart Titelleiste versteckt.

Die neue Chart Anordnung wird automatisch User bezogen persistiert, sodass diese beim nächsten Seitenaufruf erhalten bleibt.

## Report eines Charts aufrufen

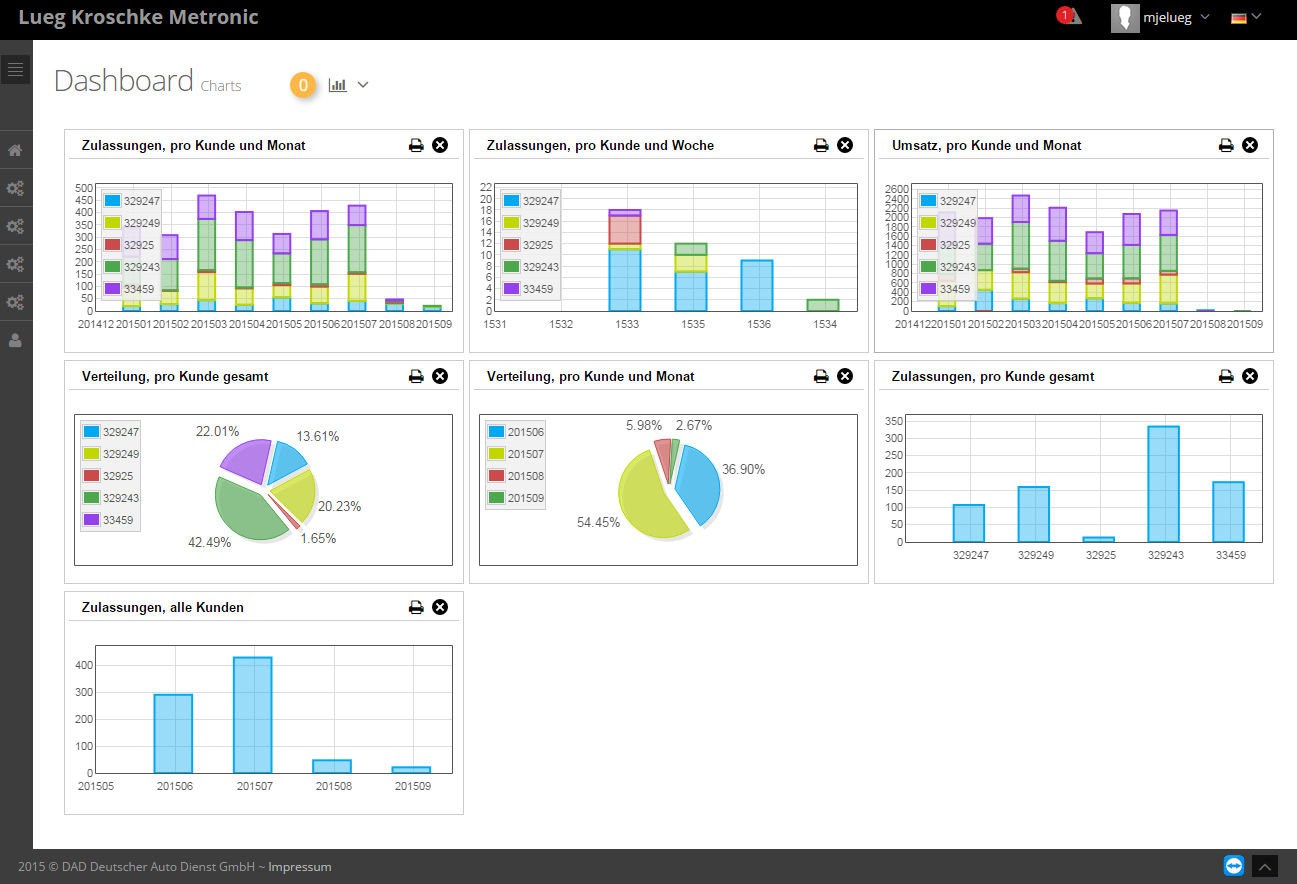


Über den „Print“ Button wird die dem Chart zugrunde liegende MVC Report Anwendung aufgerufen.

