



ix INSIDE INTREXX

UNITED PLANET INTREXX XTREME
RELEASE 4.5

Inhaltsverzeichnis

1. Einführung 5

2. Businesslogik Arbeitsweise 5

3. Systemarchitektur 5

3.1. Wichtige Begriffe und Techniken 6

3.2. Verzeichnisstruktur 7

 3.2.1. Hauptverzeichnisse 7

 3.2.2. Verzeichnisaufbau eines Portals 7

 3.2.3. External/htmlroot 7

 3.2.4. Internal 8

 3.2.5. Application 9

 3.2.6. Layout 10

 3.2.7. system 10

3.3. Die Dateien object.js, form.js und api.js 10

4. Veröffentlichen einer Applikation 11

4.1. Aufruf einer Seite (1. Transformation) 11

4.2. Aufruf einer Seite (2. Transformation) 12

5. JavaScript 12

5.1. Requestparameter 14

5.2. Velocity 16

5.3. Berechnungen mit JavaScript 17

5.4. Ausblenden von Kontrollen 19

5.5. Einfügen von Text in Ansichtskontrollen 19

 5.5.1. Typdefinierte Ansichtselemente 21

5.6. Ein- und Ausblenden von Elementen 22

5.7. Ein- und Ausblenden beim Laden einer Seite 23

5.8. Kopieren eines Datensatzes mit JavaScript 24

5.9. Datumsberechnung mit JavaScript Teil I 25

 5.9.1. Syntax der Methode *dateAdd()* 26

5.10. Datumsberechnung mit JavaScript Teil II 26

 5.10.1. Syntax der Methode *setFullDays()* 27

5.11. Umgang mit Tabellen über ein JavaScript Objekt 27

 5.11.1. Tabellenspalte berechnen 28

5.12. Automatischer Reload einer Ansichtstabelle 31

5.13. Dynamischer Sprung aus Ansichtstabelle 31

6. Velocity 33

6.1. Was ist Velocity? 33

6.2. Wo wird Velocity angewendet? 33

6.3. Referenzen 33

 6.3.1. Variablen 33

 6.3.2. Properties 33

 6.3.3. Methoden 33

6.4. Kommentare 35

6.5. #set 35

6.6. #if / #else / #elseif 36

 6.6.1. #if 36

 6.6.2. #else 36


 6.6.3. #elseif 36

6.7. Relationale und logische Operatoren 36







6.8. #foreach.....	37
6.9. #include	37
6.10. #parse.....	37
6.11. #macro.....	37
6.12. Range Operator.....	38
7. Aufruf einer Seite (2. Transformation)	38
8. Kontextobjekte unter Velocity	38
8.1. \$User	39
8.2. \$Request	40
8.3. \$Loader (BusLogicCaller).....	40
8.4. \$null	41
8.5. \$charset	41
8.6. \$lang	41
8.7. \$UriBuilder	41
8.8. \$ObjectHelper	41
8.9. \$Factory	41
8.10. \$Response	42
8.11. \$Portal.....	42
8.12. \$Session	42
8.13. \$DbConnection	42
8.14. \$Request	42
8.15. \$Loader	42
8.16. \$Chat	42
8.17. \$OMUC	42
8.18. \$VelocityContext	42
9. Direkter Velocity-Aufruf.....	42
10. Statische VMs in Velocity	42
11. Query aus Velocity.....	43
11.1. Beispiel für PreparedQuery.....	43
11.2. Einfache Datenbankabfrage.....	44
12. Velocity und Parameter	45
12.1. Tabellennamen variabel abfragen	46
13. Schaltflächen	47
13.1. Eigenschaften und Methoden des Actioncontrol-Objektes.....	48
13.2. LinkType	49
13.3. ActionId	49
14. Ajax	49
14.1. Was ist Ajax	49
14.2. Ajax in Intrexx Xtreme	49
14.2.1. Das upXmlHttp-Objekt	50
14.2.2. ActionControls.....	50
14.2.3. Request-Verarbeitung	51
14.2.4. oRequest-Objekt	51
14.3. Ajax und XML-Response zurückgeben.....	52
14.3.1. Rückgabe von Werten	52
14.3.2. Requestwerte und Formularelemente mitsenden	52


Schreibkonventionen

In diesem Dokument werden Textstellen *kursiv* dargestellt, wenn sie sich auf Einstellungen in den abgebildeten Dialogen beziehen. Menüpunkte, die in Kontextmenüs erreichbar sind, sind immer auch über das Hauptmenü erreichbar. Hauptmenüpunkte werden nicht beschrieben, es sei denn, sie sind nicht über das Kontextmenü erreichbar. Kontextmenüs können mit einem Klick mit der rechten Maustaste auf das beschriebene Element geöffnet werden.

Eine Beschreibung der allgemeinen Hauptmenüpunkte finden Sie im Handbuch  *Center*. Programmiercode im Text wird in der Schriftart Courier dargestellt.

`<xtreme>` bezeichnet im Folgenden Ihren Intrexx Installationspfad, unter Windows z.B. `C:\xtreme\`, unter Linux z.B. `/opt/xtreme/`. Folgende Symbole werden für die Kennzeichnung von speziellen Informationen verwendet:

-  Wichtige Hinweise
-  Tipps und Hintergrundinformationen
-  Verweise auf weiterführende Informationen in einem Intrexx Xtreme Handbuch
-  Verzeichnisse
-  URLs
-  Schaltflächen in Dialogen oder Assistenten

Stellen mit dem Hinweis  **[Copy-Paste]** können direkt aus dem Dokument übernommen werden, um Schreibfehler zu vermeiden.

Vorkenntnisse

Für das Verständnis dieser Dokumentation sollten Sie Erfahrung im Umgang mit Intrexx Xtreme und in der Programmierung mit Java bzw. Groovy haben.

1. Einführung

Dieses Dokument gibt eine Einführung in die internen Strukturen von Intrexx Xtreme und erläutert den Einsatz von Expertattributen, JavaScript und der verwendeten XML/XSL Technologie. Sie erhalten dadurch einen Einblick in die Hintergründe und Möglichkeiten mit Intrexx Xtreme. Um praxisnah zu erfahren, wie Sie Eingriffe in das System vornehmen können, bietet United Planet die Seminare *Entwicklung I*, *Entwicklung II* und *Entwicklung III* an. In diesen Seminaren lernen Sie in kurzer Zeit, wie sich selbst komplexe Vorgänge abbilden lassen oder interaktive Webanwendungen mit Hilfe von Ajax realisiert werden können. Weitere Informationen zu den Seminaren erhalten Sie unter www.intrexx.com/Academy.

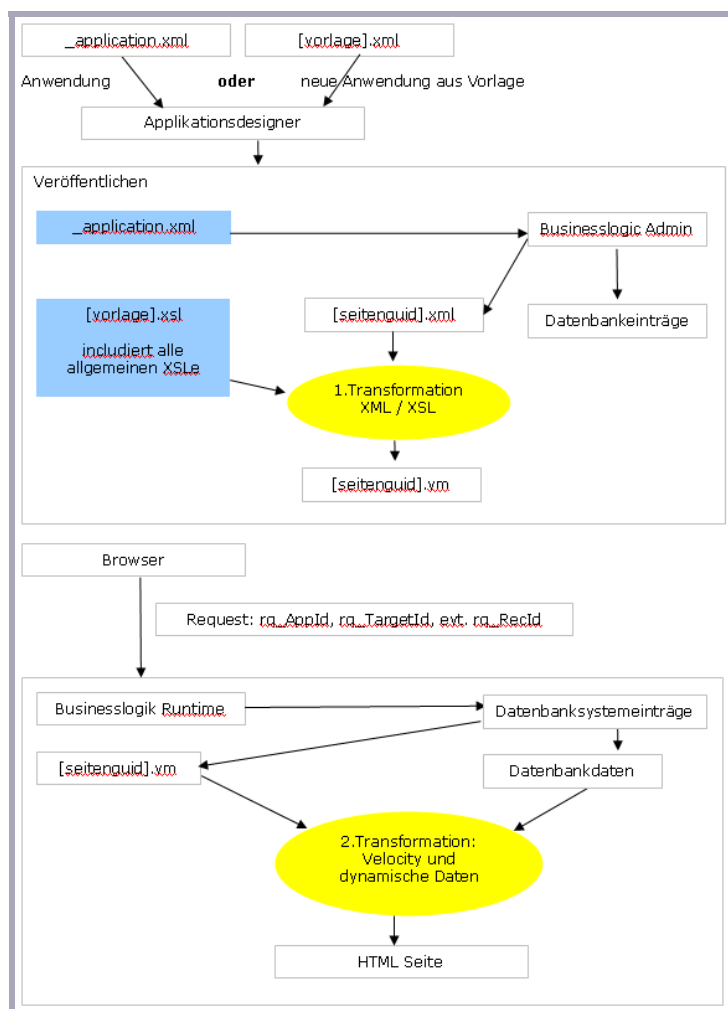
Tool Empfehlungen:

- XML/XSL Editor und Transformation
- XML-Spy <http://www.xmlspy.com> (homeedition)
- Microsoft Visual InterDev

Freie Editoren (UTF8 fähig):

- Notepad++ <http://notepad-plus.sourceforge.net/uk/download.php>

2. Businesslogik Arbeitsweise



3. Systemarchitektur

Der Intrexx Xtreme Portal Manager kommuniziert mit dem Intrexx Xtreme Application Server via SOAP auf der Basis von Apache Axis. SOAP ist ein XML basierendes Protokoll zur Kommunikation von Systemen in verteilten Umgebungen.

Der Manager erzeugt die Applikationen, deren Basisformat XML ist, durch eine Transformation via XSL in das Velocity Markup Format im gewünschten Ausgabeformat. Durch die Transformation können schnell neue Ausgabeformaten unterstützt und flexibel auf Änderungen in den Kommunikationsformen reagiert werden. Über die Verwendung von Velocity Markup ist eine hohe Performance sichergestellt.

Der Intrexx Xtreme Application Server übernimmt die Aufgaben

- Benutzerverwaltung und -authentifizierung
- Sessionmanaging
- Integration der Businesslogiken
- Zugriff auf die Datenbanken.

Über Konnektoren (im Lieferumfang enthalten) können verschiedene Webserver wie Apache, Tomcat und Microsoft Internet Information Server unterstützt werden.

3.1. Wichtige Begriffe und Techniken

XML

Extensible Markup Language, Weiterentwicklung des SGML-Standards. Im Gegensatz zu SGML-Dokumenten benötigen XML-Dokumente keine Schemabeschreibung in Form einer DTD. Sie bestehen hauptsächlich aus Text und Tags; die Tags erzeugen eine Baumstruktur im Dokument. Wenn das XML-Dokument ordnungsgemäß strukturiert ist, d.h. wenn eine ordnungsgemäße Verschachtelung der Tags vorliegt, wird es als *well-formed* (korrekt strukturiert) bezeichnet. Wenn zusätzlich eine DTD für das Dokument existiert, wird es als *valid* (gültig) bezeichnet.

XML-Data

Eine Methode für die Angabe von XML-Schemata. XML-Data erweitert die Fähigkeiten von DTDs auf Datentypen.

XML-Parser

Ein XML-Parser ist ein Verarbeitungsprogramm, das ein XML-Dokument liest und die Struktur sowie die Eigenschaften der Daten ermittelt. Es zerlegt die Daten in einzelne Bestandteile und reicht diese an andere Komponenten weiter. Wenn der Parser nicht nur anhand der XML-Regeln prüft, ob das Dokument *well-formed* (korrekt strukturiert) ist, sondern es auch anhand einer XML-DTD prüft, wird der Parser als *validating*-Parser bezeichnet. Es gibt verschiedene XML-Parser auf dem Markt:

- Xerces Java Parser von Apache/IBM
- XML4j von IBM
- XML Parser von Microsoft
- XML-Parser für Java von Oracle
- Lark von Tim Bray
- Project X von Sun
- XP von James Clark

XML Schema

Das XML Schema ist das aktuellste Verfahren zur Bereitstellung von XML-Schemata, das in zwei W3C-Arbeitsentwürfen (Structures und Datatypes) veröffentlicht wurde. Die Schemata dienen zur Beschreibung der Struktur und zum Beschränken des Inhalts von XML-Dokumenten sowie dem Zuordnen von Datentypen zu XML-Elementtypen und -attributen. Frühere Schemadesigns umfassen DTDs, XML-Data, XDR, RDF, SOX, DCDs, XSchema und DDML.

XPath

XML Path Language, eine Sprache zur Adressierung von Teilen eines XML-Dokuments, die sowohl von XSLT als auch von XPointer verwendet werden kann. Die Sprache besteht in der Hauptsache aus Adressierungspfaden und -ausdrücken:

durch *child::para[position=(1)]* könnte z.B. das erste Para Child des aktuellen Kontextknotens ausgewählt werden. Als Ausdrücke werden normale Elemente wie Boolesche Operatoren, Zahlen etc. und Node Sets verwendet.

XSD

XML-Schemadeklaration, ein Suffix von Dateien, die eine Beschreibung eines XML-Schemas gemäß der XML-Schemenspezifikation enthalten.

XSL

Extensible Stylesheet Language, eine Sprache zur Erstellung von Stylesheets, die beschreibt, wie Daten, die unter Verwendung der Extensible Markup Language (XML) über das Web gesendet werden, dem Benutzer präsentiert werden sollen.

XSLT

XSL Transformations, eine Sprache für die Umwandlung von XML-Dokumenten in andere XML-Dokumente. XSLT wurde als Teil von XSL, einer Stylesheet-Sprache für XML, entwickelt. XSL verfügt über den Umfang von XSLT hinaus über ein XML-Vokabular für die Angabe von Formatierungsinformationen. XSL bestimmt das Format eines XML-Dokuments durch Verwendung von XSLT. Mit XSLT wird beschrieben, wie das Dokument in ein anderes XML-Dokument transformiert wird, das das Formatierungsvokabular verwendet.

GUID

Global Unique Identifier – weltweit eindeutige Nummer

















Velocity

Javabasierte Template Engine, die es ermöglicht, in statische Webseiten dynamische Daten über Java-Klassenaufrufe einzubinden.






3.2. Verzeichnisstruktur

3.2.1. Hauptverzeichnisse

Der Verzeichnisbau von Intrexx Xtreme ist in folgende Hauptbereiche gegliedert:


-  *bin* ausführbaren Dateien
-  *cfg* allgemeine Konfigurationsdateien
-  *client* Dateien, die für den Portalmanager benötigt werden
-  *docs* Intrexx Xtreme Dokumentation
-  *export* Portalexporte
-  *groovy* Groovy Skript-Files
-  *help* Intrexx Xtreme Onlinehilfe
-  *hsqldb* HSQL-Datenbankdateien
-  *jre* Java Runtime Umgebung für Intrexx Xtreme
-  *lib* Programm Source Code
-  *log* Logdateien Portale
-  *org* alle angelegten Portale
-  *orgtempl* Portalvorlagen
-  *res* Ressourcen für Setup und Portal Manager
-  *tmp* Temporäre Dateien
-  *updlib* Quellen für Online-Update

3.2.2. Verzeichnisbau eines Portals

-  *org* Ein Portal ist immer in die Hauptbereiche *internal*, *external* und *log* unterteilt.
 -  *entwicklung*
 -  *external*
 -  *internal*
 -  *log*

3.2.3. External/htmlroot

Das Verzeichnis *external/htmlroot* enthält alle Dateien, die über den Webserver direkt erreicht werden können. Dazu gehören:

-  *css*
Cascading Stylesheet Dateien (Standard und über den Portaldesigner erzeugte)

 *download*

Setupdateien von Downloads (z.B. Office Integration)

 *htc*

JavaScript-Dateien, die CSS Verhalten für den IE 6.0 simulieren, da der IE 6.0 eine äußerst lückenhafte CSS Implementierung hat.

 *html*

Sonstige statische HTML-Dateien

 *images*

Bilddateien, die von Intrexx Xtreme verwendet werden. Werden Bilder direkt in Applikationen eingefügt, so wird in diesem Verzeichnis automatisch ein Verzeichnis mit dem Namen der GUID der Applikation erstellt. In diesem Verzeichnis sind die Bilddateien der Applikation enthalten.




Änderungen an den Bilddateien in diesem Verzeichnis werden beim nächsten Speichern der Applikation überschrieben. Nehmen Sie Änderungen an Bilddateien bitte nur direkt im Applikationsdesigner vor.

 *include*

JavaScript Dateien, die von Intrexx Xtreme eingesetzt werden. Ist JavaScript in Applikationen eingesetzt, so wird in diesem Verzeichnis automatisch ein Verzeichnis mit dem Namen der GUID der Applikation erstellt, der das JavaScript enthält.



Änderungen an der JavaScript-Datei in diesem Verzeichnis werden beim nächsten Speichern der Applikation überschrieben. Ändern Sie das JavaScript bitte nur im Applikationsdesigner oder kopieren Sie den Inhalt im Applikationsdesigner wieder in den Skripteditor.

