3.– 6. September 2012
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Modellgetrieben

Vom Modell zur GUI mit der UI Description Language

Dr. Shota Okujava & Ralf Quaas

Isento GmbH

isento
IT-Beratung & Services

Vom Modell zur GUI mit der UI Description Language

Isento GmbH

Dr. Shota Okujava / Ralf Quaas

isento
IT-Beratung & Services

Idee

Eine technologieunabhängige Sprache zur **vollständigen (!)** Beschreibung der grafischen Benutzeroberflächen

isento
IT-Beratung & Services

Zielsetzung

- Standardisierung
- Aufwandsminimierung

isento
IT-Beratung & Services

Zielgruppe

- Anforderungsanalysten
- Systemanalysten
- Designer

isento
IT-Beratung & Services

Umsetzung

- Mehrere DSLs
- Editor im RCP

Aufbau UI-DSL

- Die Grammatik wird aufgeteilt in einzelne Module (DSLs), um die Abhängigkeiten leichter verwalten zu können
  - Infrastructure
  - Expression
  - Persistence/Service Model
  - Sample Data
  - GUI
  - Element Catalog
  - Structure Pattern
  - Plausibilitäten
  - Activity

Übersicht: Generatoren

- RichFaces
- ADF
- Preview/Screenshot
- Specification
- ElementCatalog-Export

isento
IT-Beratung & Services

Live-Demo

isento
IT-Beratung & Services

Infrastructure

- Stellt die Basis aller DSLs zur Verfügung
- Beinhaltet abstrakte Elemente, die von den abgeleiteten DSLs konkretisiert werden
- Beispiele:
  - BasicGUIObject
  - DataBinding
  - ComplexPathElement
  - Parameter
  - PersistenceType
  - ServiceDataType
  - StaticLiteral
  - …

isento
IT-Beratung & Services

Expression

- Ermöglicht eine formale Modellierung komplexer Ausdrücke, die vom Generator ausgewertet werden können
- Beispiel:
  - "Hallo " + "Welt" + "!"
  - 1 + 2
  - $kunde.istAmProduktInteressiert && $kunde.istZahlungsfähig
  - TODAY - 18DAYS
  - (1+2 > 3)?"wahr":"falsch„
  - $kunde.status == #KundenStatus.REGISTERED
  - $kunde->adressen[LAST].strasse == "Nürnbergerstraße"

Persistence/Service Model

- Ermöglicht die Modellierung der Persistenzschicht und Session Beans als Schnittstelle zum Client

```
DomainObject Customer {
    long id key;
    String firstname notNull minLength 2;
    Lastname lastname notNull minLength 2;
    Salutation salutation;
    Date birthday;
    String note maxLength 200;
    #+Address addressList;
    #Address currentAddress;
}


Service CustomerService{
    +Customer findCustomer(String lastName) transactional TX_MANDATORY rolesAllowed (Registered);
}
```

Sample Data

- Ermöglicht die Definition der Beispieldaten, die in Vorschau, Screenshots oder für Tests genutzt werden

```
EntityData SampleCustomer type Customer {
    Attribute salutation = #Salutation.MR;
    Attribute firstname = "Max";
    Attribute lastname = "Mustermann";
    Attribute birthday = AsDate("10.10.1966");
    Attribute note = "Dies ist eine Notiz!";
    Reference addressList = CustomerAddress1, CustomerAddress2;
    Reference currentAddress = CustomerAddress1;
}

EntityData SampleCustomer2 type Customer {
    Attribute salutation = #Salutation.MRS;
    Attribute firstname = "Gerda";
    Attribute lastname = "Musterfrau";
    Attribute birthday = TODAY - 18YEARS - 2WEEKS;
    Reference addressList = CustomerAddress1, CustomerAddress2;
    Reference currentAddress = CustomerAddress1;
}
```

GUI-DSL

- Ermöglicht die Definition der Dialog und Maskenmodelle
- Definiert die UI-Komponenten in einer Bibliothek, die beliebig erweiterbar ist

```
section FormularSection layout Panel[layoutColumns 1, title "Stammdaten"] {
    comp Gender data $customer.salutation lazyLoad label "Anrede:" #Textfield;
    comp Firstname data $customer.firstname lazyLoad label "Vorname:" #Textfield[optional false];
    comp Lastname data $customer.lastname lazyLoad label "Nachname:" #Textfield[optional false];
}

section ReiterSection layout TabbedPanel{
    section Adresse1 {
        comp Street1 data $customer->addressList[0].street lazyLoad label "Straße:" #Textfield;
        comp Postcode1 data $customer->addressList[0].postcode lazyLoad label "PLZ:" #Textfield;
        comp City1 data $customer->addressList[0].city lazyLoad label "Ort:" #Textfield;
        comp Country1 data $customer->addressList[0].country lazyLoad label "Land:" #Textfield;
    }
    section Adresse2 {
        comp Street2 data $customer->addressList[1].street lazyLoad label "Straße:" #Textfield;
        comp Postcode2 data $customer->addressList[1].postcode lazyLoad label "PLZ:" #Textfield;
        comp City2 data $customer->addressList[1].city lazyLoad label "Ort:" #Textfield;
        comp Country2 data $customer->addressList[1].country lazyLoad label "Land:" #Textfield;
    }
}
```