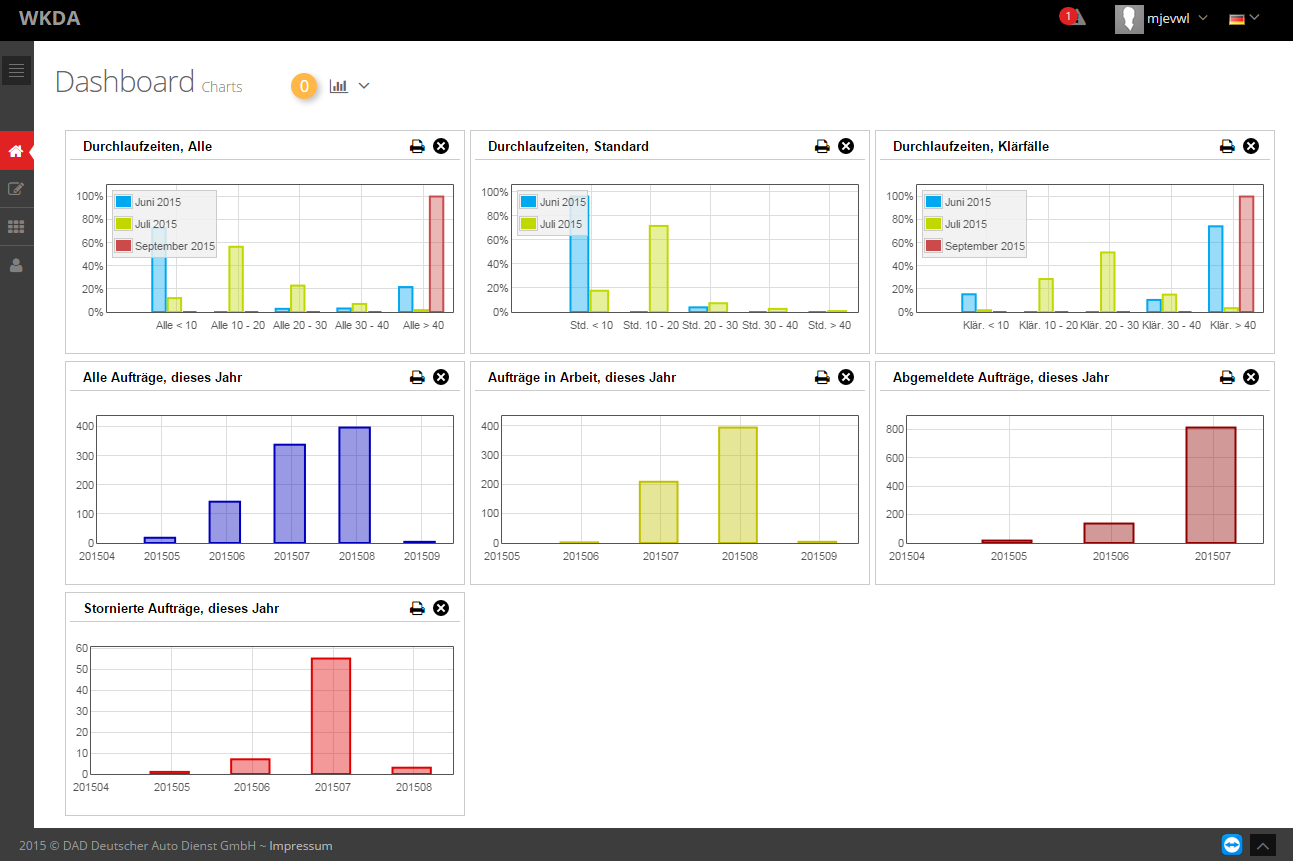
Die Verknüpfung eines Charts mit einer Report Anwendung wird weiter unten in diesem Handbuch erläutert.