Das Verzeichnis  *External/HTMLRoot/Include/Custom* enthält die Datei *custom.js*, die in allen Seiten von Xtreme eingebunden ist. In dieser Datei können Sie individuelle Anpassungen aufnehmen. Die Datei wird nicht durch ein Update überschrieben.

 *is*

Registry Dateien

 *thirdparty*

Dateien von Drittanbietern, deren Applikationen von Intrexx Xtreme verwendet werden. Aktuell liegen hier die Dateien von HTMLAREA, einem freiem HTML-Editor, der für Longtextfelder aktiviert werden kann.

 *tmp*

Temporäre Dateien, die entweder beim Löschen des Portals oder beim Abmelden eines Benutzers entfernt werden (z.B. nicht mehr verwendete Bilder).

 *userfiles*

Werden Dateien über den FCK-Editor hochgeladen, werden diese in diesem Ordner abgelegt. In weiteren Unterordnern werden die verschiedenen Dateitypen abgelegt.

 *WEB-INF*

Konfigurationsdatei für den Tomcat Webserver.

3.2.4. Internal

Das Verzeichnis *Internal* enthält alle Dateien, die (z.B. zur Verarbeitung von Anfragen) direkt im Intrexx Xtreme Server abgearbeitet werden.

 *application*

Alle Dateien, die sich direkt auf Applikationen beziehen (XML, XSL, VM)

 *bpee*

Konfigurationsdateien für *BPEE*-Skripte, die z.B. für Webservice-Calls ausgeführt werden



cfg
Konfigurationsdateien des jeweiligen Portals



files
Uploaddateien in Applikationen



layout
Direkt mit dem Portaldesigner zusammenhängende Dateien (XML, XSL, VM)



lucene
Dateien für die Suchmaschine Lucene



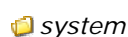
mailroot
Dateien, die per Mail versendet werden



schemas
XML-Schema-Dateien (Definitionsdateien für Intrexx Dokumententypen)



statistics
Log-Dateien für die Statistik



system
Dateien für spezielle Systemaufgaben (z.B. Terminserien, Sprachverwaltung)



tmp
Temporäre Portaldateien



uploadfiles
Temporäres Verzeichnis für hochgeladene Dateien



usrimg
Bilder, die bei den Benutzern hinterlegt sind



webservice
Konfigurationsdateien für registrierte Webservices, -aufrufe und für angebotene Webservices



workflow
Prozessbeschreibungen



wsrp
Konfigurationsdateien für WSRP

3.2.5. Application

Das Verzeichnis *Application* enthält alle Dateien, die direkt mit Applikationen in Zusammenhang stehen.



Template
Unterverzeichnis mit den Namen der vorhandenen Vorlagen. Diese Verzeichnisse enthalten jeweils die Symbole der Applikationen sowie die XML Datei, die diese Applikation beschreibt.



vm
Die Applikationen werden durch XSL Dateien in das Format *Velocity Markup* transformiert. Diese Dateien können durch den Application Server geparkt und ausgeführt werden. Hier werden die Java Businesslogiken aufgerufen.



xml
Dateien zur Validierung der XML Dateien



xsd
Dateien zur Validierung der XML Dateien

📁 *xsl/*

Dateien für die Transformation - in der aktuellen Version werden nur HTML- und Textdateien erzeugt. Die Textdateien dienen dem Mailversand (bei Einstellung *PlainText*). Auch dieses Verzeichnis enthält Dateien, die speziell für Vorlagentypen angelegt sind.

📁 *xsl/common*

XSL-Dateien, die von mehreren Ausgabeformaten benutzt werden.

Unterhalb der Verzeichnisse 📁 *xsl/common* und 📁 *xsl/html/* befindet sich jeweils ein Verzeichnis 📁 *custom*, das nicht durch Updates überschrieben wird. Hier können individuelle Anpassungen abgelegt werden.

3.2.6. Layout

Das Verzeichnis *Layout* enthält alle Dateien, die direkt mit dem Portaldesigner in Verbindung stehen.

📁 *assets*

Bildfamilien, die in Intrexx Xtreme über den Portaldesigner zugeordnet werden können, wie z.B. die im Lieferumfang enthaltene Bildfamilie *Intrexx_Default*. Eigene Bildfamilien können erstellt und in diesem Verzeichnis hinterlegt werden. Beim Veröffentlichen eines Portals wird das Verzeichnis in das Verzeichnis 📁 *External/html/images/assets* kopiert.



Direkte Änderungen an Bilddateien im Verzeichnis *External* werden beim Speichern des aktiven Layouts überschrieben.

📁 *stored*

Alle auf dem Server gespeicherten Layouts

📁 *vm*

Alle Dateien, die durch den Portaldesigner in einer XML/XSL Transformation erzeugt wurden oder als fertige VMs aus diesem Verzeichnis Verwendung finden (z.B. Portlets).

📁 *xml*

Das aktuelle Layout in Form einer XML Datei. Hier befindet sich auch die Datei *Menudesigner.xml*. Sie enthält die Menüstruktur und spezielle Formatierungen einzelner Menüpunkte.

📁 *xsl*

Alle Dateien für die Transformation des Layout-XML in die resultierenden VM Dateien.

3.2.7. system

Dieses Verzeichnis enthält im Wesentlichen statische Velocity Markup Dateien, die jeweils bestimmte Aufgaben übernehmen.

📁 *vm/html/actioncontrol*

Zusatzkontrollen, die im Portaldesigner den einzelnen Frames zugeordnet werden können. Über die Anpassung der XML Datei können eigene Zusatzkontrollen entwickelt werden.

Alle weiteren Verzeichnisse sind selbsterklärend.

3.3. Die Dateien *object.js*, *form.js* und *api.js*

Im Verzeichnis 📁 *External/htmlroot/include* finden Sie die im Folgenden beschriebenen JavaScript-Dateien, die Definitionen und Methoden für die Anpassung von Intrexx Xtreme enthalten.

object.js

Diese Datei enthält das gesamte Fensterhandling von Intrexx Xtreme. Weiterhin sind die Objektdefinitionen der Actioncontrols enthalten.

form.js

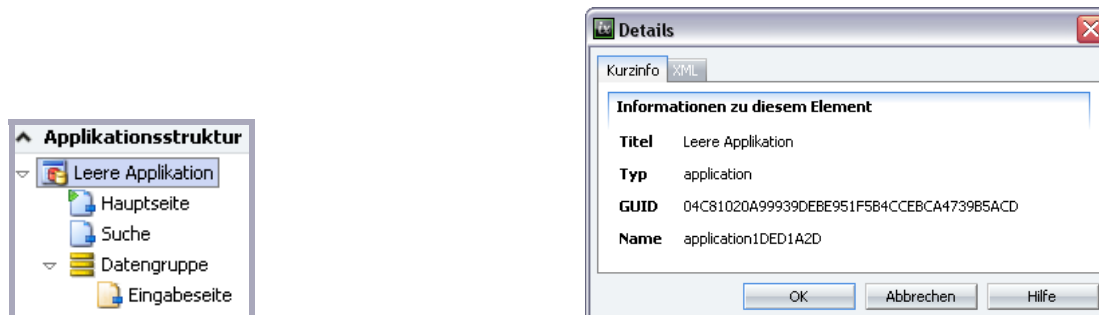
Diese Datei enthält alle Definitionen von Eingabeelementen in Intrexx Xtreme.

api.js

Die Datei *api.js* enthält hilfreiche Methoden für den Umgang mit Werten.

4. Veröffentlichen einer Applikation

Beim Veröffentlichen einer neuen Applikation wird das *_application.xml* als Grundlage der Applikation im Verzeichnis *internal\application\xml\[GUID der Applikation]* hinterlegt. Es enthält die gesamte Applikation in der XML-Notation für den Applikationsdesigner. Die *GUID* der Applikation kann über den Applikationsknoten und die Taste *<F4>* im Applikationsdesigner ermittelt werden. Voraussetzung ist, dass der Expertenmodus im Menü *Extras/Optionen* aktiviert wurde.

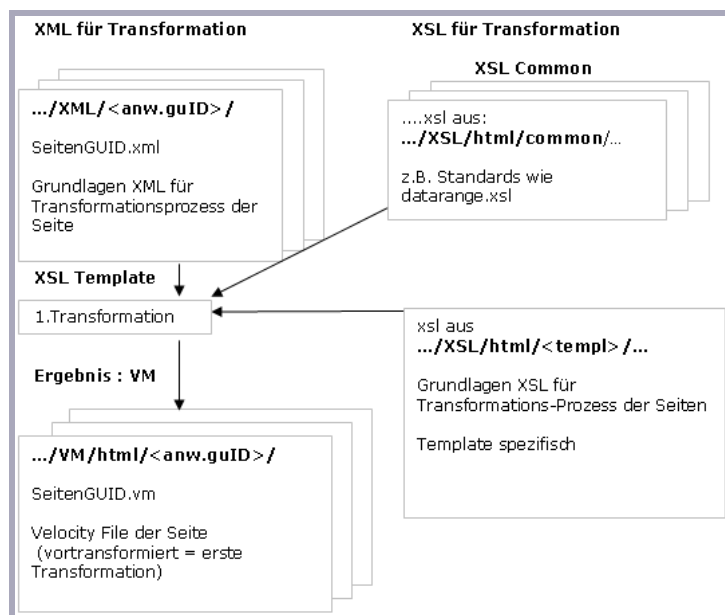


Außerdem wird für jede Seite der Applikation ein Seiten-XML geschrieben und daraus über einen ersten Transformationsprozess zum Einbinden der XSL die Seiten-VMs erzeugt und abgelegt. Für jede Seite der Applikation werden erzeugt:

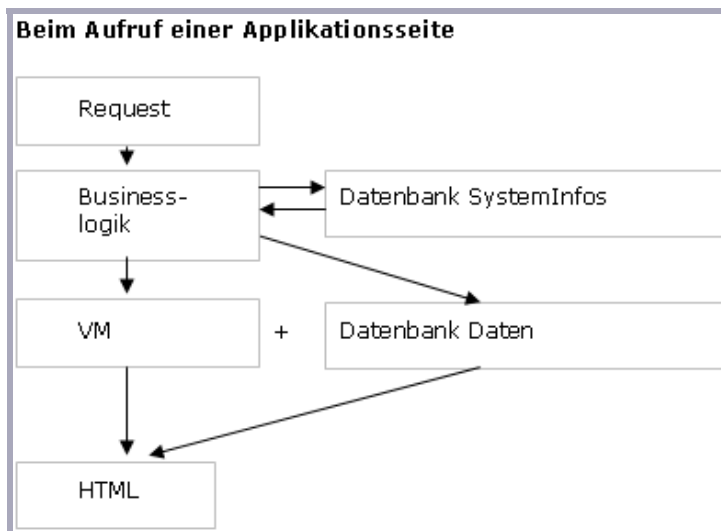
- XML/[GUID der Applikation]/[GUID der Seite].xml
- VM/[GUID der Applikation]/ [GUID der Seite].vm

Alle für die Applikation nötigen Systemdaten werden in der Datenbank eingetragen.

4.1. Aufruf einer Seite (1. Transformation)



4.2. Aufruf einer Seite (2. Transformation)



5. JavaScript

Wo wird JavaScript verwendet?

JavaScript-Funktionen können in statischen VMs verwendet werden oder über XSL in die VM geschrieben werden. Daraus wird die HTML-Seite generiert. JavaScript kann so im Browser aufgerufen und ausgeführt werden. Zum Einbinden eigener JavaScript-Funktionen steht ein JavaScript-Editor im Applikationsdesigner zur Verfügung.

Wie können Parameter aus Velocity in JavaScript verwendet werden?

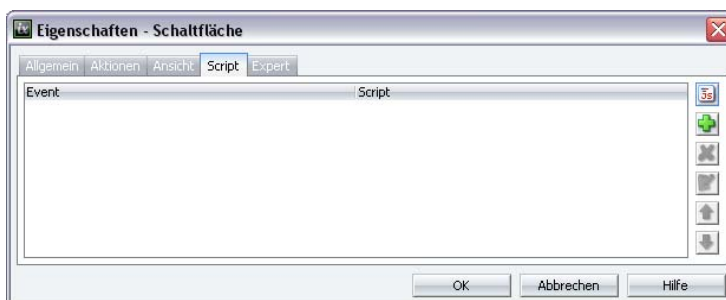
Im JavaScript-Editor können keine direkten Velocityaufrufe hinterlegt werden, da die Funktionen nicht transformiert werden. Ein Velocity-Aufruf kann aber als Parameter an eine JavaScript-Funktion übergeben werden.


JavaScript Objekte

In Intrexx Xtreme sind alle Kontrollen objektorientiert aufgebaut. Zusätzlich zur normalen HTML-Kontrolle existiert ein spezifisches, eigenes United Planet Objekt. Diese Objekte sind definiert in der Datei *object.js*.


Jede HTML-Kontrolle enthält die zusätzliche Methode *oUp*, über die das United Planet Objekt angesprochen werden kann. Die Objekte enthalten zusätzliche Methoden, wie z.B. eine Methode, die aus einer eingegebenen Zahl die Formatierung entfernt und eine Zahl liefert, mit der Berechnungen durchgeführt werden können (z.B. `var f_Float = oHtmlFloat.oUp.toJSNumber(oHtmlFloat.value);`).

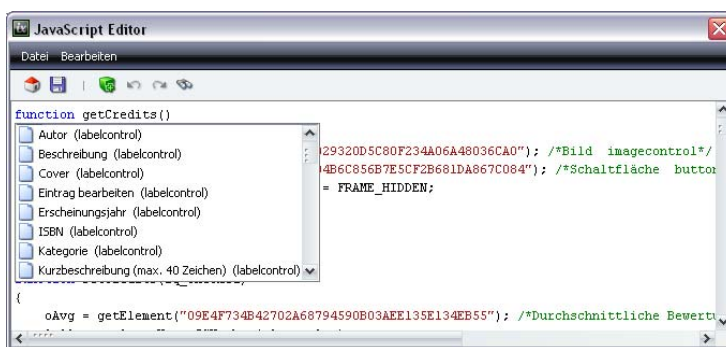
Mit JavaScript wird die Funktionalität von Intrexx Xtreme Applikationen erweitert. Im Eigenschaftendialog von Seiten und Eingabeelementen finden Sie den Reiter *Skript*. Hier können beliebige eventgesteuerte Funktionen verfasst werden, die z.B. Benutzerhinweise ausgeben oder Eingaben validieren.



Mit Klick auf  *Skript* öffnet sich der Skripteditor. Tragen Sie hier Ihre JavaScript-Funktionen ein.



 *Kontrollelement einfügen* oder ein Rechtsklick an der gewünschten Position im Editor öffnet eine Liste, aus der Eingabe- und Ansichtselemente, die sich auf der aktuellen Seite befinden, ausgewählt werden können.



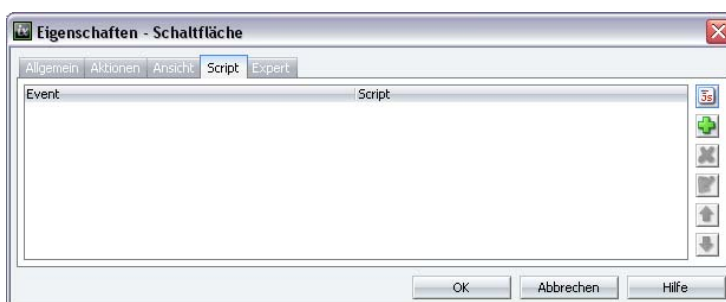
Wählen Sie das Element aus, das im Skript referenziert werden soll. Folgender Programmcode wird im Skripteditor an der aktuellen Cursorposition eingefügt:

```
getElement("GUID des Elements")
```

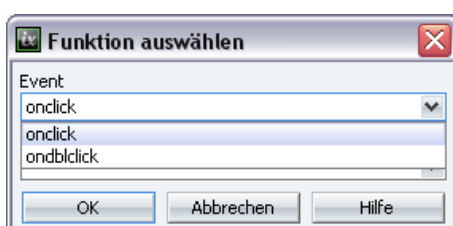
Erstellen Sie Ihre Referenzen auf das HTML-Objekt mit der Syntax

```
var NameDesElements = getElement("GUID des Elements");
```

Klicken Sie nach der Fertigstellung Ihrer Funktion auf  *Speichern des Skripts*.



Über  *Skriptaufruf hinzufügen* kann die Funktion nun dem gewünschten Event zugeordnet werden.



Jedes Element stellt entsprechend seiner Funktion spezifische Events zur Verfügung, eine Schaltfläche z.B. den Event *onclick* und *ondblclick*.

Das Skript wird nach Veröffentlichung der Applikation immer ausgeführt, wenn das Event, dem die Funktion zugeordnet ist, im Browser eintritt.

