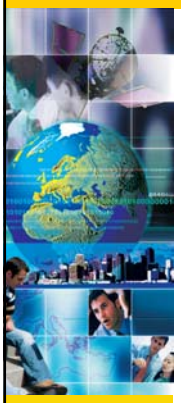
Groupware Competence Center GCC



Workgroup Computing Praktikum

Using Formula Language in Forms

University of Paderborn
Business Computing 2 – Information Management & Office Systems
Faculty of Business Administration, Business Computing & Economics
Prof. Dr. Ludwig Nastansky
Warburger Str. 100, D-33098 Paderborn
Tel.: +49--5251--60-3368
http://gcc.upb.de

Formula Language in Forms

- **Automatisches Hinzufügen, Berechnen und Formatieren von Daten**
- **Beispiele für die Verwendung von Formula Language:**
  - Anzeige eines Fenstertitels (Window Title) in Abhängigkeit von seinem Zustand (neues Dokument vs. bestehendes Dokument)
  - Berechnung eines Enddatums in Abhängigkeit vom Startdatum und der Dauer
  - Bestimmung des initialen Erstelldatums eines Dokumentes
  - Dynamische Berechnung der Werte einer Auswahlliste
  - Prüfung von Benutzereingaben auf Gültigkeit (Input Validation)
  - Anpassung von Benutzereingaben (Input Translation)
  - Realisation von Benutzerinteraktionen mittels Dialogboxen

WindowTitle

- **Forms und Pages können einen WindowTitle haben**
- **Guideline: Immer einen WindowTitle vergeben!**
- **Dynamische Berechnung mittels Formeln ist möglich**
  - Hilfreiche @function: @IsNewDoc
    - Rückgabewert 1 (true) falls das Dokument noch nicht gespeichert wurde
    - Rückgabewert 0 (false) falls das Dokument bereits gespeichert wurde
    - Beispiel für einen dynamischen WindowTitle:
      @If (@IsNewDoc; „Neues Dokument“; Name)

Übung

Einen dynamischen Titel für die Order-Form erstellen

- **Der Window Title der Order-Form soll nun dynamisiert werden. Falls eine neue Order angelegt wird, soll dies auch im Window Title berücksichtigt werden. Z.B. in der Form „New Order“. Implementiere dies auch analog für den Edit-mode.**

Computed for Display Felder

- **Werden jedes mal neu berechnet, wenn das Dokument geöffnet, aktualisiert und gespeichert wird**
- **Computed For Display Fields müssen eine Formel enthalten**

- **Der Inhalt solcher Felder wird nicht im Dokument gespeichert**
  - ACHTUNG: für Berechnungen nicht geeignet!

Computed for Display Felder benutzen, um…	**Beispiel**
Die aktuelle Zeit anzuzeigen	**@Now**
Den Benutzernamen anzuzeigen	**@UserName**
Das Erstelldatum des Dokumentes anzuzeigen	**@Created**
Den Inhalt eines Listenfeldes, wenn das Dokument geöffnet wird	WorkType

Computed Text

- **Kann auf Forms oder Pages hinzugefügt werden**

- **Ähnlich den Computed for Display Fields**
  - Beachte: der Rückgabewert darf keine Liste sein und muss aus Text bestehen

- **Kann wie jeder andere Text formatiert werden**
  - Computed Text Properties

Computed Text benutzen, um…	**Beispiel**
Text abhängig von einer Bedingung anzuzeigen	@If ((DueDate < @Today) & (Status != „Completed“); „This ist late“; ““)
Dynamische Nachrichten	“Welcome, “+ @Name([CN];@UserName)

Berechnete Werte speichern

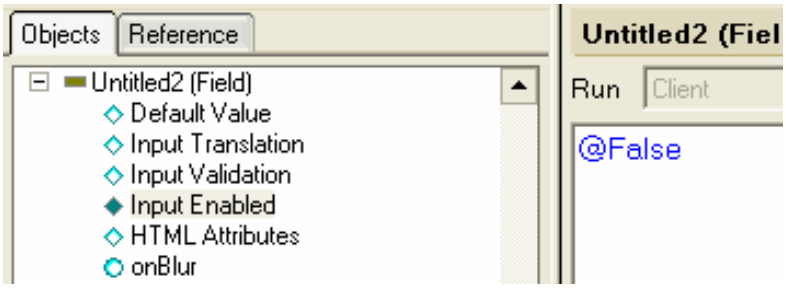
- **Um Feldinhalte in einem Dokument zu speichern, muss einer der folgenden Feldtypen benutzt werden:**
  - Computed
  - Computed when composed
  - Editable