# Beispiele für Kunden individuelle Dashboard Homepages

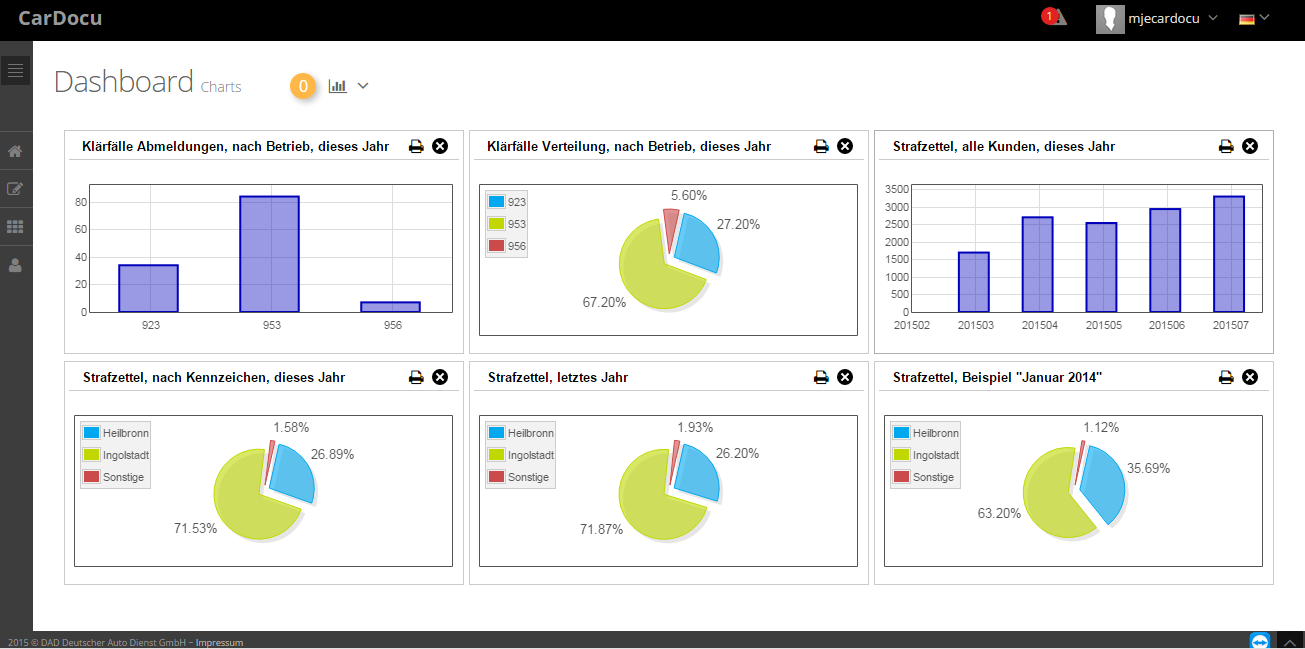
## Kunde „Lueg“



## Kunde „WKDA“

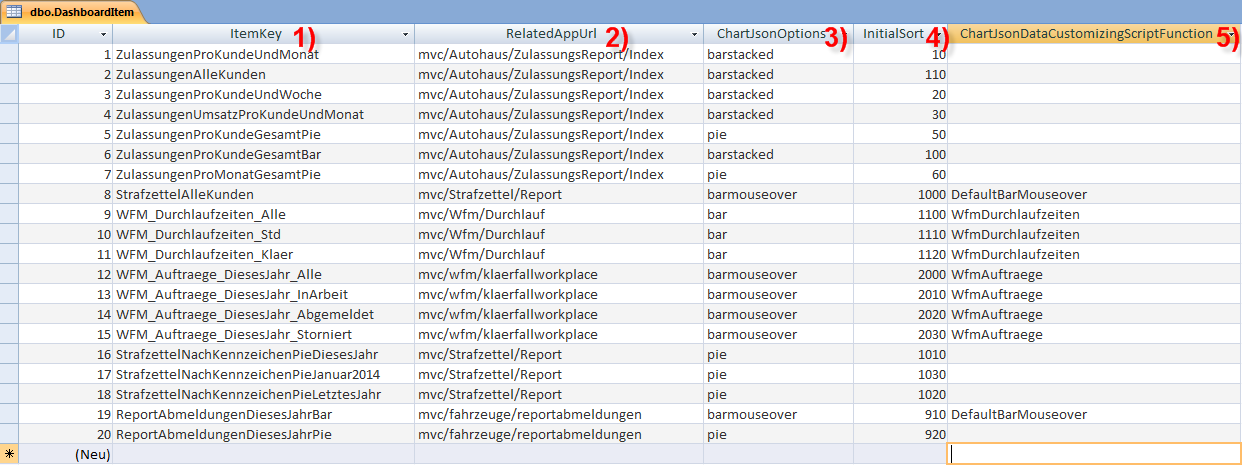


## Kunde „Cardocu“



# Chart Customizing

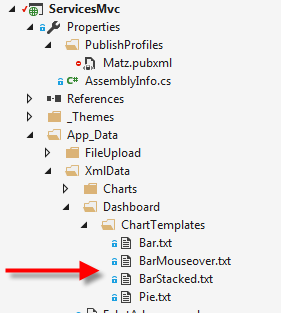
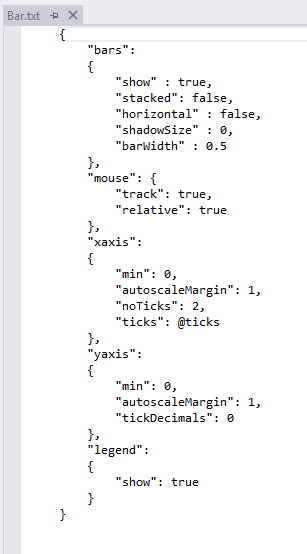
## Verknüpfung zwischen Chart und MVC Report