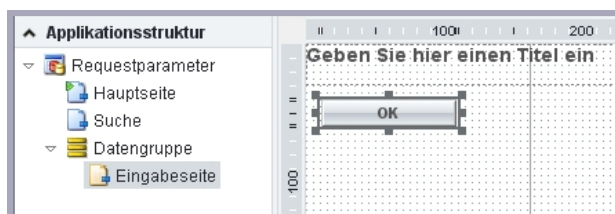
- Die Skriptfunktionen stehen nur in der aktuellen Applikation zur Verfügung. Beim Speichern der Applikation wird der Inhalt des Skripteditorfensters in eine JavaScript Datei extrahiert. Die Datei wird im Installationsverzeichnis von Intrexx Xtreme abgelegt (📁 <xtreme>\org\external\htmlroot\include\[GUID der Applikation].js). Diese Datei wird nicht wieder eingelesen. Direkte Änderungen in dieser Datei haben keine Auswirkung. Ändern Sie Skript bitte nur im Skripteditor.


5.1. Requestparameter

Ein Parameter soll an die nächste Seite übergeben werden. Im folgenden Beispiel wird von einer Schaltfläche ein Requestparameter an die nächste Seite übergeben, und dort mit JavaScript ausgelesen.

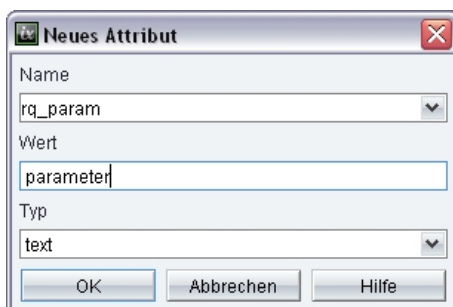
Übung 1

Erstellen Sie die Applikation *Requestparameter* auf Basis der Vorlage *Leere Applikation*. Definieren Sie auf der Eingabeseite eine Schaltfläche mit dem Titel *OK* und Sprungziel auf die Hauptseite der Applikation.



Wechseln Sie auf den Reiter *Expert* und legen Sie mit Klick auf  *Neu* ein neues Attribut an:

Name: rq_param
Wert: parameter
Typ: text




Der Wert wird in der URL an den Server übergeben und im Request-Objekt abgelegt. Alle Expertattribute, die mit *rq_* beginnen, werden als Requestparameter an die folgende Seite übergeben. Bitte beachten Sie hier die Groß- und Kleinschreibung. Auf der Hauptseite wird im *onload*-Event der Seite folgendes Skript aufgerufen:

```
function alertParameter(p_strParam)
{
    // check if set
    if(p_strParam)
        alert(p_strParam);
    return true;
}
```

Tragen Sie die Funktion im onload-Event wie folgt ein:

```
alertParameter('$!Request.get("rq_param")');
```

Definieren Sie die Eingabeseite als Startseite, speichern und testen Sie die Applikation im Browser. Bei Klick auf  OK auf der Eingabeseite wird der Sprung auf die Hauptseite ausgeführt. Der Wert des Requestparameters wird in einer Meldung ausgegeben.



Übung 2

Ein Parameter kann auch variabel übergeben werden, z.B. um den Inhalt eines Eingabe-elementes zu liefern. Im folgenden Beispiel wird von einer Eingabeseite aus ein Wert an die Hauptseite übergeben und dort mit JavaScript ausgelesen.

Erstellen Sie die Applikation *Requestparameter2* auf Basis der Vorlage *Leere Applikation*. Definieren Sie auf der Eingabeseite eine Auswahlliste mit dem Titel *Status*, dem Datentyp *Text* und den beiden benutzerdefinierten Einträgen *Freigabe* und *Bearbeitung*. Legen Sie eine Schaltfläche mit dem Titel *OK* und Sprung auf die Hauptseite an.



Öffnen Sie den Skripteditor über den Eigenschaftendialog der Schaltfläche und fügen Sie das folgende Skript ein:

```
function sendParameter(p_oHtmlButton)
{
    var oHtmlStatus = getElement("552C...FBA"); /*Status dropdowncontrol*/
    var text = "";

    if(oHtmlStatus.value == 'Freigabe')
        text = "Das Dokument wurde freigegeben.";
    else
        text = "Das Dokument wurde noch nicht freigegeben.";

    p_oHtmlButton.oUp.oTarget.addParam =
Helper.setQsValueByParam("rq_param",text,
p_oHtmlButton.oUp.oTarget.addParam);
    return true;
}
```

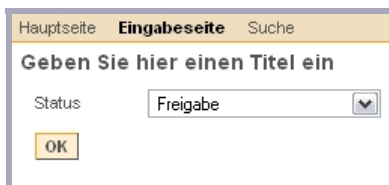
Ordnen Sie die Funktion dem *onclick*-Event der Schaltfläche wie folgt zu:

```
sendParameter(this);
```

Definieren Sie auf der Hauptseite ein Ansichtselement des Typs *statischer Text* und wählen Sie die Option *Nur Standardsprache* (z.B. für Programmierung). Geben Sie den Titel *\$Request.get("rq_param")* ein.



Definieren Sie die Eingabeseite als Startseite. Wenn nun im Browser der Wert *Freigabe* in der Auswahlliste *Status* auf der Eingabeseite ausgewählt wird und ein Klick auf die Schaltfläche *OK* die Hauptseite lädt, wird hier der im Skript hinterlegte Text ausgegeben.

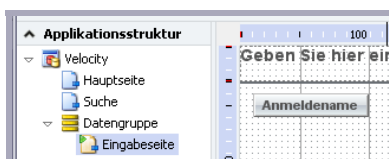


5.2. Velocity

Ein Parameter soll aus dem Velocity Kontext ausgelesen werden und als Request-Parameter an die nächste Seite übergeben werden.

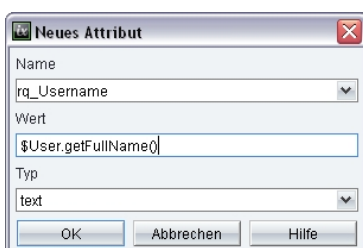
Übung

Erstellen Sie die Applikation *Velocity* auf Basis der Vorlage *Leere Applikation*. Definieren Sie auf der Eingabeseite eine Schaltfläche mit dem Titel *Anmelden* und Sprung auf die Hauptseite.

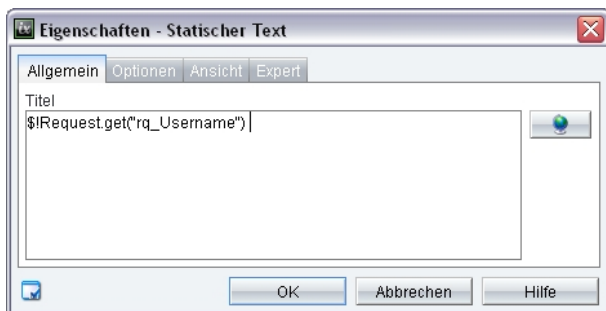


Legen Sie im Eigenschaftendialog der Schaltfläche auf dem Reiter *Expert* ein neues Attribut an:

Name: rq_username
Wert: \$User.getFullName()
Typ: text

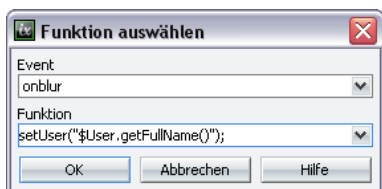


Definieren Sie auf der Hauptseite ein Ansichtselement des Typs *statischer Text* und wählen Sie die Option *Nur Standardsprache* (z.B. für Programmierung). Geben Sie den Titel *\$!Request.get("rq_username")* ein.



i Liefert eine Velocity-Anweisung NULL oder ist undefiniert, so wird der gesamte Ausdruck ausgegeben. Das Ausrufezeichen hinter dem "\$"-Zeichen unterdrückt die Ausgabe der Velocity-Anweisung, wenn z.B. in der Benutzerverwaltung kein voller Name hinterlegt ist. Vergewissern Sie sich also, dass bei dem User mit dem Sie im Browser eingeloggt sind, der Volle Name hinterlegt ist. Falls Sie ihn nachträglich füllen, müssen Sie sich im Browser neu einloggen.

Der Velocity Ausdruck liest aus der Session-Information den aktuellen Anmeldenamen aus und übergibt diesen bei Klick als Requestparameter (Übergabe in der URL) an die aufzurufende Seite. Es ist auch eine direkte Parameterübergabe über den Velocity Kontext möglich:



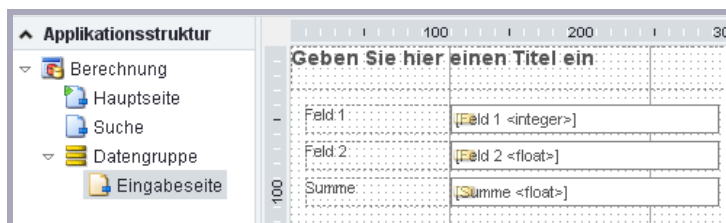
Definieren Sie die Eingabeseite als Startseite und speichern Sie die Applikation ab.

5.3. Berechnungen mit JavaScript

Übung

Erstellen Sie die Applikation *Berechnung* auf Basis der Vorlage *Leere Applikation*. Im folgenden Beispiel wird in einer Eingabeseite eine Berechnung von erfassten Werten durchgeführt und das Ergebnis in ein weiteres Eingabefeld geschrieben. Legen Sie auf der Eingabeseite drei Eingabefelder an:

Titel	Datentyp
Feld 1 Integer	Ganzzahl
Feld 2 Dezimalzahl	Dezimalzahl
Summe	Dezimalzahl



Öffnen Sie den Eigenschaftendialog des Eingabefeldes *Feld 2 Dezimalzahl*, klicken Sie auf den Reiter *Skript* und definieren Sie im Skripteditor die folgende Funktion:

```
function calcValues()
{
    var oHtmlInt = getElement("E8F..8F0"); /*Feld 1 integercontrol*/
    var oHtmlFloat = getElement("D6..8C8"); /*Feld 2 floatcontrol*/

    //Umwandeln in JavaScript Zahlen, Entfernen der Formatierung
    var l_Int = oHtmlInt.oUp.toJSNumber(oHtmlInt.value);
    var f_Float = oHtmlFloat.oUp.toJSNumber(oHtmlFloat.value);

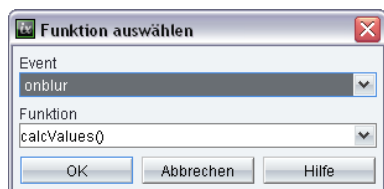
    //Berechnung
    var l_Sum = l_Int + f_Float;

    var oHtmlSum = getElement("49..58"); /*Summe floatcontrol*/

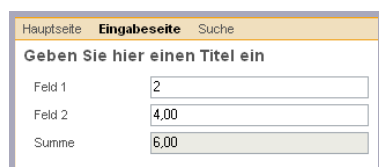
    //Schreiben des Ergebnisses mit Formatierung
    oHtmlSum.value = oHtmlSum.oUp.toLocalFormat(l_Sum);

    return true;
}
```

Diese Funktion wird im *onblur*-Event des zweiten Eingabefeldes aufgerufen.



Speichern Sie die Applikation und öffnen Sie sie im Browser. Auf der Eingabeseite kann nun die Summe aus den beiden Eingabefeldern berechnet werden.



Die *api.js* stellt Funktionen zur Verfügung, die eine kürzere und komfortablere Lösung ermöglichen:

```
function calcValuesApil()
{
    var oHtmlInt = getElement("32C1...E80F"); /*Feld 1 Integer
integercontrol*/
    var oHtmlFloat = getElement("C134...3D66"); /*Feld 2 Dezimalzahl
floatcontrol*/

    //Umwandeln in JavaScript Zahlen, Entfernen der Formatierung
    var l_Int = getNumberObject(oHtmlInt);
    var f_Float = getNumberObject(oHtmlFloat);

    //Berechnung
    var l_Sum = l_Int + f_Float;

    var oHtmlSum = getElement("AF81...DAE3"); /*Summe berechnen
floatcontrol*/

    //Schreiben des Ergebnisses mit Formatierung
    writeLocalString(oHtmlSum, l_Sum);
    return true;
}
```

Auch diese Lösung kann noch verkürzt werden:

```
function calcValuesApi2()
{
    var oHtmlInt    = getElement("32C1...E80F"); /*Feld 1 Integer
integercontrol*/
    var oHtmlFloat = getElement("C134...3D66"); /*Feld 2 Dezimalzahl
floatcontrol*/
    var oHtmlSum = getElement("AF81...DAE3"); /*Summe berechnen
floatcontrol*/

    //berechnen und das Ergebnis schreiben
    calculate(oHtmlInt, oHtmlFloat, "+", oHtmlSum);
    return true;
}
```

5.4. Ausblenden von Kontrollen

Eine Gruppierung auf einer Seite soll verborgen werden.

Übung

Legen Sie auf der Seite eine zusätzliche Schaltfläche mit dem Titel *hide* an. Gruppieren Sie alle Elemente, die versteckt werden sollen, einschließlich der Schaltfläche *hide*. Geben Sie der Gruppierung über den Eigenschaftendialog den Titel *Hidden*.

Initialisieren Sie die *Hidden*-Schaltfläche mit der Funktion *Einblenden/Ausblenden* und geben Sie den zu versteckenden Container an. Der initiale Zustand ist ausgeblendet. Auf diese Weise werden alle Elemente in der Gruppierung *hidden* im Browser nicht angezeigt.



5.5. Einfügen von Text in Ansichtskontrollen

Die aktuelle *Datensatz-Id* soll über Velocity ausgelesen und in einer JavaScript-Methode interpretiert werden. Je nach Wert soll in einem anderen Ansichtsfeld eine Ausgabe erscheinen. Um Werte aus Ansichtskontrollen über JavaScript zu lesen und in Ansichtskontrollen zu schreiben, gibt es komfortable Methoden in der Datei *api.js*:

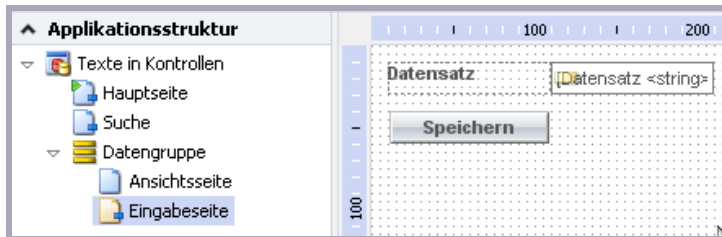
```
/*
DESCRIPTION: liest einen Textknoten aus einem Ansichtsfeld oder statischen
Text
PARAMETERS: oHtml: Referenz auf die Html-Eingabekontrolle
RETURN:      Wert
BEISPIEL:   getTextValue(getElement("F28BA735...A1433"));
*/
function getTextValue(oHtml)
{
    return Browser.getValue(oHtml);
}

//Schreiben
/*
DESCRIPTION: schreibt einen Wert in ein Ansichtsfeld oder statischen Text
PARAMETERS: oHtml: Referenz auf die Html-Eingabekontrolle
strValue:   Wert
RETURN:     true/false
*/
```

```

BEISPIEL:      setValue(getElement("F28BA...66A1433"), " myString");
*/
function setValue(oHtml, strValue)
{
    return Browser.setValue(oHtml, strValue);
}
    
```

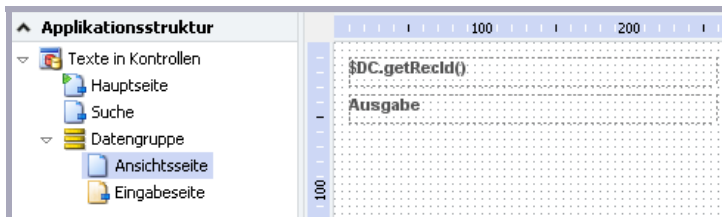
Erstellen Sie die Applikation *Texte in Kontrollen* auf Basis der Vorlage *Leere Applikation*. Definieren Sie auf der Eingabeseite ein Eingabefeld mit dem Titel *Datensatz* (*Datentyp: Text*) und eine Schaltfläche *Speichern* mit Sprung auf die Hauptseite.



Definieren Sie dann eine *Ansichtsseite* in der *Datengruppe*. Legen Sie auf der Seite ein *Ansichtselement* des Typs *statischer Text* an. Tragen Sie als Titel des Feldes den Velocity Befehl zum Auslesen der aktuellen *Datensatz-Id* ein:

```
$DC.getRecId()
```

Wählen Sie im *Eigenschaftendialog* (Reiter *Optionen*) die Option *nur Standardsprache (für Programmierung)*. Legen Sie ein zweites *Ansichtselement* des Typs *statischer Text* an und betiteln Sie es mit *Ausgabe*.



