mp technology

vaadin

Vaadin 8 – What's new

26.09.2017 Jakob Magun, Benjamin Schupp

Vaadin Release 8

- Was bringt die Version 8 von Vaadin mit sich?
- Was bedeutet dies für meine Projekte?

Recap: Neu in Vaadin 8

Vaadin 8 kurz und knapp

- Ziel: Vereinfachung und mehr Flexibilität
  - #FFS – fight for simplicity

- Mittels:
  - Baut auf Java 8 auf
    - Lambdas und funktionale Programmierung
  - Datenmodell und Databinding vollständig umgebaut
    - API Änderung des Datenmodells
    - Typisierter Zugriff auf Pojos
    - Neuer Backend Zugriff
  - Anpassung von Defaults – weniger Boilerplate

Vaadin 8 kurz und knapp

- Grid
  - Seit Version 8.1 vollständiger Table Ersatz mit
    - TreeGrid
    - Drag and Drop Support
    - ComponentRenderer

- Viele andere kleine Verbesserungen

Vollständige Liste: https://vaadin.com/framework/whatsnew

Vaadin 8 kurz und knapp

Aber:

- Java 8 Required
- Ab Servlet API 3
- IE Support: Nur noch ab Version 11

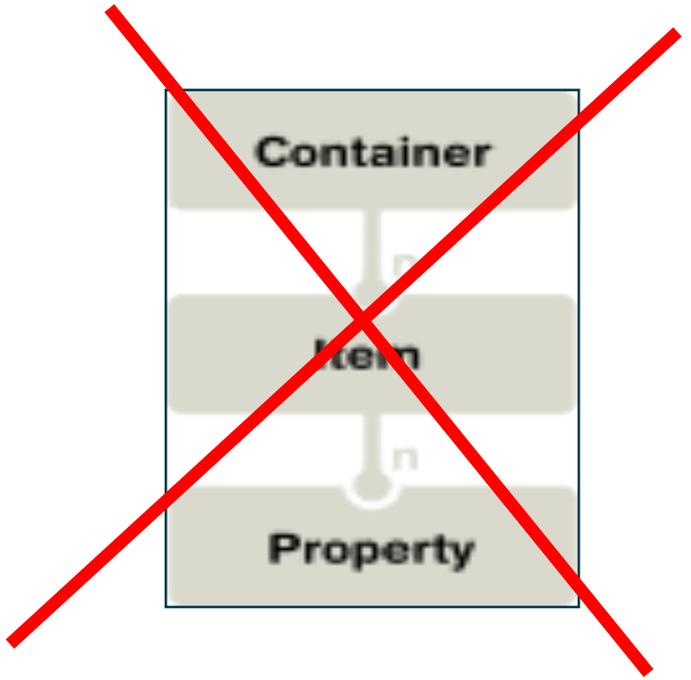
- In vielen Teile inkompatibel mit Vaadin 7
  - Felder/Formulare
  - Datenanbindung an Listen
  - Viele Plugins

Inside Vaadin 8

Wo ist mein Datenmodell?

Main API Change!!!

Vaadin 7
Data Model



#FFS

Fight For Simplicity

Datenmodell

- Das abstrakte Datenmodell wurde ersatzlos gestrichen

  -> Es wird jetzt direkt mit den Pojos gearbeitet

  -> Databinding auf Ebene Datenmodell gibt es nicht mehr!

- Aber: Jeglicher Zugriff auf die Daten vom UI Code ist nun typisiert anstelle von JavaBean Property Notation und Introspection

  -> Keine Laufzeitfehler mehr

  -> Deutliche Performance Verbessung bei grösserer Datenmenge im UI

Beispiel: Grid

```
Grid grid = new Grid();
grid.setContainerDataSource(
  new BeanItemContainer<>(persons));
grid.removeAllColumns();
grid.addColumn("firstName");
grid.getColumn("firstName")
    .setHeaderCaption("First Name");
grid.addColumn("lastName");
```

VF7

```
Grid<Person> grid = new Grid<>();
grid.setItems(persons);
grid.addColumn(Person::getFirstName)
  .setCaption("First Name");
grid.addColumn(Person::getLastName)
  .setCaption("Last Name");
```

VF8

Alt

- Zugriff auf Datenquelle indirekt über Container Implementierung
- Zugriff über String-basierte Property Notation

  -> Laufzeitfehler!