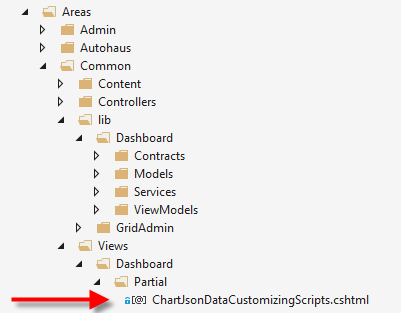
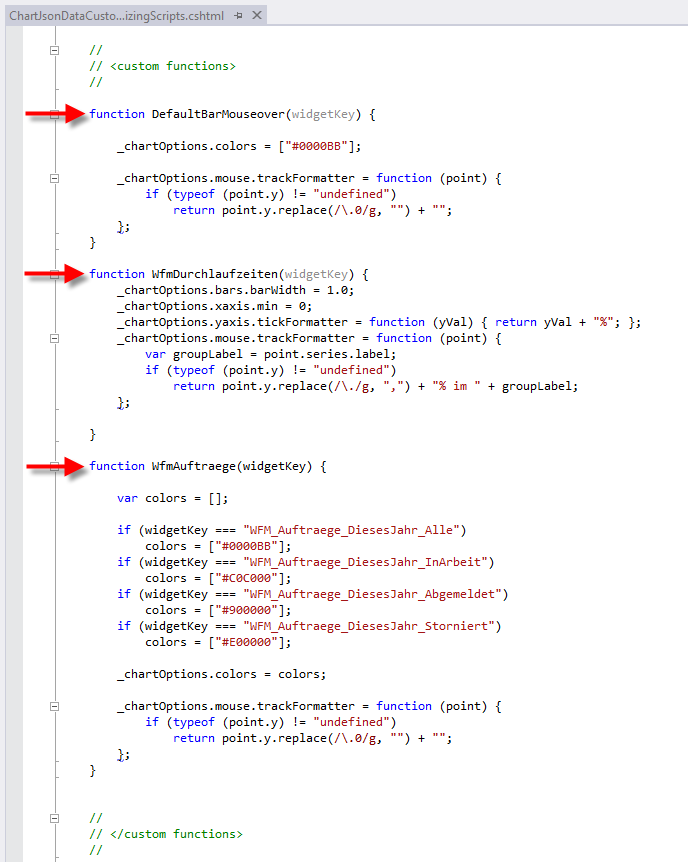
Über die SQL Tabelle „DashboardItem“ wird ein Chart deklariert. Erklärung der Tabellenspalten:

1. **ItemKey**  
   Frei definierbarer Text-Key. Dieser Key markiert auch in der Programmierung die zugehörige C# Chart Funktion. (Hierauf wird noch weiter unten in diesem Handbuch eingegangen)
2. **RelatedAppUrl**  
   Die Application Url des dem Chart zugrunde liegenden MVC Reports.  
   Mit den Spalten 1) und 2) wird die eigentliche Verbindung zwischen Chart und Report hergestellt.  
   Über die Spalte RelatedAppUrl ist gleichzeitig die Berechtigungs-Ebene geklärt.  
   (Welcher User bekommt welche Charts bereitgestellt?)
3. **ChartJSonOptions**Ein Verweis auf eine im MVC Projekt gespeicherte JSon-Template, die Typ + Erscheinungsbild des Charts definiert. (Z. B. „Pie“ oder „Bar“ Chart)
4. **InitialSort**Die initiale Anordnung der Chart Komponenten auf der Dashboard Seite.  
   Wenn der User nach dem erstmaligen Aufruf die Chart Komponenten nach eigener Vorstellung umsortiert, hat seine individuelle Sortierung Vorrang.
5. **ChartJsonDataCustomizingScriptFunction**Verweis auf eine im MVC Projekt gespeicherte Javascript Funktion, die ein individuelles Formatieren des Charts ermöglicht.   
   Eine Javascript kann in anderen Charts wiederverwendet werden.  
   Über diese Javascript Funktion kann auf alle Nuancen der Chart-Darstellung eingewirkt werden.  
   (Z. B. Legende ein- oder ausblenden bzw. individuell positionieren, Koordinaten Achsen definieren, Beschriftung ein- und ausblenden, Labeltexte frei definieren, etc)

## JSon-Templates

Die in der Spalte „ChartJSonOptions“ definierbaren JSon-Templates werden wie folgt im MVC Projekt abgelegt:  
  
  
  
Beispiel für eine JSon-Template:  
  