Schreiben Sie die folgende JavaScript-Funktion:

```

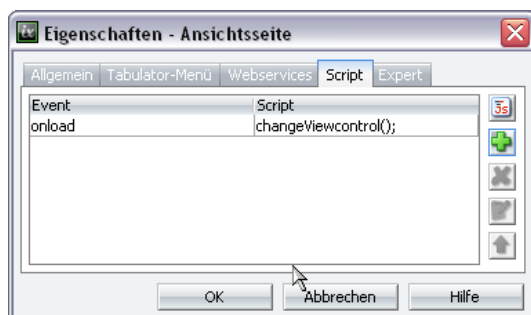
function changeViewcontrol()
{
    var oHtmlRecordid = getElement("C..E"); /*$DC.getRecId()
labelcontrol*/
    var oHtmlAusgabe = getElement("C2...DB5"); /*Ausgabe labelcontrol*/

    //Auslesen der aktuellen Datensatz-Id
    var recordid = getTextValue(oHtmlRecordid);

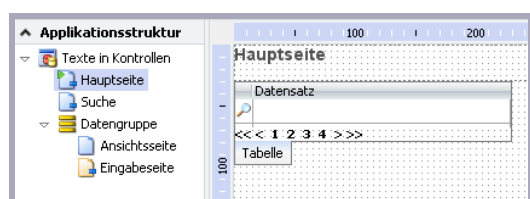
    // Interpretieren und Verfassen des Ausgabetextes
    var text = "";
    if(recordid == 1)
        text = "Dies ist der erste Datensatz";
    else
        text = "Dies ist ein weiterer Datensatz";

    //Setzen des Textes
    setValue(oHtmlAusgabe, text);
    return true;
}
    
```

Rufen Sie die Funktion im *onload*-Event der *Ansichtsseite* auf.



Legen Sie nun auf der Hauptseite eine Ansichtstabelle für die *Datengruppe* mit der Spalte *Datensatz* und Sprung auf die Ansichtseite im *Tooltip* an.



Speichern Sie die Applikation. Im Browser können nun auf der Eingabeseite Datensätze angelegt werden. Werden diese über die Ansichtstabelle geladen, so wird der im Skript festgelegte Text ausgegeben.



5.5.1. Typdefinierte Ansichtselemente

Sollen Berechnungen mit Werten in Ansichtselementen durchgeführt werden, kann der Wert des Ansichtselementes ausgelesen und über selbst definierte JavaScript-Methoden in einen Datentyp für Berechnungen konvertiert werden. Ab der Version 2.5 werden diese Konvertierungsmethoden mit dem Expertattribut

Name: jsubject
Wert: true
Typ: boolean

bereitgestellt. Das Expertattribut baut eine bidirektionale Beziehung zwischen dem HTML-Objekt der Ansichtskontrolle und dem zugehörigen Up-Objekt für die Verwendung in JavaScript auf. Bei Eingabekontrollen ist die bidirektionale Beziehung Standard. So können z.B. mit dem Wert eines Datums-Ansichtsfeldes Berechnungen durchgeführt werden, wobei die kontrollspezifischen Formatierungen berücksichtigt werden. Das Expertattribut kann bei folgenden Kontrolltypen eingesetzt werden:

- upIntegerVControl
- upFloatVControl
- upCurrencyVControl
- upDateTimeVControl
- upDateVControl
- upTimeVControl
- upCheckVControl


Der Wert der Ansichtskontrollen lässt sich direkt über die Funktion

```
getElement("GUID").oUp.displayValue
```

abfragen. Beim Ansichtselement *upCheckVControl* kann der Wert über die Funktion

```
getElement("GUID").oUp.checked
```

abgefragt werden. Der Rückgabewert ist *true* oder *false*.

 Eine Abfrage über *Browser.getValue()* ist nur noch eingeschränkt möglich, da im Rückgabewert nicht nur der Inhalt des (sichtbaren) Ansichtsfeldes enthalten ist, sondern auch der (im Browser unsichtbare) JavaScript-Block. Die neuen Methoden *displayValue* und *checked* stehen aus deshalb nur in diesem Zusammenhang für die Ansichtskontrollen zur Verfügung.

5.6. Ein- und Ausblenden von Elementen

Abhängig von der Auswahl eines Wertes sollen Kontrollelemente ein- bzw. ausgeblendet werden. Auf einer Eingabeseite wird über eine Auswahlliste eine Eingabe vorgenommen. Abhängig davon, welcher Wert gewählt wird, sollen unterschiedliche Zusatzkontrollen angezeigt werden.

Übung

Erstellen Sie die Applikation *Einblenden / Ausblenden* auf Basis der Vorlage *Leere Applikation*. Definieren Sie auf der Eingabeseite eine Auswahlliste mit dem Titel *Status* und den benutzerdefinierten Werten *Freigabe* und *Bearbeitung*.

Legen Sie zwei weitere Eingabefelder an: *Bearbeiten bis* mit Datentyp *Datum* und *Freigegeben* mit Datentyp *Text*. Gruppieren Sie jedes der Eingabefelder mit seinem Titel. Geben Sie der ersten Gruppierung über den Eigenschaftendialog den Titel *Bearbeitung*, der zweiten den Titel *Freigabe*.



Definieren Sie folgende Funktion für das *onchange*-Event der Auswahlliste:

```
function showControls()
{
    var oHtmlStatus = getElement("51E2...6A02"); /*Status
dropdowncontrol*/
    var oHtmlFreigabe = getElement("FA2F...0E12"); /*Freigabe
simplegroup*/
    var oHtmlBearbeiten = getElement("8A90...92AB"); /*Bearbeiten
simplegroup*/

    if (oHtmlStatus.value == 'Freigabe')
    {
        //Einblenden Freigabe
        oHtmlFreigabe.style.visibility = "visible";
        oHtmlFreigabe.style.display = "block";

        //Ausblenden Bearbeiten
        oHtmlBearbeiten.style.visibility = "hidden";
        oHtmlBearbeiten.style.display = "none";
        return true;
    }
    if (oHtmlStatus.value == 'Bearbeitung')
    {
```

```

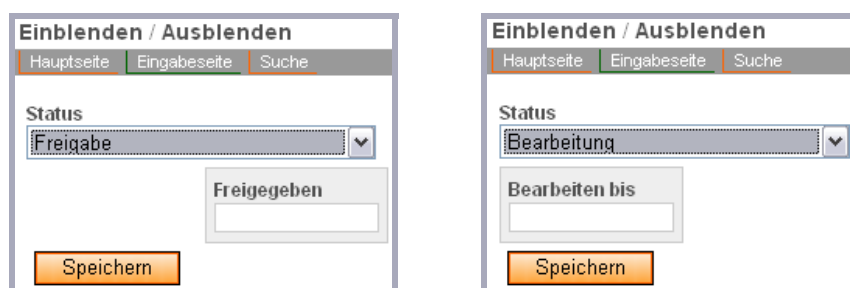
        //Ausblenden Freigabe
        oHtmlFreigabe.style.visibility = "hidden";
        oHtmlFreigabe.style.display = "none";

        //Einblenden Bearbeiten
        oHtmlBearbeiten.style.visibility = "visible";
        oHtmlBearbeiten.style.display = "block";
        return true;
    }

    //Ausblenden Bearbeiten
    oHtmlBearbeiten.style.visibility = "hidden";
    oHtmlBearbeiten.style.display = "none";

    //Ausblenden Freigabe
    oHtmlFreigabe.style.visibility = "hidden";
    oHtmlFreigabe.style.display = "none";
    return true;
}
    
```

Bei Auswahl eines Eintrags in der Auswahlliste im Browser wird die entsprechende Gruppierung ausgeblendet.



- ! Das *onchange*-Event wird nur bei Auswahl in einer Auswahlliste ausgelöst. Definieren Sie die Auswahlliste im Eigenschaftendialog (Reiter *Allgemein*) als Pflichtfeld (*Eingabe erforderlich*), so muss eine Auswahl getroffen werden. Damit ist sichergestellt, dass das *onchange*-Event auf jeden Fall ausgelöst wird.

Damit beim Laden der Seite die beiden zusätzlichen Eingabefelder nicht sichtbar sind oder in Abhängigkeit zum eingestellten Wert angezeigt werden, sollte die Funktion auch beim *onload*-Event der Seite aufgerufen werden. Mit dem gleichen Verfahren können auch beliebige Kontrollen ein- und ausgeblendet werden.

Das automatische Ein- und Ausblenden kann auch mit Hilfe einer Schaltfläche gelöst werden, deren Aktion im Eigenschaftendialog auf *Ein-/Ausblenden* eingestellt ist. Der Vorteil ist, dass bei langsamen Browsern die zu versteckenden Elemente beim Laden der Seite nicht kurz angezeigt werden, sondern von vornherein unsichtbar sind.

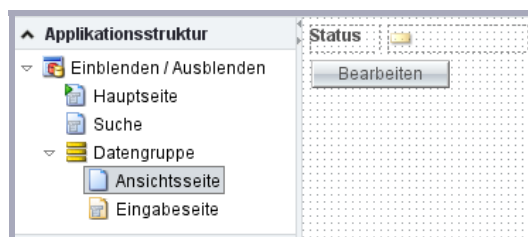
- ! Das Ein- und Ausblenden von Container kann über folgende Funktion mit JavaScript gesteuert werden: `oButton.oUp.flipFlop(true)`; . Ist der Wert *true*, wird der Container eingeblendet, ist der Wert *false*, wird er ausgeblendet.

5.7. Ein- und Ausblenden beim Laden einer Seite

Kontrollelemente können alternativ beim Laden der Seite (und nur hier) im Expertmodus aus- bzw. eingeblendet werden. Abhängig von einem Wert in einem Ansichtsfeld soll eine Schaltfläche beim Laden der Seite ein- bzw. ausgeblendet werden.

Übung

Definieren Sie in der Applikation *Einblenden / Ausblenden* eine Ansichtssseite. Legen Sie hier ein Ansichtsfeld an und verbinden Sie es mit dem Datenfeld *Status*. Legen Sie außerdem eine Schaltfläche *Bearbeiten* an.



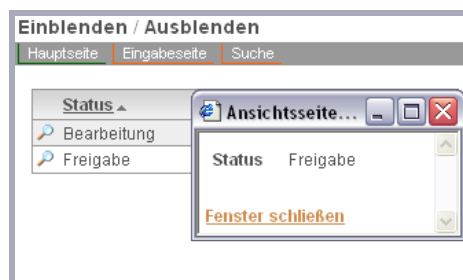
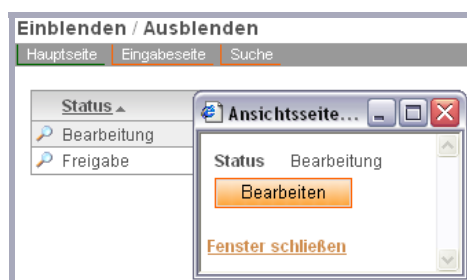
Öffnen Sie den Eigenschaftendialog der Schaltfläche. Wechseln Sie dort auf den Reiter *Expert*. Legen Sie die folgenden neuen Expertattribute an:

Attribut additionalcheck
Wert \$DC.getAttribute('textviewcontrol123456')
Typ Text

Attribut valueadditionalcheck
Wert der Vergleichswert - in unserem Beispiel *Bearbeitung*.
Typ Text

Ersetzen Sie im Wert des ersten Eintrages *textviewcontrol123456* mit dem tatsächlichen Namen des Ansichtsfelds. Dieser kann mit der Taste *F4* oder alternativ im Eigenschaftendialog auf dem Reiter *Expert* ermittelt werden. Definieren Sie auf der Hauptseite der Applikation eine Ansichtstabelle mit Sprung auf die eben erstellte Ansichtsseite im *Tooltip*.

Geben Sie über die Eingabeseite einen Eintrag mit Status *Freigabe* und einen Eintrag mit Status *Bearbeitung* ein. Wird nun ein Datensatz mit Status *Bearbeitung* ausgewählt, so wird die Schaltfläche eingblendet. Beim Status *Freigabe* wird die Schaltfläche nicht angezeigt.



5.8. Kopieren eines Datensatzes mit JavaScript

Ein bestehender Datensatz soll innerhalb der gleichen Datengruppe kopiert werden.

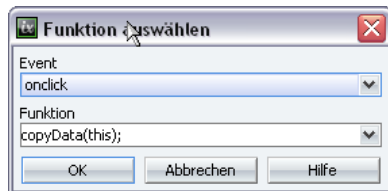
Übung

Erstellen Sie die Applikation *Datensatz kopieren* auf Basis einer leeren Applikation. Legen Sie auf der Eingabeseite ein Eingabefeld mit dem Titel *Datensatz* an. Definieren Sie eine Schaltfläche zum Speichern des Datensatzes mit Sprung auf die Hauptseite und den Einstellungen *Popup/Tooltip schließen* und *Öffnendes Fenster neu laden*. Kopieren Sie diese Schaltfläche und geben Sie der Kopie den Titel *Kopieren*.



Ordnen Sie dem *onclick*-Event der Schaltfläche *Kopieren* die Funktion *copyData(this)* zu und schreiben Sie das folgende Skript:

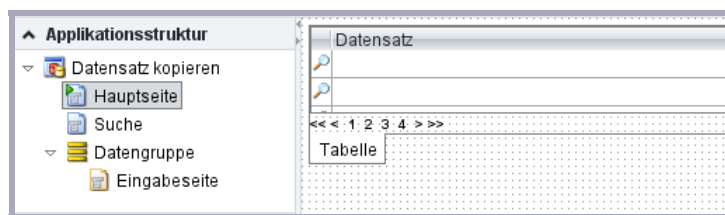
```
function copyData(p_oButton)
{
    var oFup = p_oButton.oÜp.oFup;
    oFup.recId = "-1";
    return true;
}
```



Wird nun im Browser ein bestehender Datensatz geladen, so ist die Datensatz-Id bereits definiert. Bei der Referenz der Schaltfläche auf das Formular wird als Variable *recId* hinterlegt. Durch das Setzen der Datensatz-Id auf den Wert *-1* wird beim Speichern ein Insert-Vorgang ausgelöst.

i Durch Setzen der *RecId* auf einen bestimmten Wert kann ein bestimmter Datensatz aufgerufen werden.

Legen Sie auf der Hauptseite der Applikation eine Ansichtstabelle für das Feld *Datensatz* mit Sprung auf die Eingabeseite im Tooltip an.



Speichern Sie die Applikation. Um das Ergebnis zu testen, geben Sie beliebige Sätze über die Eingabeseite ein und speichern sie. Wird ein bestehender Datensatz über die Ansichtstabelle ausgewählt, so kann dieser mit Klick auf die Schaltfläche *Kopieren* dupliziert werden.



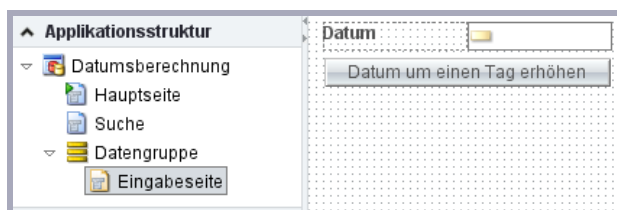
i Bitte beachten Sie, dass bei dieser Vorgehensweise weder Datensätze aus untergeordneten Datengruppen noch Dateien kopiert werden.

5.9. Datumsberechnung mit JavaScript Teil I

Ein vorhandenes Datum soll mit Klick auf eine Schaltfläche um einen Tag erhöht werden.

Übung

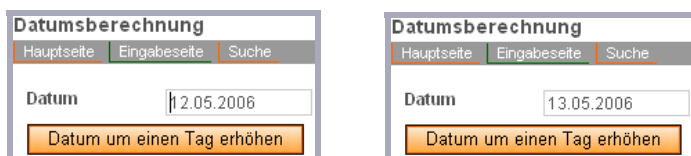
Erstellen Sie die Applikation *Datumsberechnung* auf Basis der Vorlage *Leere Applikation*. Definieren Sie auf der Eingabeseite ein Eingabefeld mit dem Titel *Datum* und dem Datentyp *Datum*. Öffnen Sie den Eigenschaftendialog des Eingabefeldes und stellen Sie auf dem Reiter *Optionen* die Vorgabe *aktuelles Datum und Uhrzeit* ein. Legen Sie dann eine Schaltfläche mit dem Titel *Datum um einen Tag erhöhen* an.



Fügen Sie für das *onclick*-Event der Schaltfläche das folgende Skript ein:

```
function tomorrow()
{
    var oHtmlDate = getElement("00B8...77B1"); /*Datum datecontrol*/
    var dtOldDate = oHtmlDate.oUp.setDateObjectFromLocal(oHtmlDate.value);
    var dtNewDate = dateAdd("d", 1, dtOldDate); // + 1 Tag zum aktuellen
Datum
    oHtmlDate.value = oHtmlDate.oUp.toLocalDateString(dtNewDate);
    return true;
}
```

Mit der Variablen *oHtmlDate* wird eine Referenz auf das Eingabefeld *Datum* gebildet. Das Datum wird mit der Methode *setDateObjectFromLocal()* in ein JavaScript-Date-Objekt umgewandelt und der Variablen *dtOldDate* zugewiesen. Mit der Methode *dateAdd()* wird das Datum um einen Tag erhöht und der Variablen *dtNewDate* zugewiesen. Dies reicht jedoch nicht aus, um den Wert im Formular anzuzeigen. Die Formatierung muss angepasst werden. Dazu bedienen wir uns der Methode *toLocalDateString()*, die wir dem Wert von *oHtmlDate* wieder zuweisen. Bei Klick auf die Schaltfläche im Browser wird das eingegebene Datum um einen Tag erhöht.