Feld	Berechnet einen Wert	Anwendung
Computed	Jedes Mal wenn das Dokument erstellt, aktualisiert oder gespeichert wird	Ergebnis einer Berechnung abhängig von anderen Feldern, z.B. Gesamtsumme
Computed when composed	Das erste Mal wenn das Dokument gespeichert wird	Den ursprünglichen Autor Ein Wert, der sich nie ändert, z.B. Erstelldatum

Formeln in editable Fields

- **Um die Benutzerfreundlichkeit bei editable Fields zu erhöhen kann man folgende Funktionalitäten nutzen:**
  - Default Value (Vorgabewert)
    Um einen Default Value zu berechnen, wenn ein Dokument erstellt wird
  - Input Translation
    Benutzereingaben formatieren
    z.B. Telefonnummern mit Klammern, Bindestrich oder Punkten versehen
  - Input Validation
    Benutzereingaben auf Gültigkeit überprüfen
    - Verhindert das Speichern von Dokumenten, mit unvollständigen und/oder falschen Dateneingaben

- Input Enabled
  - **Bearbeiten von Feldern erlauben oder verbieten**
  - Die Formel sollte **<true>** oder **<false>** liefern



- **Input Translation**
  - Wird beim Speichern oder Aktualisieren ausgeführt
  - Das Ergebnis muss den gleichen Datentyp haben wie das Feld

Function	Beschreibung
@Trim(String)	Entfernt überflüssige Leerzeichen
@ProperCase(String)	Jeder Anfangsbuchstabe wird groß geschrieben, alle anderen klein
@UpperCase(String)	Alle Buchstaben werden in Großbuchstaben transformiert
@LowerCase(String)	Alle Buchstaben werden in Kleinbuchstaben transformiert

Formeln in editable Fields

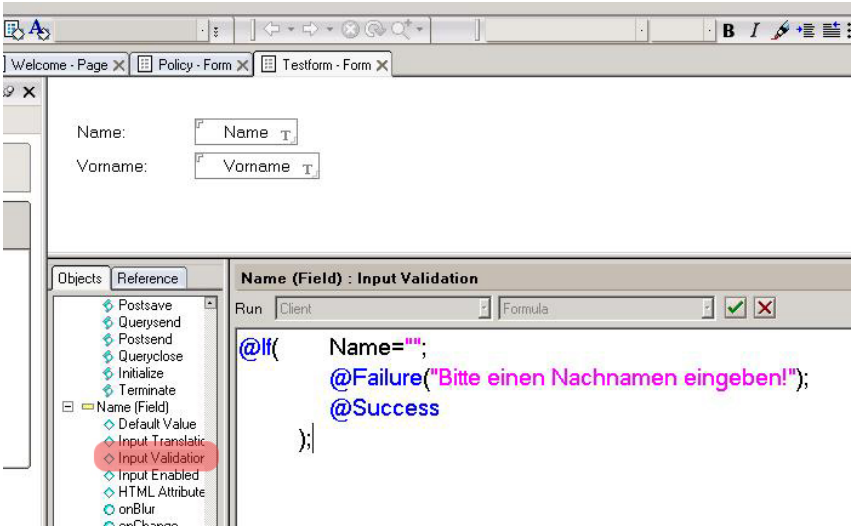
- **Input Validation**
- **Wird beim Speichern und Aktualisieren ausgelöst**
  - Überprüft die Gültigkeit eines Wertes in einem Feld
  - Setzt den Speichervorgang bei Gültigkeit fort
  - Sonst: erzeugt eine Fehlermeldung und bricht den Speichervorgang ab

- **Hinweis:**
  Die Input Validation wird **nach** der Input Translation ausgelöst

Wenn der Wert…	Benutze …	Um…
Richtig ist	@Success	Speichervorgang wird fortgesetzt
Falsch ist	@Failure(Meldung)	Eine Pop-up Messagebox mit Fehlermeldung anzuzeigen und den Speichervorgang abzubrechen

Formeln in editable Fields

- **Beispiel: Input Validation**



Order – Form erweitern

- Eine Order soll nur gespeichert werden können, wenn die Felder number, vehicle type, vehicle und owner ausgefüllt wurden.
- Der Vor- und Nachname des Besitzers sollte auch dann mit einem Großbuchstaben beginnen, wenn ein Mitarbeiter sich vertippt hat.
- Füge weiterhin zur der Form ein neues Feld „duedate“ hinzu. Dies soll das dem Kunden zugesicherte Fertigstellungsdatum des Auftrages speichern. Es soll weiterhin berücksichtigt werden, dass in der Regel die Aufträge nach drei Tagen abgearbeitet sind.