## Chart Customizing Javascript Funktionen

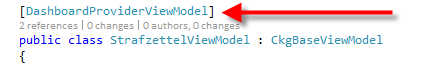
Die in der Spalte „ChartJsonDataCustomizingScriptFunction“ definierbaren Javascript Funktionen werden wie folgt im MVC Projekt abgelegt:  
  
  
  


# Chart Programmierung

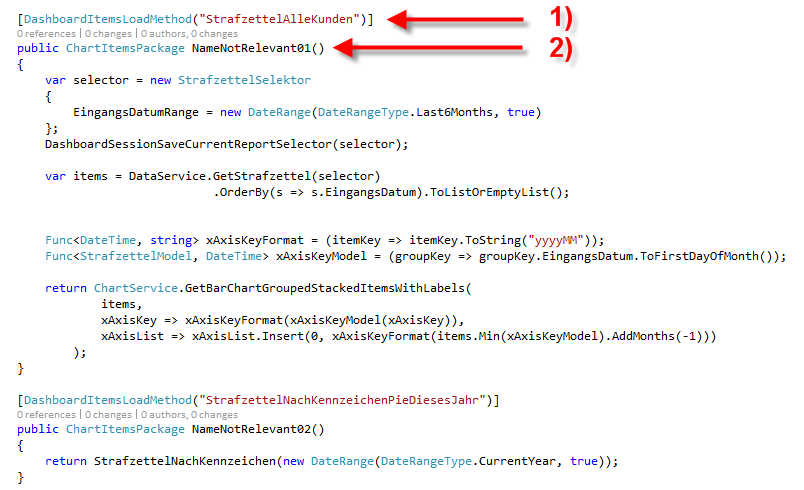
In diesem Kapitel wird die grundlegende Vorgehensweise der Chart Programmierung beschrieben.   
Es wird dabei nur rudimentär auf Programmierdetails eingegangen.

## Vorbereitung der ViewModels

Zur Identifizierung der ViewModels die Dashboard + Chart Funktionen bereitstellen sucht das .NET Dashboard System alle ViewModel Klassen mit bestimmten „Markern“.  
Diese Marker werden in der .NET Architektur auch „Attribute“ genannt.

Eine ViewModel Klasse die vom Dashboard System korrekt erkannt werden soll,   
muss wie folgt attribuiert werden:  
  


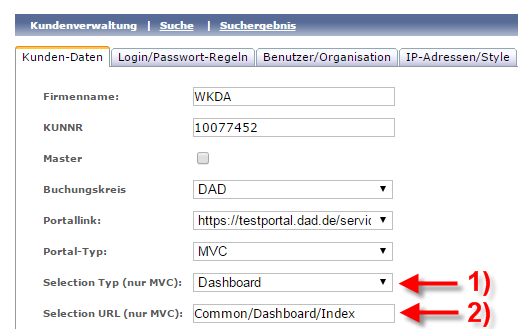
## Programmierung der Chart Funktionen



1. Über das Attribut [DashboardItemsLoadMethod] wird eine C# Funktion dem Dashboard System als Chart Funktion bekannt gemacht.  
   Der Attribut-Parameter enthält den Chart Key (z. B. „StrafzettelAllKunden“), der auf den entsprechenden Tabelleneintrag in der Customizing Tabelle verweist.
2. Der Chart Funktionsname an sich ist nicht relevant, da die Funktion vom Dashboard System über .NET Reflection aufgerufen wird.  
   Einzig der Rückgabewert muss vom Typ „ChartItemsPackage“ sein.  
   Dies ist eine Datenstruktur, die sämtliche aufbereitete Chart-Daten enthält.

# Dashboard Administration

## Kunden bezogene Dashboard Aktivierung



1. Die Aktivierung des Dashboards für einen Kunden erfolgt im Web über die Kunden Administration.  
   Als Homepage (Selection Typ) wird die Variante „Dashboard“ eingestellt.
2. Die Homepage Url ist fest vorgegeben und kann bei Variante „Dashboard“ nicht editiert werden.

## User bezogene Chart Aktivierung

Die User bezogene Chart Aktivierung erfolgt voll automatisch, weil die dem User auf seinem Dashboard zur Verfügung gestellten Charts von den Anwendungsberechtigungen des Users gesteuert werden.  
Charts mit zugrunde liegenden Applikationen, für die der User keine Berechtigung hat, werden dem User im Dashboard auch gar nicht zur Auswahl angeboten.

# Praktischer Einsatz

## Kroschke „On“ Portal

Das erste Kunden Feedback zum neuen Dashboard Feature war bereits positiv, sodass das DAD Projektmanagement dieses Feature dem CKG Beirat am Beispiel des Autohaus Kunden „Lueg“ präsentieren konnte.