5.9.1. Syntax der Methode *dateAdd()*

dateAdd(Interval, Number, Date)

Interval (Typ String)

Mögliche Werte:

"ms" für Millisekunden
 "s" für Sekunden
 "mi" für Minuten
 "h" für Stunden
 "d" für Tage
 "mo" für Monate
 "y" für Jahre

Number (Typ Integer)

Anzahl der in *Interval* vorgegebenen Einheit, die dem Datum hinzugefügt werden soll. Soll das Intervall subtrahiert werden, wird die Anzahl als negativer Wert angegeben (z.B. -1).

Date (Typ Date)

JavaScript Datumsobjekt, das zur Grundlage der Berechnung genommen werden soll.

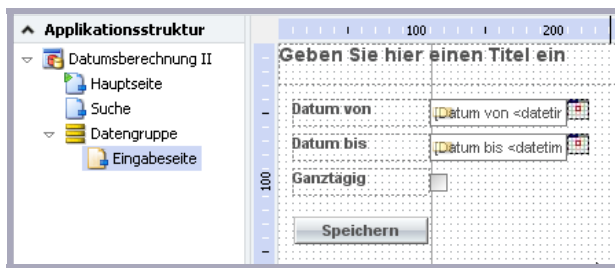
5.10. Datumsberechnung mit JavaScript Teil II

Ein Termin soll durch Setzen einer Checkbox als „ganztägig“ markiert werden.

Übung

Erstellen Sie die Applikation *Datumsberechnung II* auf Basis der Vorlage *Leere Applikation*. Definieren Sie auf der Eingabeseite ein Eingabefeld mit dem Titel *Datum von* und dem Datentyp *Datum & Uhrzeit*. Wiederholen Sie diese Vorgehensweise und legen Sie ein

weiteres Feld *Datum bis* vom Datentyp *Datum & Uhrzeit* an. Nun benötigen wir noch ein *Kontrollkästchen* mit der Bezeichnung *ganztägig*. Legen Sie dann eine Schaltfläche mit dem Titel *Speichern*, Sprung auf die *Hauptseite*, *Popup/Tooltip schließen* und *öffnendes Fenster neu laden* an.



Fügen Sie für das *onclick*-Event der *Checkbox* das folgende Skript ein:

```
function myFullDay()
{
    var oHtmlFrom = getElement("07...63"); /*Datum von datetimecontrol*/
    var oHtmlTo = getElement("67...73"); /*Datum bis datetimecontrol*/
    var oHtmlCheck = getElement("AC...C1"); /*Ganztägig checkcontrol*/

    return setFullDays(oHtmlCheck, oHtmlFrom, oHtmlTo);
}
```

Mit der Variablen *oHtmlDateFrom* wird eine Referenz auf das Eingabefeld *Datum von* gebildet. Mit der Variablen *oHtmlDateTo* eine Referenz auf das Eingabefeld *Datum bis* gebildet. Die Variable *oHtmlcheck* stellt eine Referenz auf das Kontrollkästchen dar.

5.10.1. Syntax der Methode `setFullDays()`

```
setFullDays(checkbox, Element1, Element2)
```

5.11. Umgang mit Tabellen über ein JavaScript Objekt

Mit dem Expertattribut *jsobject* kann auf Werte von Spalten und Reihen in Tabellen zugegriffen werden. Wird das Expertattribut

Name: jsobject
Wert: true
Typ: boolean

bei einer Ansichtstabelle oder frei gestalteten Tabelle eingesetzt, so wird automatisch ein *UpTable*-Objekt erzeugt. Das *UpTable*-Objekt kann über das globale Objekt *oTableReg* erreicht werden. Neben Eigenschaften wie *ID* und *GUID* der Tabelle enthält es

- eine Liste der *RecIds* der auf der Browserseite angezeigten Datensätze
- eine Liste von den Spalten der Tabelle

Ein Spaltenobjekt enthält Properties wie

- *ID*
- *GUID*
- *displayName*
- *controlType*

und bietet Methoden für den Zugriff auf Nodes (*<a>* oder ** Tags) oder auf Inhalte der Textnodes. hier ein Beispiel für den Zugriff auf das Objekt *oTableReg*:

```
oTableReg.getTableByGuid("GUID der Tabelle").getColByGuid("GUID der Spalte").getNodeValue(recid)
```

 Bei der freien Tabelle wird hier die *RecordID* des Datensatzes übergeben, bei Ansichtstabellen der *VelocityCounter*. Diese Methode liefert den Wert aus der Spalte für

die übergebene Spalte:

```
oTableReg.getTableByGuid("GUID der Tabelle").aRecIds
```

Hier wird ein Array der RecordIDs der Seite gebildet (new Array(1,2,5)).

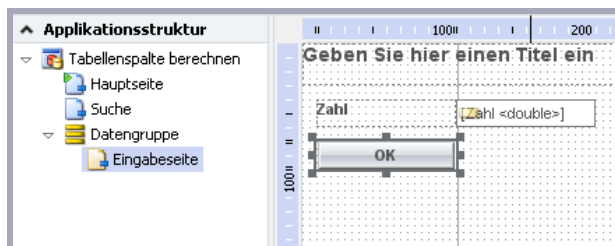
Weitere Methoden sind in der Datei *object.js* enthalten.

5.11.1. Tabellenspalte berechnen

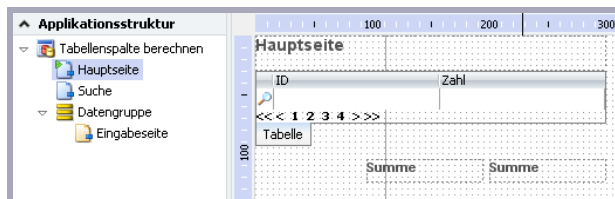
Die Werte eines Dezimalzahlenfeldes in einer Tabellenspalte sollen summiert und in einem Summenfeld ausgegeben werden.

Übung

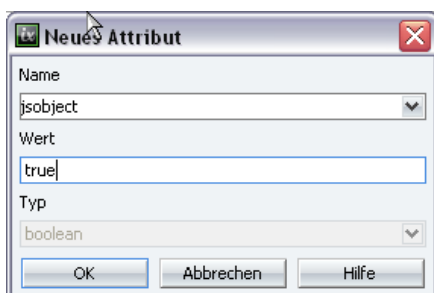
Erstellen Sie die Applikation *Tabellenspalte berechnen* auf Basis der Vorlage *Leere Applikation*. Definieren Sie auf der Eingabeseite ein Eingabefeld des Typs *Dezimalzahl* mit dem Titel *Zahl* und eine Schaltfläche *Speichern* mit der Aktion *Speichern* und Sprung auf die Hauptseite der Applikation.



Auf der Hauptseite erstellen Sie eine Ansichtstabelle mit den Spalten (PK) (S) *Id* und *Zahl*. Unterhalb der Tabelle definieren Sie ein Ansichtsfeld des Typs *Statischer Text* mit dem Titel *Summe*.



Öffnen Sie den Eigenschaftendialog der Tabelle und setzen Sie hier das neue Expertattribut:



Im *onload*-Event der Hauptseite wird die folgende Funktion aufgerufen:

```
function calcTable()
{
    //Referenz auf das UP-Tabellenobjekt
    var oTable = oTableReg.getTableByGuid("1BB6...01C3");
    //Referenz auf das UP Spaltenobjekt
    var oColumn = oTable.getColByGuid("E8DB...F9B3");

    var summe = 0;

    //Summe aller Spalten bilden
    for (var i=1; i <= oTable.aRecIds.length; i++)
```

```

    {
        var oValue = oColumn.getNodeValue(i);
        //Entfernen der Formatierung
        var jsValue = Helper.parseFloat(oValue);
        //Enthält jsValue einen Wert???
        if ( isNaN(jsValue) == false)
            summe = summe + jsValue;
    }

    var oHtml = getElement("FBEC...3ECA"); /*Summe labelcontrol*/
    var oFloat = new upFloatControl();


    //Ausgabe formatieren und ausgeben
    var ausgabe = oFloat.toLocalFormat(summe);
    setValue(oHtml, ausgabe);
    return true;
}

```

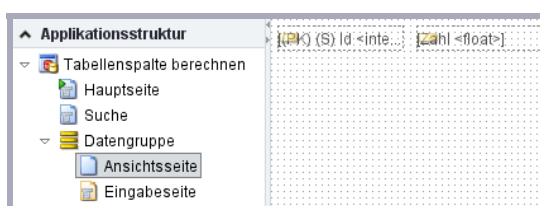
Die GUID der Tabellenspalte kann über den Eigenschaftendialog der Ansichtstabelle mit Klick auf die Schaltfläche *Format* bei markiertem Feld *Zahl* ermittelt werden. Wechseln Sie dann auf den Reiter *Expert* und kopieren Sie die GUID der Spalte.

Geben Sie im Browser auf der Eingabeseite eine beliebige Anzahl von Zahlenwerten ein. Bei Aufruf der Hauptseite wird die Summe der sichtbaren Werte im Summenfeld ausgegeben.

Id	Zahl
2	1,00
4	2,00
3	5,00
1	6,00
Summe	14,00

 Es werden nur die auf der Seite angezeigten Werte der Spalte *Zahl* berechnet. Enthält die Tabelle weitere Seiten, die über das Navigationselement am Fuß der Tabelle erreichbar sind, so fließen diese aktuell nicht angezeigten Werte nicht in die Berechnung ein.

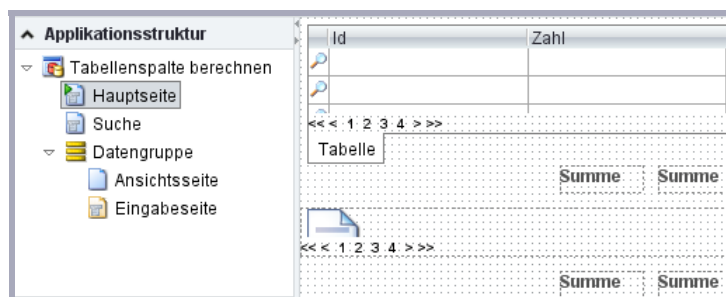
Um die Summe der Spalte einer frei gestalteten Tabelle zu berechnen, definieren Sie eine Ansichtsseite mit Ansichtsfeldern für die Datenfelder *Zahl* und *(PK) (S) Id*.



Legen Sie auf der Hauptseite eine frei gestaltete Tabelle für die neu erstellte Ansichtsseite unterhalb der Ansichtstabelle an. Definieren Sie auch hier das neue Expertattribut

Name: jsubject
Wert: true
Typ: boolean

Für die Ausgabe der Summe wird wieder ein Ansichtsfeld des Typs *Statischer Text* mit dem Titel *Summe* eingesetzt.



Für die Berechnung der Summe in einer frei gestalteten Tabelle wird das Skript im *onload*-Event der Hauptseite wie folgt erweitert:

```
function calcTable()
{
    // Summe für freie Tabelle berechnen

    var oTable = oTableReg.getTableByGuid("CC2B....CDC1");
    var oColumn = oTable.getColByGuid("E550....2508");
    var oRecids = oTable.aRecIds;
    var summe = 0;

    for (i in oRecids)
    {
        var recid = oRecids[i];
        var oValue = oColumn.getNodeValue(recid);
        var jsValue = Helper.parseFloat(oValue);

        if ( isNaN(jsValue) == false)
            summe = summe + jsValue;
    }
    var oHtml = getElement("6DCC....D0A8"); /*Summe labelcontrol*/
    var oFloat = new upFloatControl();
    var ausgabe = oFloat.toLocalFormat(summe);
    setTextValue(oHtml, ausgabe);
    return true;
}
```

Die GUID der frei gestalteten Tabelle kann in der Applikationsstruktur ermittelt werden. Blenden Sie die Elemente der Seite ein, auf der sich die Tabelle befindet und öffnen Sie den untergeordneten Baum, bis Sie auf das Element *shapedtablebase* treffen.



Bei eingeschalteten Expertenoptionen können Sie nun mit der Taste F4 die korrekte GUID, die im Script für das Auslesen von Spaltenwerten verwendet werden muss, ermitteln. Als GUIDs der einzelnen Spalten setzen Sie die jeweiligen GUIDs der Ansichtselemente auf der Ansichtseite ein. Speichern Sie die Applikation im Browser und testen Sie.

Tabellenspalte berechnen		
Hauptseite Eingabeseite Suche		
Id	Zahl ▲	
2	1,00	
4	2,00	
3	5,00	
1	6,00	
Summe		14,00

Id	Zahl	
2	1,00	
4	2,00	
3	5,00	
1	6,00	
Summe		14,00

Über folgende Befehle können auch Tabellenzellen für die Ausgabe überschrieben werden:

```
for (i in oRecids)
{
    var oNode = oColumn.getNode(i);
    setTextValue( oNode, "value");
}
```

5.12. Automatischer Reload einer Ansichtstabelle

Mit der Funktion `reload_[tablerecords]();` soll eine Ansichtstabelle in regelmäßigen Abständen aktualisiert werden. Ordnen Sie die Funktion dem `onLoad`-Event der Seite zu, auf der sich die zu aktualisierende Ansichtstabelle befindet.

```
window.setTimeout('reload_tablerecords12...()',200); // Ausführen der reload-
Funktion in 200 Millisekunden-Abstand...
```

Ersetzen Sie `tablerecords12...` mit dem tatsächlichen Namen der `Tablerecords` in ihrer Applikation. Der Name kann ermittelt werden, indem Sie

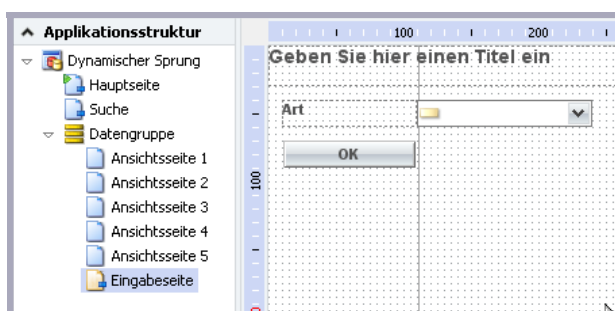
- die Ansichtstabelle markieren,
- mit der Tast `F4` (Expert-Modus muss aktiviert sein)
- in den XML-Dialog wechseln,
- hier das Tag `<tablerecords` und das Attribut `name=""` suchen, diesen Wert markieren und kopieren.

5.13. Dynamischer Sprung aus Ansichtstabelle

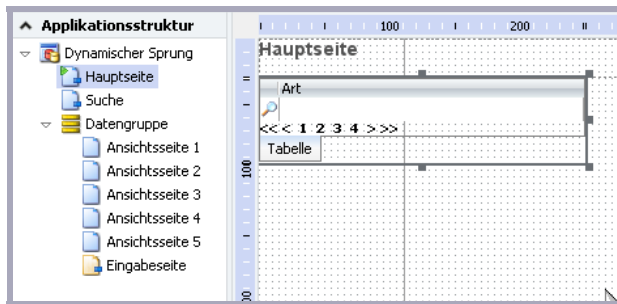
Aus einer Spalte einer Ansichtstabelle soll in Abhängigkeit eines Spalten-Wertes auf unterschiedliche Ansichtsseiten gesprungen werden.

Übung

Erstellen Sie die Applikation *Dynamischer Sprung* auf Basis der Vorlage *Leere Applikation*. Definieren Sie auf der Eingabeseite eine Auswahlliste mit dem Titel *Art*, dem Datentyp *Text* und den benutzerdefinierten Einträgen *Art 1*, *Art 2*, *Art 3...* - *Art 5* und eine Schaltfläche mit der Aktion *Speichern*, dem Titel *OK* und Sprung auf die Hauptseite an. Legen Sie unterhalb der Datengruppe 5 Ansichtsseiten für die unterschiedlichen Arten an und benennen sie diese entsprechend.



Erstellen Sie nun auf der Hauptseite eine Ansichtstabelle mit den Spalten *Art* und *Sprungziel* auf eine der *Ansichtsseiten* im Tooltip.