Element Catalog

- Ermöglicht die Definition der langfristig stabilen IDs für die UI-Komponenten
- Wird generiert und selektiv mit dem Maskenmodell abgeglichen

```
Module CustomerEditScreen {
    Screen CustomerEditScreen Element MainSection (c0);
    Screen CustomerEditScreen Element FormularSection (c1);
    Screen CustomerEditScreen Element Gender (c2) $customer.salutation;
    Screen CustomerEditScreen Element Firstname (c3) $customer.firstname;
    Screen CustomerEditScreen Element Lastname (c4) $customer.lastname;
    Screen CustomerEditScreen Element ReiterSection (c7);
    Screen CustomerEditScreen Element Adresse1 (c8);
    Screen CustomerEditScreen Element Street1 (c9) $customer->addressList[0].street;
    Screen CustomerEditScreen Element Postcode1 (c10) $customer->addressList[0].postcode;
    Screen CustomerEditScreen Element City1 (c11) $customer->addressList[0].city;
    Screen CustomerEditScreen Element Country1 (c12) $customer->addressList[0].country;
    Screen CustomerEditScreen Element Adresse2 (c13);
    Screen CustomerEditScreen Element Street2 (c14) $customer->addressList[1].street;
    Screen CustomerEditScreen Element Postcode2 (c15) $customer->addressList[1].postcode;
    Screen CustomerEditScreen Element City2 (c16) $customer->addressList[1].city;
    Screen CustomerEditScreen Element Country2 (c17) $customer->addressList[1].country;
}
```

isento
IT-Beratung & Services

Structure Pattern

- Definiert die Schablonen der Masken oder Maskenteile
- Die Schablonen werden als Vorlage für neue Masken/Abschnitte oder auch zur Validierung der Maskeninhalte genutzt

```
section pattern Ergebnisabschnitt NONSTRICT{
    component ref (Ergebnistabelle || Ergebnisliste) multiplicity ONE alert type ERROR;
    section ref ErgebnistabelleAktionen multiplicity ONE alert type ERROR;
}

section pattern ErgebnistabelleAktionen STRICT{
    component ref BackButton multiplicity ONE alert type ERROR;
    component ref Button multiplicity ONE alert type ERROR [caption "Auswählen" position 1];
}

component pattern InputFeld {
    capability INPUT requireType "Textfield" [position 1]
}
```

isento
IT-Beratung & Services

Plausibilitäten

- Ermöglicht die Definition der Textmeldungen und der Plausibilitätsbedingungen, die dann von der Maske referenziert werden können

```
Messages MyMessages{
    FirstnameTooLong "Der eigngegebene Vorname ist zu lang..." icon "warnung.gif";
    LastnameTooShort "Der eingegebene Nachname ist zu kurz,
                        Sie müssen mindestens zwei Zeichen eingeben";
    LastnameInvalid "Der eingegebene Name ist nicht zulässig. Bitte ändern Sie den Namen";
}
Module CheckCustomerInput{
    data firstname type String
    data lastname type String

    Check CheckFirstnameTooLong : LENGTH $firstname > 100 Message FirstnameTooLong;

    Check CheckLastnameTooShort : LENGTH $lastname < 2 Message LastnameTooShort;

    Check IllegaleNachnamen : $lastname
                    %%Der Nachname darf folgende Zeichenketten nicht enthalten...%%
                        Message LastnameInvalid;

    Group CheckFirstAndLastname {
        elementRef CheckFirstnameTooLong
        elementRef CheckLastnameTooShort
    }
}
```

Activity

- Ermöglicht eine textuelle Darstellung eines UML-Aktivitätsdiagramms
- Activity-Modelle werden genutzt, um eine Maskenübergreifende semantische Validierung des Objektflusses durchzuführen

```
activity ModifyCustomer parameters (
        IN customerId type long
        OUT customer type Customer
    ) {

    start StartActivity;

    callOperation GetCustomer ref CustomerService.getCustomer parameters (
        IN customerId type long
        OUT customer type Customer
    );

    callBehavior EditCustomerData dialog CustomerManagementDialog
            parameters (
                IN_OUT dialogCustomer type Customer
                IN_OUT dialogCustomerOverview type CustomerOverview
            );

    end FinishModify;
    end CancelModify;

    objectFlow from customerId to GetCustomer.customerId;
    objectFlow from GetCustomer.customer to EditCustomerData.dialogCustomer;
    objectFlow from EditCustomerData.dialogCustomer to customer;
}
```

isento
IT-Beratung & Services

Vom Modell zur GUI mit der UI Description Language

Isento GmbH

Dr. Shota Okujava / Ralf Quaas

Vielen Dank!

Dr. Shota Okujava & Ralf Quaas

Isento GmbH