Neu

- Typisiert über die Pojo als Datenquelle
- Beans werden direct als Datenquelle gesetzt (setItems())
- Spaltendefinition mittels typensicherer lambda Ausrücke

Beispiel: Dropdown mit Custom Caption

VF7

```
List<Person> persons = Backend.getPersons();
ComboBox comboBox = new ComboBox();
BeanItemContainer<Person> bic =
    new BeanItemContainer<>(persons);
GeneratedPropertyContainer gpc =
    new GeneratedPropertyContainer(bic);
gpc.addGeneratedProperty("name",
    new PropertyValueGenerator<String>() {
  @Override
  public String getValue(
      Item i, Object id, Object o1) {
    Person p = (Person) id;
    return
        p.getFirstName() + " " + p.getLastName();
  }

  @Override
  public Class<String> getType() {
    return String.class;
  }

  @Override
  public Container.Filter
      modifyFilter(Container.Filter filter)
      throws UnsupportedFilterException {
    final SimpleStringFilter ssf =
        (SimpleStringFilter) filter;
    return new Container.Filter() {
      @Override
      public boolean passesFilter(
          Object itemId, Item item)
          throws UnsupportedOperationException {
        final Item genItem = gpc.getItem(itemId);
        String fullname =
            genItem.getItemProperty("name")
```

VF8

```
List<Person> List<Person> persons = Backend.getPersons();
ComboBox<Person> comboBox = new ComboBox<>();
comboBox.setItemCaptionGenerator(
  p -> p.getFirstName() + " " + p.getLastName()
);
comboBox.setItems(persons);
```

Neu

- Typisiert über die Pojo als Datenquelle
- CaptionGenerator definiert mittels lambda Ausdruck

Beispiel: Change Listener

```
comboBox.addValueChangeListener(evt -> {
    Person p =
        (Person) evt.getProperty().getValue();
    assert(p.getId() == 42);
});
```

VF7

Requires casting

```
comboBox.addValueChangeListener(evt -> {
    assert(evt.getValue().getId() == 42);
});
```

VF8

Alt

Listener Klassen greifen über die Datenmodell Abstraktion auf die Parameter zu (Item, Property) und arbeiten mit java Object. Sie müssen stets gecastet werden.

Neu

- Listener sind parametrisiert mit den korrekten Datetypen und erlauben so eine klare Implementierung der Handler Logik

Backend Anbindung

Alt

Bisher wurde das Backend immer über die das Vaadin DataModel angebunden:
Container API

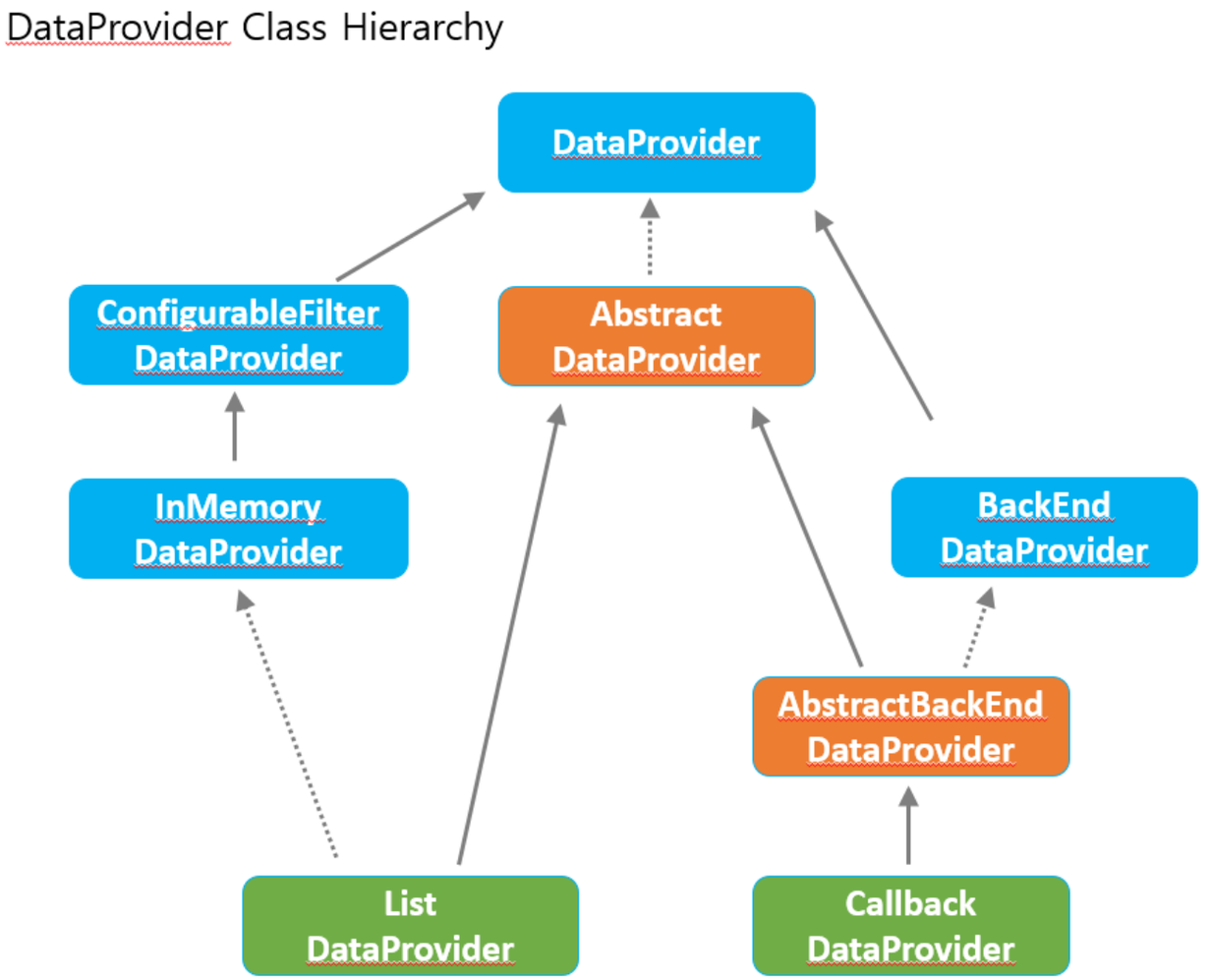
- Implementierungen:
  - In Memory (BeanItemContainer)
  - SqlContainer
  - JPAContainer
  - LazyQueryContainer

```
public void useReflectionBasedContainer() {

    List<Customer> customers = CustomerService.getInstance().findAll();

    BeanItemContainer<Customer> vaadinDataModel =
        new BeanItemContainer<Customer>(Customer.class);
    vaadinDataModel.addAll(customers);

    table.setContainerDataSource(vaadinDataModel);
}
```