Öffnen Sie den Skripteditor über das Menü *Bearbeiten / JavaScript editieren* und fügen Sie das folgende Skript ein:

```
function getArtandJump(oHtml, p_strArt)
{
    if(p_strArt == "Art 1")
        oHtml.oUp.oTarget.rq_TargetGuid = "7069A...0A6A9";
    else if(p_strArt == "Art 2")
        oHtml.oUp.oTarget.rq_TargetGuid = "85D3B...37511";
    else if(p_strArt == "Art 3")
        oHtml.oUp.oTarget.rq_TargetGuid = "4BEF...E52FD1";
    else if(p_strArt == "Art 4")
        oHtml.oUp.oTarget.rq_TargetGuid = "7ED6...E8255";
    else if(p_strArt == "Art 5")
        oHtml.oUp.oTarget.rq_TargetGuid = "B473A...110D4";

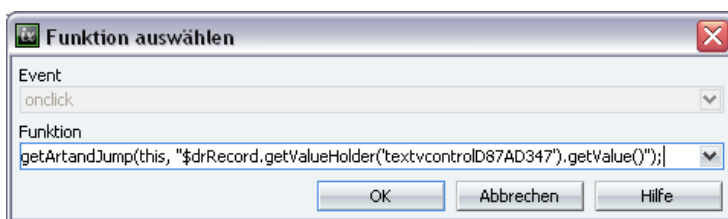
    return true;
}
```

! Bitte beachten Sie, dass die zu prüfenden Werte in der JavaScript-Funktion gleich den Werten der Auswahlliste *Art* sein müssen.

Ersetzen Sie die einzelnen GUIDs mit dem tatsächlichen GUIDs der *Ansichtsseiten 1-5*. Diese können im Detailsdialog der Seiten ermittelt werden.

Ordnen Sie die Funktion dem *onclick*-Event der Spalte, die als Sprung definiert ist, wie folgt zu:

```
getArtandJump(this, "$drRecord.getValueHolder('textvcontro...').getValue()");
```




Ersetzen Sie im Wert des ersten Eintrages *textvcontrolD87AD347* mit dem tatsächlichen Namen der Spalte *Art*. Dieser kann im Eigenschaftendialog der Spalte auf dem Reiter *Expert* ermittelt werden.

Speichern Sie die Applikation. Um das Ergebnis zu testen, geben Sie 5 Arten über die Eingabeseite ein und speichern sie. Wird ein bestehender Datensatz über die Ansichtstabelle ausgewählt, so wird über die JavaScript-Funktion auf die dazugehörige Ansichtstabelle verlinkt.

6. Velocity

6.1. Was ist Velocity?

Velocity ist ein Open-Source-Projekt der Jakarta-Projektgruppe von Apache. Velocity ist eine javabasierte Template Engine. Sie ermöglicht den direkten Aufruf von Javaklassen aus Web-Seiten heraus. VTL ist die Abkürzung für *Velocity Template Language*.

 VTL-Referenzen beginnen mit dem Zeichen \$ und werden für Abfragen verwendet (z.B. Variablendefinitionen oder Klassenaufrufe, die Daten liefern). VTL-Anweisungen beginnen mit dem Zeichen # und werden für die Ausführung verwendet.

6.2. Wo wird Velocity angewendet?

Velocity-Aufrufe können ausschließlich auf VM-Seiten verarbeitet werden. Velocity-Befehle können direkt in eine statische VM eintragen oder über das XSL in dynamische Seiten eingebunden werden.

Der Applikationsdesigner bietet die Möglichkeit der Verarbeitung von Velocity-Befehlen im

- Expertmodus oder
- in den VTL-Kontrollen.

Darüber hinaus können über die Eigenschaftendialoge der Elemente im Applikationsdesigner Informationen aus Java-Klassen abgefragt werden (z.B. Informationen über den aktuellen Benutzer oder Parameter aus dem Request-Objekt).

6.3. Referenzen

6.3.1. Variablen

```
$foo
$a

#set( $a = "Velocity" )
$a ist der Name der Variablen, Velocity der Wert.

Beispiel
<html>
<body>#set( $foo = "Velocity" )
        Hello $foo World!
</body>
</html>
```

6.3.2. Properties

```
$customer.Address
$purchase.Total
```

6.3.3. Methoden

```
customer.getAddress()
$purchase.setTitel("My Home Page")
$person.setAttributes(["gross", "klein", "mittel"])
```

Formale Referenz Notation

```
${customer.address}
```

Die formale Notation ist notwendig für folgenden Text:

Dies ist ein \${art}fall. --- \${art} wird nun als Variable behandelt und ersetzt.

Stille Referenz Notation

Wenn Velocity auf eine undefinierte .Referenz trifft, wird einfach der Name der Referenz ausgegeben.

Beispiel:

```
<input type="text" name="email" value="$email"/>
```

Wenn \$email keinen Wert hat, sollte besser auch der *value* leer bleiben. Dies erreicht man durch folgende Notation: `<input type="text" name="email" value="$!email"/>`

Kombiniert mit der formalen Notation ist auch folgende Notation möglich:
`<input type="text" name="email" value="${!email}"/>`

Escaping Referenzen

```
#set($email = "foo")
```

```
$email
```

```
\$email
```

```
\\$email
```

```
\\\ $email
```

Die Ausgabe sähe so aus:

```
foo
```

```
$email
```

```
\foo
```

```
\\$email
```

Ist \$email nicht definiert:

```
$email
\\$email
\\\$email
\\\a href="#">$email
```

6.4. Kommentare

Kommentare sind in der Ergebnisdatei nicht sichtbar.

```
## Dies ist ein einzeliger Kommentar.
#*
Dies ist ein mehrzeiliger Kommentar, der im Web nicht angezeigt wird.
*#
```

6.5. #set

Die #set Anweisung weist einer Referenz ihren Wert zu.

```
#set ( $eigenschaft = "freundlich" )
#set ( $customer.behavior = $eigenschaft )
```

Links muss eine Variable oder Property Referenz angegeben werden.

Auf der rechten Seite sind möglich:

- Variablen Referenz
- Zeichenkette
- Property Referenz
- Methoden Referenz
- Nummer
- ArrayList
- Map

```
Beispiel für Nummer: #set ( $nummer = 123 )
ArrayList : #set ( $array = [ "Not", $my , "fault" ] )
Map : #set ( $map = { "banana" : "good", "roast beef" : "bad" } )
```

Hinweis: Zugriff auf die ArrayList über

\$array.get(index), d.h. \$array.get(0) liefert den Wert 'Not'.

Zugriff auf die Map über

\$map.get(name), d.h. \$map.get("banana") liefert den Wert ,good'.

Rechts können Berechnungen stehen:

```
#set ( $value = $foo + 4 )
```

Der return Wert null überschreibt **nicht** den Inhalt einer Referenz, d.h.

```
#set ( $result = $query.criteria("name" ) )
#set ( $result = $query.criteria("adresse" ) )
```

Wenn \$query.criteria("adresse") als Wert null zurückliefert, behält \$result den Wert aus dem ersten #set, also den Wert, den \$query.criteria("name") geliefert hat, z.B. Maier.

Beispiel: für eine for-each sollte daher Folgendes verwendet werden

```
#set( $criteria = ["name", "address" ] )

#foreach( $criterion in $criteria )

    #set( $result = false )
    #set( $result = $query.criteria($criterion) )

    #if( $result )
        Query was successful
    #end

#end
```

Für Stringlitterale gilt:

Verwenden von doppelten Hochkommata ""

```
#set( $directoryRoot = "www" )
#set( $templateName = "index.vm" )
#set( $template = "$directoryRoot/$templateName" )
$template
```

Die Ausgabe ist:

www/index.vm

Verwenden von einfachen Hochkommata ' ' - es findet kein parsing statt, d.h. Referenzen werden nicht aufgelöst.

```
#set( $foo = "bar" )
$foo
#set( $blargh = '$foo' )
$blargh
```

Die Ausgabe ist:

bar
\$foo

6.6. #if / #else / #elseif**6.6.1. #if**

```
#if( $foo )
  <strong>Velocity!</strong>
#end
```

6.6.2. #else

```
#set ( $foo = "deoxyribonucleic acid" )
#set ( $bar = "ribonucleic acid" )

#if ( $foo == $bar )
  In this case it's clear they aren't equivalent. So...
#else
  They are not equivalent and this will be the output.
#end
```

6.6.3. #elseif

```
#if( $foo < 10 )
  <strong>Go North</strong>
#elseif( $foo == 10 )
  <strong>Go East</strong>
#elseif( $bar == 6 )
  <strong>Go South</strong>
#else
  <strong>Go West</strong>
#end
```

6.7. Relationale und logische Operatoren

Bedingungen

Gleichheits Operator: #if(\$foo == \$bar)

Größer als: #if(\$foo > 42)

Kleiner als: #if(\$foo < 42)

Größer gleich: #if(\$foo >= 42)

Kleiner gleich: #if(\$foo <= 42)

Gleiche Zahl: #if(\$foo == 42)

Gleiche Zeichenkette: #if(\$foo == "bar")

Logisches NOT: #if(!\$foo)

logical AND

```
#if( $foo && $bar )
  <strong> This AND that</strong>
#end
```

logical OR

```
#if( $foo || $bar )
  <strong>This OR That</strong>
#end
```

logical NOT

```
#if( !$foo )
  <strong>NOT that</strong>
#end
```

6.8. #foreach

ArrayList:

```
#set ( $allProducts = [ "Handschuh", "Mütze", "Stiefel" ] )
<ul>
#foreach( $product in $allProducts )
  <li>$product</li>
#end
</ul>
```

Automatische Zählvariable: \$velocityCount (beginnt bei 1)

Map:

```
#set ( $allProducts = { "banana" : "good", "roast beef" : "bad" } )
<ul>
#foreach( $key in $allProducts.keySet() )
  <li>Key: $key -> Value: $allProducts.get($key)</li>
#end
</ul>
```

6.9. #include

Erlaubt den Import einer lokalen Datei.

```
#include( "one.txt" )
```

6.10. #parse

Erlaubt das Ausführen einer Velocity Datei.

```
#parse ( "statische.vm" )
#parse ( $varfürDateiname )
```

6.11. #macro

Erlaubt das Definieren von VTL Teilabschnitten, die beliebig oft wiederholt werden können.

Definieren des Macros

```
#macro( d )
<tr><td></td></tr>
#end
```

Aufruf des Macros

```
#d()
```

Definieren des Macros mit Parametern

```
#macro( tablerows $color $somelist )
#foreach( $something in $somelist )
```

```
<tr><td bgcolor=$color>$something</td></tr>
#end
#end
```

Aufruf des Macros mit Parametern

```
#set( $greatlakes = [ "Superior", "Michigan", "Huron", "Erie", "Ontario" ] )
#set( $color = "blue" )
<table>
  #tablerows( $color $greatlakes )
</table>
```

Ergebnis

```
<table>
  <tr><td bgcolor="blue">Superior</td></tr>
  <tr><td bgcolor="blue">Michigan</td></tr>
  <tr><td bgcolor="blue">Huron</td></tr>
  <tr><td bgcolor="blue">Erie</td></tr>
  <tr><td bgcolor="blue">Ontario</td></tr>
</table>
```

6.12. Range Operator

```
[n..m]

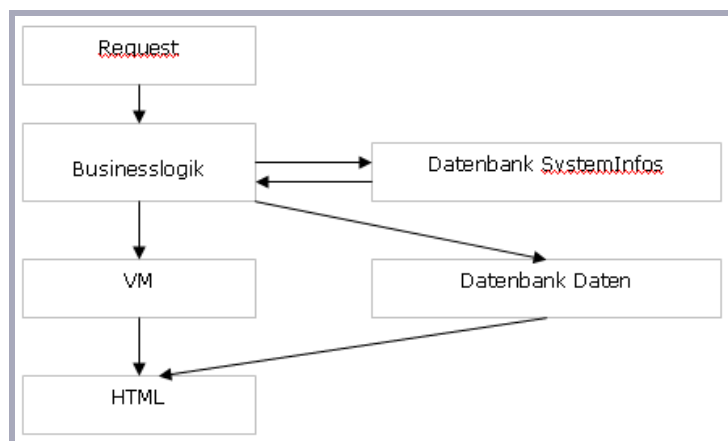
#foreach( $foo in [1..5] )
$foo
#end

#foreach( $bar in [2..-2] )
$bar
#end

#set( $arr = [0..1] )
#foreach( $i in $arr )
$i
#end
```

7. Aufruf einer Seite (2. Transformation)

Beim Aufruf einer Applikationsseite



8. Kontextobjekte unter Velocity

Das \$User-Objekt stellt alle Informationen zur Verfügung, die den aktuell angemeldeten User betreffen.

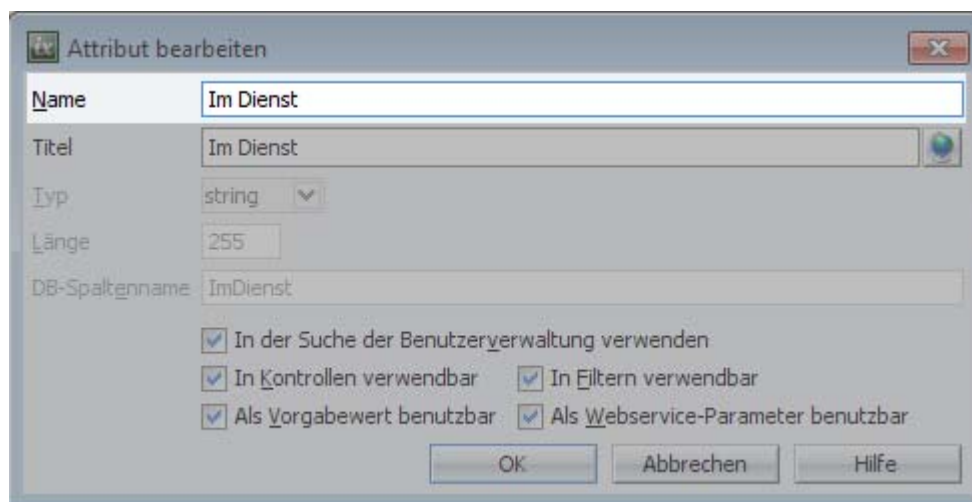
Beispiel Aufruf: *\$User.getLoginName()*

8.1. \$User

Das *\$User-Objekt* stellt alle Informationen zur Verfügung, die den aktuell angemeldeten User betreffen. Folgende Methoden stehen zur Verfügung:

Methode	Information zum aktuellen Benutzer
int getId()	ID
java.lang.String getGuid()	GUID
java.lang.String getEmployeeNo()	Personalnummer
java.lang.String getFirstName()	Vorname
java.lang.String getMiddleName()	2. Vorname
java.lang.String getTitle()	Titel
java.lang.String getFullName()	Vorname und Nachname
java.lang.String getLastName()	Nachname
java.lang.String getStreet()	Straße
java.lang.String getCountry()	Land
java.lang.String getZipCode()	Postleitzahl
java.lang.String getCity()	Ort
java.lang.String getEmailBiz()	eMail geschäftlich
java.lang.String getPhoneBiz()	Telefon geschäftlich
java.lang.String getPhoneMobileBiz()	Telefon Mobil geschäftlich
java.lang.String getPhoneFax()	Fax geschäftlich
java.lang.String getEmailHome()	eMail privat
java.lang.String getPhoneHome()	Telefon privat
java.lang.String getPhoneMobileHome()	Telefon Mobil privat
java.lang.String getDescription()	Beschreibung
java.util.Date getBirthday	Geburtsdatum
java.util.Date getEnterDate()	Eintrittsdatum
int getGender()	Geschlecht
byte[] getImageData()	Passfoto
java.lang.String getLoginName()	Anmeldename (ohne Domäne)
java.lang.String getQualifiedLoginName()	Loginname mit Domäne
java.lang.String getLoginDomain()	Domäne
java.util.TimeZone getTimeZone()	Zeitzone
int getBoss()	Vorgesetzter (ID)
java.util.Map getCustomMapVH	Liefert eine Map der Attribute
java.lang.String getPhonePager()	Pager
boolean isAnonymous()	Anmeldestatus
boolean isDisabled()	Konto gesperrt
java.lang.String getExternalLogin(0)	Loginname 1 unter Externe Logins
Java.lang.String getExternalPassword(0)	Passwort 1 unter Externe Logins

- ! Mit der Methode `$User.getLoginName()` wird der Anmeldename des aktuellen Benutzers ausgelesen. Attribute können mit der Methode `getCustomMapVH()` ausgelesen werden, wobei `name` der Name (die Bezeichnung) des *Attributs* ist: `$User.getCustomMapVH().get("name").getValue()`
 z.B: `$User.getCustomMapVH().get("Im Dienst").getValue()`



Beachten Sie bitte darüber hinaus, dass nach der Implementierung einer Abfrage von Benutzerfeldern der Browsercache geleert werden sollte und eine Neuansmeldung am Portal erfolgen muss.

Der Datenbank-Feldname des Attributs kann in der im Applikationsdesigner geöffneten Applikation *Benutzer* ermittelt werden. Mit Rechtsklick auf die Datengruppe *Benutzer* können alle Datenfelder über das Kontextmenü eingeblendet werden.