- Eine Liste enthält mehrere Werte des gleichen Datentyps

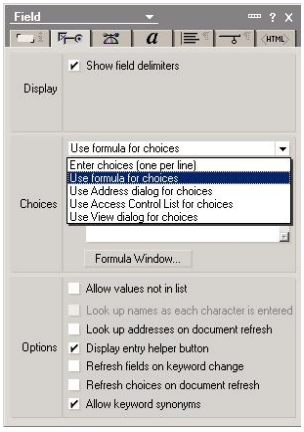
@Function	Beschreibung	Beispiel
@Elements(list)	Anzahl der Elemente der Liste	@Elements(Colors) **Liefert**: Anzahl der Elemente im Feld Colors
@Explode(string; separator)	Teilt einen String in eine Liste	@Explode(“Red Green Blue“; “ “) **Liefert**: “Red“:“Green“:“Blue“
@Implode(Liste; Separator)	Konvertiert eine Textliste in einen String	@Implode(“Rot“:“Grün“:“Blau“; “*“) **Liefert**: Red*Green*Blue
@IsMember(value; list)	Liefert wahr, falls der Wert in der Liste enthalten ist	@IsMember(“Orange“;“Red“:“Green“:“Blue“) **Liefert**: 0 (False)

Mit Listen arbeiten

@Function	Beschreibung	Beispiel
@Member(value;list)	Position des Werte in der Liste	@Member(“Green“; “Red“:“Green“:“Blue“) **Liefert:** 2
@Subset(List; number)	Liefert einen Teil der Liste mit Anzahl „number“ Elementen	@Subset(“Red“:“Green“:“Blue“; 2) **Liefert:** “Red“:“Green“
@Unique(list)	Entfernt doppelte (mehrfache) Einträge	@Unique(“Rot“:“Grün“:“Blau“: “Grün“:“Blau“) **Liefert:** “Rot“:“Grün“:“Blau“

Wiederholung zu Listenfeldern

- **Um Auswahlmöglichkeiten für Listenfelder (ComboBox, DialogList, …) zu erzeugen hat man folgende Möglichkeiten:**
  - Von Hand eintragen
    - Per Formel generieren
  - (siehe nächste Folie)
  - Per Adress-Dialog
  - Per ACL-Dialog
  - Per View-Dialog



Auswahl im Listenfeld per Formel berechnen

@DbColumn

- Liefert eine Spalte einer View als Liste
- Benutzen um die Auswahl eines Listenfeldes zu berechnen

Syntax:

@DbColumn(class:mode; server:database; view; columnNumber)

- *class:mode* - Typ der Datenbank und Zwischenspeichern des Ergebnisses
- *server:database* - Speicherort und Name der Datenbank
- *view* - Name der View
- *columnNumber* - Nummer der Spalte, die zurück gegeben werden soll

- Beispiel:

  @DbColumn("Notes":"NoCache";"";"In Progress";2)

  - Liefert die Werte 2. Spalte der View mit dem Namen oder Alias „In Progress" der aktuellen Datenbank als Liste

Übung

Dynamisierung

- **Die Auswahl des Fahrzeugtyps ist bisher statisch vorgegeben. Diese soll nun dynamisiert werden.**
- **Erstell dazu eine Form „Vehicle type" mit der neue Fahrzeugtypen angelegt werden können.**
- **Für diese neue Form sinnvoll in die vorhandene Outline mit ein.**
- **Lege mit dieser Form Dokumente mit folgenden Fahrzeugtypen an:**
  - Van
  - Truck
  - Passenger car
  - Pickup
  - Bus
- **Lege nun eine View an, die nur die Dokumente der Form „Vehicle type" anzeigt.**
- **Ändere die Order Form so, dass die Kategorieauswahl aus der View „Vehicle type" automatisch berechnet wird (Hinweis: benutzt @DBColumn)**

Interaktion mit dem Benutzer

- **@Prompt ermöglicht Interaktion mit dem Benutzer via Dialog Box**
  - Eingabe von Werten
  - Antworten auf Fragen
  - Auswahl aus einer Liste treffen