Backend Anbindung

Neu: DataProvider API

- Class Hierarchy
  - Two core Interfaces:
    - InMemory
    - BackEnd
- With two core implementations
  - ListDataProvider
  - CallbackDataProvider



Backend DataProvider

```
DataProvider<File, String> dataProvider = DataProvider.fromFilteringCallbacks(
    query -> query.getFilter()
            .map(filter -> Stream.of(file.listFiles((file, name) -> name.endsWith(filter))))
            .orElse(Stream.of(file.listFiles()))
            .skip(query.getOffset()).limit(query.getLimit())
    ,
    query -> query.getFilter()
            .map(filter -> file.listFiles((file, name) -> name.endsWith(filter)).length)
            .orElse(file.listFiles().length));
```

Backend DataProvider

```
public class CustomerBackEndDataProvider
        extends AbstractBackEndDataProvider<Customer, String> {

    private CustomerService customerService;
    private String nameFilter;

    public CustomerBackEndDataProvider(CustomerService customerService) {
        Objects.requireNonNull(customerService);
        this.customerService = customerService;
    }

    public void setFilter(String name) {
        nameFilter = name;
    }

    @Override
    protected Stream<Customer> fetchFromBackEnd(Query<Customer, String> query) {
        return customerService.findAll(
            nameFilter, query.getOffset(), query.getLimit()).stream();
    }

    @Override
    protected int sizeInBackEnd(Query<Customer, String> query) {
        return Long.valueOf(customerService.count(nameFilter))
            .intValue();
    }
}
```

Formulare

Alt

FieldGroup mit Binder

- Nicht type safe
- Zugriff über Item Datenmodell über Property Notation
- Validatoren/Converter auf den Feldern und fix im Lifecycle verknüpft
- Massig Boilerplate:
  - Commit über checked CommitException

```
@Override
protected void init(VaadinRequest vaadinRequest) {
    VerticalLayout layout = new VerticalLayout();

    TextField firstName = new TextField("Firstname");
    TextField lastName = new TextField("Lastname");

    CustomerDTO customer = new CustomerDTO();

    FieldGroup fieldGroup = new FieldGroup();
    fieldGroup.bind(firstName, "firstName");
    fieldGroup.bind(lastName, "lastName");
    fieldGroup.setItemDataSource(new
         BeanItem<CustomerDTO>(customer));

    Button save = new Button("Save", new Button.ClickListener()

        @Override
        public void buttonClick(ClickEvent event) {
            try {
               fieldGroup.commit();
            }
            catch(CommitException e) {
               // show errors
            }
        }
    });
```

vaadin}>

Formulare

Neu

Bean typed helper class

- Contains a set of bindings for a bean property to a UI field
- Typed binding: Use callbacks to access bean properties
  - method references
  - lambdas or
  - callback interface instances

```
Binder<Customer> binder = new Binder<>(Customer.class);
binder.forField(firstName).bind(Customer::getFirstName, Customer::setFirstName);
binder.forField(lastName).bind(Customer::getLastName, Customer::setLastName);
binder.readBean(customer);

Button save = new Button("Save", e -> binder.writeBeanIfValid(customer));
```