8.2. \$Request

Das Requestobjekt ist ein JavaObjekt. Es enthält alle Daten, die zur Request-Verarbeitung benötigt werden. Bei einer Anfrage an den Server (z.B. bei Klick auf eine Schaltfläche) wird dieses Objekt in der Methode `processRequest()` erzeugt, mit den notwendigen Daten gefüllt, (z.B. den Werten der Kontrollen, die gespeichert werden sollen) und an den Server geschickt. Die Methode `processRequest()` ist in der Datei `object.js` enthalten. Über das so erzeugte Objekt können auch Werte von einer Seite zur nächsten geschickt werden, die dort wiederum über Velocity abgefragt und weiterverarbeitet werden können. Folgende Methoden stehen im Request zur Verfügung:

java.lang.String \$Request.put("rq_meinParam", "wert")

Diese Methode übergibt einen Wert an das Request-Objekt, hier mit dem Namen `rq_meinParam`.

java.lang.String \$Request.get("rq_meinParam")

Diese Methode liefert den Wert zurück, der unter dem Namen `rq_meinParam` abgelegt ist.

Die Werte aus Eingabekontrollen werden unter dem Namen der Eingabekontrolle abgelegt. Damit können die Werte auf der folgenden Seite unter diesem Namen abgefragt werden (s. Kapitel *JavaScript / Automatisches Update*).

8.3. \$Loader (BusLogicCaller)

Über die *DataCollection* können die folgenden Werte abgefragt werden. Das entsprechende Businesslogikobjekt kann innerhalb von Velocity wie folgt angesprochen werden:

`$Loader.getDataCollection()`

Oder kurz:

`$DC`

Methode	Information
IDataRange getDataRange()	Datarange Objekt für die entsprechende Kontrolle
IDataRange getDataRange().getQuery	SQL-Select, der zum Aufbau des Datarange Objekt für die entsprechende Kontrolle abgesetzt wurde.
IDataRange getDataRangeBySysident()	Datarange Objekt für die entsprechende Kontrolle
java.lang.String getParentId()	liefert die ID des Elternsatzes (ParentId). Wenn keine ParentId gesetzt ist, wird der Wert -1 geliefert.
java.util.Map.getPropertiesVH()	Properties aus Property Datengruppe
java.lang.String getRecId()	liefert die aktuelle ID des Datensatzes
java.lang.String getUserId()	liefert die ID des Benutzers (UserId) des Datensatzes, bestimmt über das Feld, das den Sysident <i>UserId</i> gesetzt hat.
IValueHolder getValueHolder()	liefert den ValueHolder der Kontrolle mit dem in <i>Kontrollenname</i> angegebenen Namen.
IValueHolder getValueHolderBySysident()	
java.lang.String getValueStrByFieldGuid()	
java.lang.String getValueStrBySysident()	

\$Row in DC().getDataRange(Name der Kontrolle)
\$Row.getAttribute('Spaltenname')
\$Row.getRecId()

Listenelement für jede Zeile: Inhalt einer Spalte im Datarange Objekt

Velocity Debug Klasse

\$DEBUG.inspect(\$referenz)

Übersicht über Methoden und Eigenschaften des als Parameter übergebenen Objektes.

8.4. \$null

Hat den Wert null.

8.5. \$charset

Enthält den Wert des aktuell gültigen charsets. Dieser Wert wird auch in den HTTP-Header geschrieben.

8.6. \$lang

Enthält den Sprach-Identifizierer wie er zu Beginn der Requestverarbeitung aus Querystring, Cookies und den Systemeinstellungen (de.uplanet.lucy.util.UpLocaleFactory) ermittelt wurde.

8.7. \$UrlBuilder

Klasse, um Intrex-URLs aufzubauen. de.uplanet.lucy.server.composer.UrlBuilder

8.8. \$ObjectHelper

Hilfsklasse. Wird gegenwärtig verwendet, um Objekte auf null zu prüfen. de.uplanet.lucy.server.auxiliaries.ObjectHelper

8.9. \$Factory

Factory-Objekt zum Erzeugen von Objekten, die nicht in jedem Velocity-Kontext vorhanden sein müssen. Hauptsächlich dafür gedacht, die Performance und den Ressourcenverbrauch des Servers zu verbessern. de.uplanet.lucy.server.auxiliaries.ObjectFactory

- 8.10. **\$Response**
Das Response-Objekt ähnlich der HttpServletResponse.
de.uplanet.lucy.server.connector.web.HttpResponseWrapper
- 8.11. **\$Portal**
Das globale Webanwendungs-Objekt (entspricht dem Application-Objekt in ASP).
de.uplanet.lucy.server.auxiliaries.Portal.
- 8.12. **\$Session**
Die aktuelle Session.
de.uplanet.lucy.server.session.ISession
- 8.13. **\$DbConnection**
Die dem Kontext zugeordnete Datenbank-Connection
de.uplanet.jdbc.JdbcConnection.
- 8.14. **\$Request**
Das Request-Objekt.
de.uplanet.lucy.server.connector.IServerBridgeRequest
- 8.15. **\$Loader**
de.uplanet.lucy.server.composer.BuslogicCaller
- 8.16. **\$Chat**
Proxyobjekt für Chat und Notizen.
de.uplanet.lucy.server.auxiliaries.ChatProxy
- 8.17. **\$OMUC**
Wrapper-Klasse für diverse Use-Cases der Organisationsmanagements.
de.uplanet.lucy.server.auxiliaries.UserManagerWrapper
- 8.18. **\$VelocityContext**
Das VelocityContext-Objekt (als Initialisierungsparameter für andere Objekte im Kontext).
org.apache.velocity.VelocityContext
- 9. **Direkter Velocity-Aufruf**
Velocity Befehle können direkt in einer Ansichtskontrolle aufgerufen werden. Über einen direkten Aufruf soll der LoginName des Benutzers ausgegeben werden.

Übung

Erstellen Sie die Applikation *HTML und Velocity* auf Basis der Vorlage *Leere Applikation*. Legen Sie auf der Hauptseite ein Ansichtselement *Statischer Text* an und stellen Sie auf dem Reiter *Optionen* die Option *Nur Standardsprache (z.B. für Programmierung)* ein. Geben Sie im Titel des Ansichtselements den Velocity Befehl `$User.getLoginName()` ein. Erzeugen Sie über das Kontextmenü *Titel erzeugen* des Ansichtselements den Titel *Aktuell angemeldeter Benutzer*.



Der im Ansichtselement enthaltene Velocity Code wird ausgeführt und zeigt im Browser den Anmeldenamen des aktuell angemeldeten Benutzers an.



- 10. **Statische VMs in Velocity**
Über das Ansichtselement *VTL Include* lassen sich statische VMs direkt aufrufen. Statische VMs laufen nicht durch die XML/XSL Transformation. Sie werden direkt in der VM eingebunden.




Die VM kann im Eigenschaftendialog mit Pfad angegeben werden. Das obige Tabellenbeispiel kann auch ein *VTL Include* realisiert werden.

11. Query aus Velocity

11.1. Beispiel für PreparedQuery

Als Beispiel für PreparedQuery sehen Sie hier die *sample.vm* aus dem Verzeichnis

 [xtreme]\org\[Portalname]\internal\system\vm\html\include.

```
##this is a sample file for the integration of vm code to an application
##you can easiliy add an VTL Include to your Apllication, if you install the
VTL Include Patch available from United Planet

#####
##Following you find a sample to connect to a database and get the results
as an iterator object
#####

#####
## output macro
#####

#macro(output_data)

<td nowrap class="SCUP_Datarange_RowText_BG">
  <span class="SCUP_Datarange_RowText_normal">
    $!l_row.getValueHolder($!l_counter).getValue()
  </span>
</td>
#set($!l_counter = $!l_counter+1)
#if($!l_counter <= $!l_arglength)
  #output_data()
#end
#end

#####
##define your query and execute it
#####

#set($!l_stmt = $PreparedQuery.prepare($DbConnection, "SELECT LID as
id,STRLOGIN AS Login,STRFULLNAME as Name,STRMAILBIZ as Mail FROM DSUSER"))
#set($!l_rs = $!l_stmt.executeQuery())

#####
## write the header to the html output
#####
<table cellpadding="0" cellspacing="0"
class="SCUP_Datarange_Container_normal_BG" style="empty-cells: show;">
  <tr>
    <td nowrap class="SCUP_Datarange_Header_BG" align="right"><span
class="SCUP_Datarange_Header">ID</span></td>
    <td nowrap class="SCUP_Datarange_Header_BG" align="right"><span
class="SCUP_Datarange_Header">Login</span></td>
    <td nowrap class="SCUP_Datarange_Header_BG" align="right"><span
class="SCUP_Datarange_Header">Name</span></td>
    <td nowrap class="SCUP_Datarange_Header_BG" align="right"><span
class="SCUP_Datarange_Header">Mail</span></td>
  </tr>
```

```
#####
## get the content
#####
    #foreach($l_row in $l_rs.iterator())
        #set($l_arglength = $l_row.size())
        #set($l_counter = 1 )
        #if($l_even)
            #set($l_even = false)
            #set($l_style="SCUP_Datarange_RowOdd_normal_BG")
        #else
            #set($l_even = true)
            #set($l_style="SCUP_Datarange_RowEven_normal_BG")
        #end
        <tr class="$l_style">
            #output_data()
        </tr>
    #end
</table>

#####
##close the recordset
#####
$l_rs.close()
$l_stmt.close()
```

11.2. Einfache Datenbankabfrage

Das folgende Beispiel führt eine Abfrage auf die Benutzertabelle aus und liefert als Rückgabewert ein Resultset mit allen Einträgen dieser Tabelle. Die Abfrage wird in diesem Fall ohne Parameter innerhalb des Statements ausgeführt.

```
#set($l_statement = $PreparedQuery.prepare($DbConnection, "SELECT
STREMPLOYEENO AS No, STRFULLNAME as Name, STRPHONEBIZ as Tel, STRMAILBIZ as
Mail FROM dsuser"))
```

Erzeugt das Statement, in diesem Fall auf die Systemdatenbank (*\$DbConnection*).

```
#set($l_resultset = $l_statement.executeQuery())
```

Führt die Abfrage aus und gibt ein Resultset als Rückgabewert zurück.

```
<table>
<tr>
    <th>Nummer</th><th>Voller Name</th><th>Tel.</th><th>Mail</th>
</tr>

#foreach($l_row in $l_resultset.iterator())
Schleife durch alle Datensätze des Resultsets.
    <tr>
        <td>${l_row.getValueHolder(1).getValue()}</td>
        <td>${l_row.getValueHolder(2).getValue()}</td>
        <td>${l_row.getValueHolder(3).getValue()}</td>
        <td>${l_row.getValueHolder(4).getValue()}</td>
```

Die Rückgabewerte der Methoden *getValueHolder(int p_idx)* sind ValueHolder, die über Renderer in ein formatiertes Ausgabeformat gebracht werden können. In diesem Beispiel wurde der Übersichtlichkeit wegen darauf verzichtet und die Methode *getValue()* des ValueHolder-Objektes aufgerufen. Diese Methode liefert einen String mit der lexikalischen Entsprechung des Datentyps zurück.

```
    </tr>
#end
</table>

$l_resultset.close()
$l_statement.close()
```

12. Velocity und Parameter

Übung

In der Benutzerverwaltung im Menü *Benutzer / Schemamanager* wird das Attribut *Länderkennzeichen* definiert.

Bei jedem Benutzer wird eines der drei Länderkennzeichen *de*, *en* oder *fr* eingetragen.

Attribut	Typ	Größe	Wert
Länderkennzeichen	string	2	de

Geben Sie nun in der Applikation *HTML und Velocity* über die Eingabeseite zwei Testkunden ein. Tragen Sie bei einem Kunden das Länderkennzeichen *de*, beim anderen das Länderkennzeichen *en* ein.

KlNr.	Firma	Länderkennzeichen
10.000	ABC Engineering Inc.	en
10.001	Müller GmbH & Co. KG	de

In einer VM-Datei wird die SQL-Abfrage, die nach dem Länderkürzel des aktuellen Benutzers filtert, ausgeführt. Wechseln Sie in das Verzeichnis

`xtreme\org\[Portalname]\internal\system\vm\html\include` und erstellen Sie eine Kopie der Datei *sample.vm*. Benennen Sie die VM um in *laenderkennzeichen.vm*. Öffnen Sie die Datei und suchen Sie nach dem Kommentar *define your query and execute it*. Fügen Sie den folgenden Code in diesem Abschnitt ein:

```
## Länderkennzeichen
#set($lkz = $User.getCustomMapVH().get("Länderkennzeichen").getValue())

#set($sink = $CustomQuery.open("?"))
#set($query="SELECT T0.L_KUNDENNRINTEGER_32334E86 as KNr,
T0.STR_FIRMASHORTTEXT_586A0F18 as Firma, T0.STR_LNDRERKENNZEICHEN_BE378880 as
Laenderkennzeichen FROM datagroupD349D2AE T0 WHERE
T0.STR_LNDRERKENNZEICHEN_BE378880='$lkz' ORDER BY
T0.STR_FIRMASHORTTEXT_586A0F18 ASC")
#set($l_it = $CustomQuery.executeQuery($query))
```

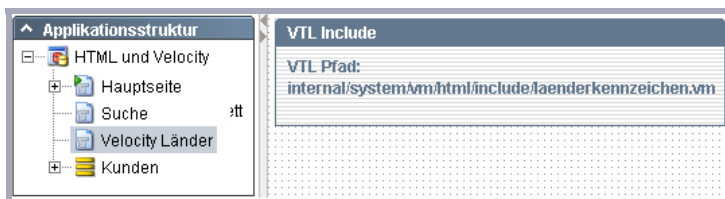
```
#set($lkz = $User.getCustomMapVH().get("Länderkennzeichen").getValue())
```

Ersetzen Sie die Angabe Länderkennzeichen ggfs. mit dem von Ihnen im Schemamanager vergebenen Namen (nicht Titel) des Attributs.

#set(\$query="SELECT....)

Ersetzen Sie ggfs. die hier angegebenen Datenfeldnamen und den Namen der Datengruppe mit den aktuellen Namen aus Ihrer Applikation. Die Feldnamen können bei geöffneter Applikation über das Kontextmenü *Datenfelder anzeigen* der Datengruppe *Kunden* ermittelt werden. Selektieren Sie das jeweilige Datenfeld in der Applikationsstruktur. Mit der Taste *F4* kann der Name aus dem Dialog *Details* kopiert und eingesetzt werden.

Legen Sie die neue Ansichtseite *Velocity Länder* in der Applikation an. Auf der Seite wird ein Ansichtselement *VTL Include* mit der neuen VM *laenderkennzeichen.vm* verbunden.



Nehmen Sie die neue Ansichtseite in das Applikationsmenü auf, speichern Sie die Applikation und melden Sie mit einem der Benutzer am Portal an, dem Sie ein Sprachkürzel in der Benutzerverwaltung zugeordnet haben. Testen Sie das Ergebnis im Browser.



12.1. Tabellennamen variabel abfragen

Beim Kopieren der Applikation *Fileexport* ändert sich der Tabellennamen. Damit die gleiche VM-Datei verwendet werden kann, kann der Tabellennamen variabel mit folgendem Verfahren ermittelt werden:

Im Applikationsdesigner wird über den Eigenschaftendialog der Datengruppe dem Expertattribut *sysident* ein Wert zugewiesen (z.B. *exportdaten*).



Mit dem folgenden Select-Statement kann der Tabellennamen ermittelt werden:

```
#set($l_strCardTablename =
$AppWalker.getTableBySysident($Request.get('rq_AppGuid'),
'exportdaten'))
```

Auf der Ansichtseite *Tabellennamen* wird das Select-Statement in einem Ansichtsfeld *statischer Text* als Titel eingefügt. Auf dem Reiter *Optionen* im Eigenschaftendialog wird die Option *Nur Standardsprache* (z.B. für Programmierung) gesetzt. Im Browser wird der aktuelle Tabellennamen der Datengruppe ausgegeben.

13. Schaltflächen

Schaltflächen führen in Intrexx Xtreme Aktionen aus, wie z.B. das Speichern, Löschen, Ändern und Auswählen von Datensätzen, das Laden von anderen Seiten, den Versand von eMails etc. Die Darstellung im Browser ist möglich als

Schaltfläche (*upButtonControl*)
Bild (*upImageActionControl*)
Text (*upTextActionControl*)

Die Funktionalität ist dabei, unabhängig von der Darstellung der Schaltfläche, immer dieselbe. Ein *upTextActionControl* bewirkt also dasselbe wie ein *upButtonControl*. Ein Action-control enthält eine Instanz des *upSource-Objektes* und eine Instanz des *upTarget-Objektes*. In diesen Instanzen sind die relevanten Informationen für den Request enthalten. Das *upSource-Objekt* definiert Eigenschaften der Quellseite des Requestes. Das *upTarget-Objekt* definiert die Anforderungen an die nächste zu ladende Seite, z.B. welche Seite mit welchem Datensatz geladen werden soll.