- **Syntax:**

@**Prompt**([*style*]:[NOSORT]; *title*; *prompt*; *defaultChoice* ; *choiceList* ; *filetype*)

style	Typ der Dialogbox
nosort	Sortierung der Liste
title	Titel der Dialogbox
prompt	Der Text, der angezeigt werden soll
defaultChoice	Wert, der als Vorgabe ausgewählt sein soll
choiceList	Werte die in der Dialogbox angezeigt werden sollen
filetype	Nur für den Style LOCALBROWSE interessant

- **Beispiel:**

@Prompt([OK]; “Prompt Box Titel“; “Hinweis an den Benutzer“)

Auslösen von @Prompt

Einsatzort	Einsatzszenario
Field Formulas	Benutze @Prompt, um den Benutzer zur Eingabe eines Wertes aufzufordern
Button formulas	@Prompt kann benutzt werden, um eine Eingabe vom Benutzer zu fordern
Action Buttons in einer Form oder View oder in manuel agents	z.B. kann man den Benutzer nach einem Wert fragen und dann ein Feld entsprechend setzen
Password-Field formulas	In Kombination mit @Password, kann @Prompt benutzt werden, um nach einem Passwort zu fragen

Form Events benutzen

Form Event	Wann wird das Event ausgeführt
QueryOpen	Bevor das Dokument geöffnet wird
PostOpen	Nachdem das Dokument geöffnet wurde und angezeigt wird
QueryModeChange	Bevor der Modus gewechselt wird (Editiermodus – Lesemodus)
PostModeChange	Nachdem der Modus gewechselt wurde
PostRecalc	Nach dem Aktualisieren (normalerweise nach drücken von F9)
QuerySave	Vor dem Speichern
PostSave	Nach dem Speichern
QueryClose	Vor dem Schließen

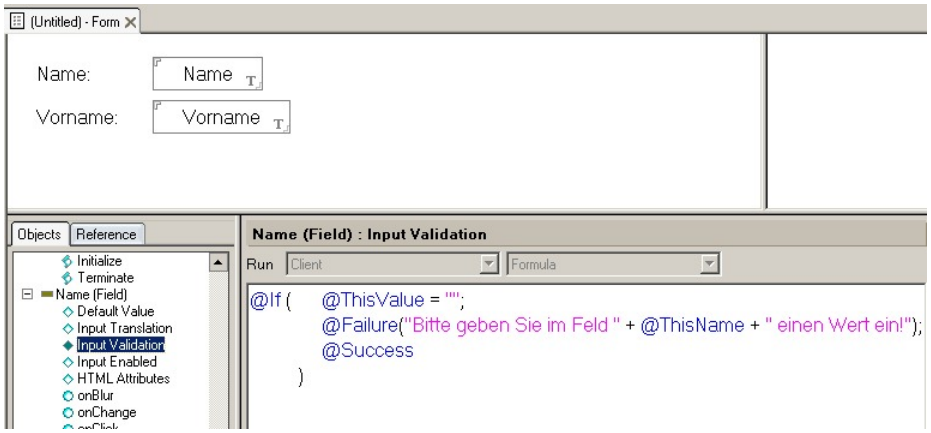
Wieder verwendbaren Code erstellen

- **Code erstellen, der an mehren Stellen verwendet werden kann ohne,**
  dass er geändert werden muss

- **@ThisValue**
  - Liefert den Wert des aktuellen Feldes
- **@ThisName**
  - Gibt den Namen des aktuellen Feldes zurück
- **Benutzen von @ThisValue und @ThisName**
  - Das hard-codieren von Feldnamen ist damit nicht nötig
  - Code einmal schreiben und per copy und paste in verschiedene Felder einfügen
  - Der Code muss nicht geändert werden, wenn sich der Feldname ändert
  - Aber: immer an die Verständlichkeit der Benutzerschnittstelle denken!

Beispiel

- **Verwendung von @ThisName und @ThisValue in einer Input Validation:**



Übung

Wiederverwendbaren Code erstellen:

- **In der Order-Form wurden bereits die Felder number, vehicle type, vehicle und owner dahingehend geprüft, ob sie mit Daten gefüllt wurden. Nun soll für diese Felder wieder verwendbarer Code benutzt werden.**
  - Erstelle eine entsprechende Formel
  - Füge sie in die genannten Felder ein
  - Speichere und teste die Form

Selbststudium

- **Wozu dienen Input Translations und Validations und wann werden sie ausgeführt?**