Formulare

Validierung

Validierung und Konvertierung kann jetzt fliessend
und in beliebiger Abfolge definiert werden

```
new Binder<Person>().forField(tf)
    .withValidator(str -> str.length() == 4, "Must be 4 chars")
    .withConverter(new StringToIntegerConverter("Must be Integer"))
    .withValidator(integer -> integer.equals(2017), "Wrong date")
    .bind(Person::getBirthYear, Person::setBirthYear);
```

VF8

Aber meine Projekte…

Migration from Version 7

- Vaadin hilft:
  - ***Compatibility packages***: Deprecated Klassen wurden in ein anderes Package verschoben (`com.vaadin.v7.*`) und können mittles separater jar Pakete eingebunden werden:
    - vaadin-compatibility-server
    - vaadin-compatibility-client
    - vaadin-compatibility-client-compiled
    - vaadin-compatibility-shared
    - vaadin-compatibility-themes
  - ***Migration Tool:*** Maven plugin welches die Imports in Java Klassen auf die legacy Paketnamen anpasst:

    ```
    mvn vaadin:upgrade8
    ```

Migration from Version 7

- Plugins
  - Viele verbreitete Plugins sind bereits für Vaadin 8 aktualisiert
  - Notfalls müssen die Sourcen selber via compatibility packages und migration tool migriert werden

Migration from Version 7

- Refactoring
  - Folgender Code kann zwar über die compatibility packages weiter verwendet werden, ist aber nur sehr bedingt zu migrieren, und muss in weiten Teilen neu implementiert warden:
    - Formulare
      - Field Binding, Validatoren, Converter
    - Datenanbindung
      - Code mit Vaadin Datenmodell (Property, Item, Container) muss vollständig neu implementiert werden
    - Tables
      - Mit Grid neu implementieren

Challenges

Excel ähnliche Tabelle mit

- Hierarchischer Baum Navigation
- Vielen Zeile (>2000)
- Vielen Spalte (>50)
- Eingabeelemente in vielen Zellen (inline editing)
- Dynamisch berechnete Werte über die gesamte Tabelle verteilt
- Custom UI Funktionalität: Gruppierung von Kopfzeilen, Eingefrorene Spalten, Persistierte Spaltenkonfiguration, etc.