Alle Requests, die an den Server geschickt werden, werden in der Methode *processRequest()* aus der Datei *Object.js* assembliert.

Beispiel: Schaltflächentyp Schaltfläche, Speichern und Sprung auf eine andere Seite

```
<input id="ID_buttoncontrol1" name="buttoncontrol1" type="submit" style=""
class="SCUP_Button_Standard_normal"
onmouseover="if(this.oUp){this.oUp.setStatus(true);}return true;"
onmouseout="if(this.oUp){this.oUp.setStatus(false);}return true;"
value="Speichern">

<script type="text/javascript">
obuttoncontrol1 = new upButtonControl();
obuttoncontrol1.biDirectUpHtml (obuttoncontrol1,
'ID_buttoncontrol1');
obuttoncontrol1.description = "OK";
obuttoncontrol1.oFup = oeditpageB598206B;
obuttoncontrol1.linkType = "16";
obuttoncontrol1.oSource = new upSource();
obuttoncontrol1.oSource.fr_ActionId = "1";

obuttoncontrol1.oTarget = new upTarget();
obuttoncontrol1.oTarget.rq_TargetId = "1000";
obuttoncontrol1.oTarget.rq_AppId = "1013";
obuttoncontrol1.userName = "Speichern";
</script>

<script type="text/javascript">
obuttoncontrol1.oHtml.onclick = onclickHandlerbuttoncontrol1;
function onclickHandlerbuttoncontrol1(e)
{
var rv = true;
this.oUp.processRequest(e);
if(!bProcessFormAction)return false;
}
</script>
```

Beispiel: Schaltflächentyp Text, Login

```
<a id="id_textactioncontrollogin" href="#"
class="SCUP_Login_Standard" onmouseover="this.oUp.setStatus(true);return
true;" onmouseout="this.oUp.setStatus(false);return true;">Text</a>

<script type="text/javascript" language="javascript">
otextactioncontrollogin = new upTextActionControl();
```

```

    otextactioncontrollogin =
otextactioncontrollogin.bIDirectUpHtml(otextactioncontrollogin,
'ID_textactioncontrollogin');
    otextactioncontrollogin.linkType = "14";
//z.B. Für Submit Aktion, welches Form soll submitted werden
//otextactioncontrollogin.oHtml.form = oformgroup7f27ab17.oHtml;
    otextactioncontrollogin.oTarget = new upTarget();
    otextactioncontrollogin.oTarget.newWindow = "1";
    otextactioncontrollogin.oTarget.rq_TargetFrame = "wndLogin";
    otextactioncontrollogin.oTarget.rq_Template =
"internal/system/vm/html/login/login.vm";

    otextactioncontrollogin.oHtml.onclick =
onclickHandlertextactioncontrollogin;
    function onclickHandlertextactioncontrollogin(e)
    {
        this.oUp.processRequest(e);
        return false;
    }
</script>

```

13.1. Eigenschaften und Methoden des Actioncontrol-Objektes

Parameter	Werte	Bedeutung	Attribut	Pflicht
Allgemein				
oMyAction	new upTextActionControl();	Definiere TextActionControl Object		ja
oMyAction = oMyAction.bIDirectUpHtml(oMyAction, 'ID_button');		Erzeuge Bezug zum HTML Element (bIDirektionale Assoziation)		Ja
.description	= Beschreibung	Text für Statuszeile und Tooltip		
.linkType	Siehe Werteliste			
.oHtml.form	"[FORM NAME]"	Referenz zum Html-Formular (beim Button schon vorhanden)		
.userName	Name des Anwenders	Anzeigename für den Anwender, z.B. beim Validieren des Formulars in Fehlermeldungen.		
.oHtml.onclick	onclickHandlerbutton	Handler, der beim Click auf das Control aufgerufen werden soll		
oTarget				
.oTarget	new upTarget();	Erzeuge Targetobject		
.oTarget.newWindow	"1" / "0"	Öffne neues Fenster	new- Window	
.oTarget.rq_Template	VM-File, wenn Ziel statisches VM			
.oTarget.addParam	Helper.setQsValueByParam("Name", "value", oMyAction.o Target.addParam);	Setze zusätzliche Requestparameter		
.oTarget.windowSettings	"400,300,0,0,1";	Bei neuem Fenster: Größe setzen, Fupid des aufrufenden Fensters übergeben (width, height, offset x, offset y, boolean Positionierung am ActionControl)		Ja, bei Popups
.oTarget.props	"dependent=yes,menubar= no,toolbar=no,scrollbars=ye s,resizable=yes"	Bei neuem Fenster: Eigenschaften		
.oTarget.rq_TargetId	Fupid des Ziels, wenn kein rq_Template			
.oTarget.rq_AppId	AppId des Ziels, wenn kein rq_Template			
.oTarget.rq_ReclId	Zu ladender Datensatz			
.oTarget.rq_SId	SessionId, falls Cookies deaktiviert sind			
.oTarget.rq_ClientType	Zur Zeit nur <i>html</i>	Type des Clients: determiniert, welche VM die Daten verarbeitet.		
.oTarget.rq_Lang	Sprachkürzel nach ISO	Steuert auszugebende Sprache		
.oTarget.rq_TargetFrame	Frame Name, in dem Ziel geöffnet werden soll			

oSource				
.oSource.fr_ActionId	Siehe Werteliste			
.oSource.rq_SourceId	FupId der aktuellen Seite			
.oSource.rq_SourceAppId	AppId der aktuellen Seite			
.oSource.rq_SourceRecId	RecordId der aktuellen Seite			
.oSource.rq_SourceParentId	Datensatz-Id des übergeordneten Datensatzes			

13.2. LinkType

Wert	Beschreibung
0	Sprung auf bestehenden Datensatz
1	Aufruf einer externen URL; in rq_TargetUrl wird die eingestellte URL geschrieben
2	Sprung auf eine Seite einer Mutterdatengruppe ODER Speichern eines neuen Datensatzes mit Sprung auf denselben Datensatz
4	Wechseln der Sprache (changeLang () statt processRequest())
5	Datapicker auslösen
6	neuen Datensatz aus Popup hinzufügen
7	Datarange in neuem Popup öffnen
8	Datumspicker
9	Sprung auf neuen Datensatz
10	Textexport aus Datarange
11	Mailen aus Datarange
12	Download von FileControls / ImageFileControls (noch aktiv?)
13	Kategorienauswahl
14	Login
15	Flip Flop
16	Sprung auf Hauptseite

13.3. ActionId

Wert	Beschreibung
1	Update / Insert (Abhängig, ob recId = -1)
2	Löschen

14. Ajax

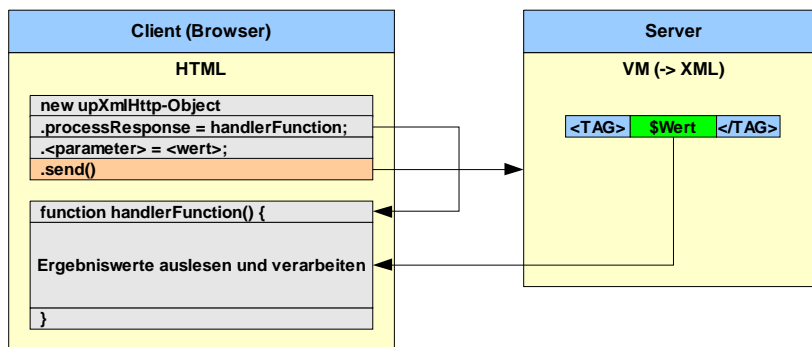
14.1. Was ist Ajax

Ajax steht für „Asynchronous JavaScript and XML“ und bedeutet die asynchrone Kommunikation zwischen dem Server und dem Browser. Es wird primär verwendet, um mehr Performance im Portal sowie die Bedienung für den Benutzer intuitiver zu gestalten. Performance bedeutet, überflüssige Reloads ganzer Seiten zu verhindern und gezielt im Hintergrund Daten abzurufen und direkt an betroffener Stelle zu ändern (DHTML). Die Realisierung von „Rich Applications“, man meint hierbei web basierende Anwendungen, die sich nun wie eine compilierte, lokal installierte Software verhält (z.B. Office-Software, Groupware, ...).

14.2. Ajax in Intrexx Xtreme

Für das upXmlHttp-Objekt muss zunächst auf der entsprechenden Intrexx-Seite eine neue Instanz erstellt werden. Für diese Objekt-Instanz werden nun verschiedene Parameter definiert, darunter auch die Funktion, welche die Antwort des Requests ausliest und verarbeitet sowie die VM-Datei, welche den Request serverseitig behandelt und die Ergebnisse in XML-Form bereitstellt.

Liefert der Server eine Antwort auf den Request, wird das processResponse-Ereignis im Browser ausgelöst und die Handler-Funktion ausgeführt. Über das oRequest-Objekt können nun die Ergebniswerte ausgelesen und verarbeitet werden.



! Im Internet-Explorer 6 wird das XMLHttpRequest-Objekt über ein ActiveX realisiert. Daher ist diese Ajax-Funktionalität nur mit „aktiviertem“ ActiveX nutzbar. Ab Version 7 ist dies nicht mehr der Fall.

14.2.1. Das upXmlHttpRequest-Objekt

Die folgende Tabelle zeigt alle Eigenschaften und Methoden des upXmlHttpRequest-Objekts.

Methode/Eigenschaft	Beschreibung
.bProcessResponse = <true/false>	Initialisierung
.processResponse = <Handlerfunktion>	Zuweisung der Handlerfunktion, die beim Auslösen des Request-Antwort-Ereignisses abgearbeitet werden soll.
.bAsync = <true/false>	Asynchrone Kommunikation (default = true) Achtung: bei false = synchrone Kommunikation sind Endlos-Schleifen möglich (Browser-Freeze)
.strMethod	Absendemethode: „POST“, „GET“
.bAddFormData	
.strURL	URL für VM-Datei, die den Request verarbeiten soll
.send()	Methode zum Senden des Requests

Beispiel:

```
var meinXmlHttpRequest = new upXmlHttpRequest;
meinXmlHttpRequest.bProcessResponse = true;
meinXmlHttpRequest.processResponse = meinHandler;
meinXmlHttpRequest.bAsync = true;
meinXmlHttpRequest.strURL = "";
meinXmlHttpRequest.send();
```

14.2.2. ActionControls

Das Ausführen eines Ajax-Requests kann auch über ActionControls erfolgen, um Werte via Requestparameter an die Velocity-Datei zu übergeben. In dieser können die Werte entsprechend verarbeitet werden.

```
var oAction = new upActionControl();
oAction.oHtml = new Object();
oAction.oTarget = new upTarget();
oAction.oTarget.rq_Template = "internal/system/vm/html/chat/getnotifymessage.vm";
oAction.requestType = 2;
oAction.oXmlHttpRequest.bAsync = true;
oAction.oXmlHttpRequest.bProcessResponse = true;
oAction.oXmlHttpRequest.processResponse = processPull;
oAction.processRequest();
```

RequestType	Beschreibung
2	xmlHttp "Get"
3	xmlHttp "Post"

14.2.3. Request-Verarbeitung

Die Verarbeitung eines Requests erfolgt mit Hilfe einer Velocity-Datei. Diese kann z.B. über eine customQuery Informationen ermitteln und bereitstellen. Wichtig ist, dass die Velocity-Datei das Ergebnis in XML-Form rendert und bereitstellt. Hierzu muss der Header entsprechend erzeugt werden. Die Tags für die Rückgabeparameter sind durch einen übergeordneten Tag mit der Bezeichnung <response> zusammenzufassen.

```

$Response.setIgnoreWrite(true)

:
: Hier werden mit Velocity und Java-Klassen die Ergebnisse ermittelt
: bzw. Aktionen ausgeführt!
:

$Response.setIgnoreWrite(false)
$Response.setHeader("Cache-Control","no-cache")
$Response.setHeader("Content-Type","text/xml; charset=$charset")
$Response.setIgnoreWrite(false)<?xml version="1.0" encoding="UTF-8"
standalone="yes"?>

<response>
:
  <tag>$Variable</tag>
  : Ergebnisse immer in XML-Tags
:
</response>

<response>

<tag>Eintrag1</tag> -> referenz.getElementsByTagName('tag-
name')[0].firstChild.data;
<tag>Eintrag2</tag> -> referenz.getElementsByTagName('tag-
name')[1].firstChild.data;

<tag>$ArrayToString<tag>
    
```

14.2.4. oRequest-Objekt

Das oRequest-Objekt ermöglicht das Auslesen bzw. das Auswerten der Ergebnisse des Requests. Über das Objekt kann im ersten Schritt das Dokument referenziert werden, welches das Ergebnis der Requesanfrage enthält (XML aus VM). Anschließend kann über das DOM auf die einzelnen Inhalte per TAG zugegriffen werden.

Eigenschaft	Beschreibung
.responseXML.documentElement	Referenzierung auf das XML-Dokument. Weiteres Auslesen über das DOM (nachfolgende Eigenschaften)
.responseText	XML-Antwort
.readyState	= 4 (Vollständig), 1,2,3 (Unvollständig)
.status	= 200 (ok), <> 200 Fehler
.statusText	Statustext (bei status <> 200 = Fehlermeldung)

14.3. Ajax und XML-Response zurückgeben

14.3.1. Rückgabe von Werten

Übung

Erstellen Sie eine neue Applikation auf Basis der Vorlage *Leere Applikation*, benennen Sie diese um in *Ajax und XML Response*. Fügen Sie auf der Hauptseite ein *statisches Ansichtelement* ein und löschen Sie den Titel. Stellen Sie die Anzeige auf *HTML* um (Reiter *Optionen*) und wählen Sie auf dem Reiter *Ansicht* die Stilklasse *Standard*.

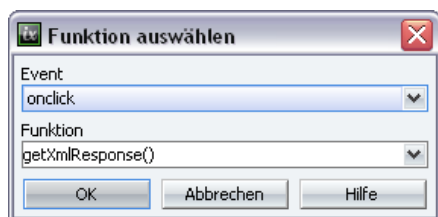
Erstellen Sie eine Schaltfläche mit dem Titel *Ausgabe* ohne Aktion und wechseln Sie in den Skripteditor:

```
function getXmlResponse()
{
    /*
    * Neues Objekt erzeugen vom Typ upSimpleAjax
    */
    var mySimpleAjax = new upSimpleAjax();

    mySimpleAjax.oProcessFunc = myXmlHandler1; // ResponseHandler

    /*
    * Sends a request to a given vtl-file,
    * which produces an xml response.
    */
    mySimpleAjax.loadXmlVm("internal/system/vm/custom/xml.vm");
}
```

Ordnen sie die genannte Funktion dem *onclick*-Event der Schaltfläche zu.



Erstellen Sie die JS-Funktion *myXmlHandler1* wie folgt.

```
function myXmlHandler1(oXMLDocumentElement)
{
    var oLabelAusgabe = getElement("BD5E...D5CE"); /* labelcontrol*/
    oLabelAusgabe.innerHTML = oXMLDocumentElement.firstChild.data;
}
```

Speichern die Applikation.

Legen Sie die in Funktion *getXMLResponse* aufgerufene VM-Datei *xml.vm* im nächsten Schritt im Verzeichnis `<xtreme>/org/<portal>/internal/system/vm/custom/` an. Als Inhalt schreiben wir mit Hilfe eines Editors folgende Zeile ein:

Dieser Text ist **fett**.

Testen Sie die Applikation im Browser.

14.3.2. Requestwerte und Formularelemente mitsenden

Übung

Erstellen Sie auf der Hauptseite die zwei Eingabefelder *Datum* (Typ Datum & Uhrzeit, Keine Verknüpfung) *Preis* (Typ Währung, Keine Verknüpfung). und fügen Sie ein *statisches Ansichtelement* ein, löschen Sie den Titel, stellen unter *Optionen* auf *HTML* um und unter *Ansicht* auf die Stilklasse *Standard*.

Erstellen Sie anschließend eine Schaltfläche *Ausgabe* ohne Aktion und wechseln Sie in den Skript-Editor:

```
function getXmlResponse4()
{
    var mySimpleAjax = new upSimpleAjax();
    mySimpleAjax.oProcessFunc = myXmlHandler4;

    // DateTime und Float Werte posten
    var oLocalDateTime = getElement("1747...ABF5"); /*Datum
datetimecontrol*/
    var oJsDateTime = getDateObject(oLocalDateTime);
    var strISODatetime =
oLocalDateTime.oUp.toISODatetimeString(oJsDateTime);

    var strLocalFloat = getElement("A930...4403").value; /*Preis
currencycontrol*/
    var strFloat = Helper.getFloatStringByLocal(strLocalFloat);

    mySimpleAjax.loadXmlVm("internal/system/vm/custom/xml4.vm", {rq_param1
:strISODatetime, rq_param2:"value2"}, {fr_param1:strFloat});
}
```

Ordnen Sie die Funktion dem *onclick*-Event der Schaltfläche zu.