Challenges

	BW	BB	VS	FFK	ST | Bfm | FFK | Prod / Anz.	Plan Bfm	Plan Ums	Plan Prod	Δ BB Bfm	Δ BB Ums	Tool FFK	Tool STiAnz.	Tool Bfm	Tool Ums	Tool Prod	Referenz (Cugy) STiAnz.	Referenz (Cugy) Bfm	Referenz (Cugy) Ums	Referenz (Cugy) Prod
▼ Near-/Nonfood						127.56	2'308'558	18'098	0.71	234'987			127.56	2'308'558	18'098		100.59	1'639'090	16'295
▸ BEAUTY SELFCARE WORLD	09					25.00	671'990	26'880	0.42	65'466			25.00	671'990	26'880		23.74	521'158	21'953
▸ DAMEN-WELT	10					11.50			-1.67	-76'451			11.50				3.80		
▸ HERREN-WELT	11					4.94			-0.16	-43'159			4.94				2.20	25'444	11'565
▼ SCHUHE	12					5.25			1.32	-19'532			5.25				0.95	9'439	9'936
SCHUHE TABLAR	12	01																	
▸ SCHUHE	12	03		C	C	5.25			1.32	-19'532	C		5.25				0.95	9'439	9'936
▸ BABY	13					11.73	139'710	11'910	0.18	2'601			11.73	139'710	11'910		8.16	119'424	14'635
▸ KINDER	14					10.74	89'830	8'364	-2.70	-26'296			10.74	89'830	8'364		5.35	68'289	12'764
▼ HAUSHALT	15					56.20	1'377'306	24'507	2.85	360'747			56.20	1'377'306	24'507		55.15	815'944	14'795
▼ WASCHEN/PAPIER/REINIGEN	15	01		B	B	16.00	517'680	32'355	2.39	156'399	B		16.00	517'680	32'355		14.43	299'490	20'755
SCANNING HKS	15	01	2158		---						B							18	
KASSENZONE WASCHEN REINIG	15	01	2160		---						B								
KASSENZONE PAPIER DIVERSE	15	01	2186		---						B								
WASCHMITTEL	15	01	2675		004 | 3.00 | B | 36'500	3.00	109'500	36'500			B	004	3.00	109'500	36'500	004	2.28	39'328	17'249
SPEZIALWASCHMITTEL	15	01	2677		004 | 1.00 | B | 34'200	1.00	34'200	34'200			B	004	1.00	34'200	34'200	004	1.00	16'484	16'484
TEXTILPFLEGE	15	01	2678		004 | 1.00 | B | 23'800	1.00	23'800	23'800			B	004	1.00	23'800	23'800	004	0.72	10'073	13'990
GESCHIRR	15	01	2679		004 | 1.20 | B | 50'900	1.20	61'080	50'900			B	004	1.20	61'080	50'900	004	1.20	40'003	33'336
REINIGUNGSMITTEL	15	01	2680		004 | 3.20 | B | 27'600	3.20	88'320	27'600			B	004	3.20	88'320	27'600	004	3.30	60'497	18'333
TOILETTENPAPIER	15	01	2681		003 | 3.00 | B | 34'300	3.00	102'900	34'300			B	003	3.00	102'900	34'300	002	2.65	65'234	24'617
HAUSHALTPAPIER	15	01	2682		003 | 2.00 | B | 21'700	2.00	43'400	21'700			B	003	2.00	43'400	21'700	003	1.78	30'278	17'010
INSEKTIZIDE	15	01	2683		004 | 0.40 | B | 47'100	0.40	18'840	47'100			B	004	0.40	18'840	47'100	004	0.36	15'055	41'819
SCHUHPFLEGE	15	01	2684		002 | 0.20 | B | 27'200	0.20	5'440	27'200			B	002	0.20	5'440	27'200	002	0.14	2'099	14'994
LUFTERFRISCHER	15	01	2685		004 | 1.00 | B | 30'200	1.00	30'200	30'200			B	004	1.00	30'200	30'200	004	1.00	20'421	20'421
▸ CATEGORY VERBRAUCH	15	02		B	B	10.60	170'130	16'050	-1.55	4'728	B		10.60	170'130	16'050		13.20	147'587	11'181
▸ CATEGORY HAUSHALT ACC.	15	03		B	B	5.50	144'574	26'286	0.58	44'947	B		5.50	144'574	26'286		5.00	70'878	14'176
▸ CATEGORY KÜCHE	15	04		B	B	7.40	143'076	19'335	-0.47	16'279	B		7.40	143'076	19'335		8.43	108'590	12'881
▸ CATEGORY TISCH	15	05		B	B	6.70	109'499	16'343	0.59	30'215	B		6.70	109'499	16'343		6.58	63'314	9'622
▸ PAPETERIE	15	06		B	B	10.00	155'327	15'533	1.31	31'226	B		10.00	155'327	15'533		7.51	97'851	13'029
▸ KEHRICHTGEBUEHR	15	07		B	B		137'020			76'953	B			137'020				28'235	

BEAUTY SELFCARE WORLD

Vorgehen/Historie

- Vaadin Pre-8: Table/TreeTable
  - Unterstützt Komponenten in Zellen
  - Spaltenzeile in HTML über Client-side Hack
  - Probleme:
    - Performance Probleme (Usability, Akzeptanz)
    - Springendes UI Verhalten
    - Kaum wartbare Codebasis durch Vererbung von Vaadin Klassen und Hacks im Client gwt Code

Vorgehen/Historie

- Vaadin 8.0:
  - Grid anstelle von Table, aber
    - Kein Tree Support -> Navigation selbst implementieren
    - Kein drag und drop support
    - Keine native Unterstützung von Komponenten in Zellen
    - Inline Input Optionen:
      - Grid editor -> Zu träges UI Verhalten
      - ComponentRenderers über 3rd Party Plug-ins

Vorgehen/Historie

- Vaadin 8.1:
  - TreeGrid
  - Native ComponentRendererSupport
  - Drag and Drop

Kontakt

mp technology AG
Hardturmstrasse 76
8005 Zürich
+41 44 296 67 00
contact@mptechnology.ch

Jakob Magun
Managing Partner

+41 44 296 67 04
magun@mptechnology.ch

Benjamin Schupp
Lead Architect

+41 44 296 67 03
schupp@mptechnology.ch