Coding Conventions für FirstSpirit Projekte

Version	**1.8**
Status	**in process**
Datum	**2011-11-10**
Abteilung	Techn. Documentation
Autor/ Autoren	Professional Service
Dateiname	FirstSpirit_Coding_Conventions.doc
Dokumentvorlage	FirstSpirit_Dokumentenvorlage.dot

e-Spirit AG
Barcelonaweg 14
44269 Dortmund | Germany

phone +49 231 . 286 61-30
fax +49 231 . 286 61-59

eMail info@e-spirit.com
web www.e-spirit.com

Inhaltsverzeichnis

Informationen zum Dokument

Änderungs-Historie			
Version	**Datum**	**Autoren**	
1.0	10.07.2003	Andreas Knoor	initiale Version
1.1	14.08.2003	Sascha Rusch	Anmerkung hinzugefügt, dass Site-Store-Variablen grundsätzlich in der Wurzel definiert werden sollen. set-ft- für Formattemplates hinzugefügt (CMS_RENDER) ...sc.. für Skripte hinzugefügt set-ps- für Projekteinstellungen hinzugefügt Hinweis auf die richtige Wahl des Resultnames bei „include“-Funktionen
1.2	27.08.2003	Sascha Rusch	...cs... für Content-Schemata hinzugefügt
1.3	20.04.2006	Sebastian Gockel	Präfix-Trenner von – auf _ geändert Vorbereitung 4.0
1.4	09.04.2007	Andreas Knoor	Vereinheitlichung von Bezeichnungen
1.5	26.09.2011	Tobias Klein	Einführung tt_ statt cs_ für Tabellenvorlagen und kleinere Ergänzungen
1.6	26.09.2011	Raphael Richter	Weitere Ergänzung von Variablen-Typen und Beispielen
1.7	28.09.2011	Raphael Richter	Do’s and Don’ts für Templating und Scripting hinzugefügt
1.8	10.11.2011	Tobias Klein	Kleinere Anpassungen

1 Einführung

Dieses Dokument ergänzt die Basis FirstSpirit Entwickler-Dokumentation [1] um Konventionen im Bereich der Benennung von Ordnern und Objekten innerhalb der verschiedenen FirstSpirit-Verwaltungen.

Ein besonderer Schwerpunkt liegt auf Konventionen im Bereich der Variablen-Benennung, die es Entwicklern erleichtern soll, die Herkunft und den Deklarationsort von Variablen anhand einer passenden Benennung zu bestimmen und dadurch die Übersichtlichkeit (gerade in Projekten, in denen mehrere Entwickler arbeiten) zu fördern.

Im Laufe der Jahre hat sich dieses Dokument zu einem wichtigen Leitfaden für FirstSpirit-Projekte entwickelt.

2 Benennung von Ordnern und Objekten

Im Folgenden werden Namenskonventionen für Referenznamen von Ordnern und Objekten definiert.

Referenznamen haben große Bedeutung für die Wartbarkeit. Von Beginn an soll auf die korrekte Verwendung von Referenznamen geachtet werden. Für die Entwickler bietet sich die Anzeige der Referenznamen in der Baumansicht an.

Zufällig falsch vergebene Referenznamen sollten auf jeden Fall direkt korrigiert werden, auch wenn dies zu großem Änderungsaufwand führt.

Im Gegensatz zu den Referenznamen werden in diesem Dokument keine Vorgaben für die sprachabhängig pflegbaren Anzeigenamen der Objekte gemacht. Dies ist meist eine fachliche Entscheidung, die im Projekt gefällt werden muss und somit Bestandteil des Template-Konzeptes ist.

Die Referenznamen von Ordnern und Objekten in der Inhalte-Verwaltung, der Struktur-Verwaltung, der Medien-Verwaltung und der Vorlagen-Verwaltung sind folgendermaßen zu wählen:

- kleingeschrieben
- einheitlich auf englisch oder projektspezifisch
- Trennung von Wörtern per Unterstrich
- möglichst nicht zu lang (besonders in der Struktur-Verwaltung)
- Trennung von Indizes per Unterstrich

Beispiele:

- news
- product_categories
- pressarchive
- whoiswho_changed_1
- whoiswho_changed_2

Wichtig: Bei der Vergabe von Ordnernamen nach diesem Schema ist in der Struktur-Verwaltung zu beachten, dass der Name für die Anzeige in der Sitemap noch manuell umgeändert werden muss!

Dies sollte man immer tun, egal ob das Projekt zum aktuellen Zeitpunkt die Sitemap-Funktion verwendet oder nicht.

Wichtig: Templates haben zusätzlich zu sprachunabhängigen Referenz- und sprachabhängigen Anzeigenamen noch sprachunabhängige Beschreibungstexte. Wird der Beschreibungstext nicht für andere projektspezifischen Informationen genutzt, so sollte man die Beschreibungstexte wie folgt wählen:

- auf englisch bzw. projektspezifisch an die Sprache der Entwickler angepasst
- möglichst aussagekräftig und eindeutig
- Trennung der Wörter durch Leerzeichen
- ggf. mit Hinweis auf den verwendeten Bereich versehen

2.1 Benennung von Variablen

Ein besonderer Schwerpunkt gilt der Benennung von Variablen, die an verschiedenen Orten definiert werden können.

Je nach Verwendung sollten Variablen mit bestimmten Präfixen versehen werden. Dies erleichtert das Verständnis für das FirstSpirit-Projekt mitunter erheblich und gibt - korrekt verwendet - Auskunft über die Herkunft einer Variablen.

2.1.1 Allgemeine Regeln

Allgemein sind Variablennamen folgendermaßen zu wählen:

- englisch oder einheitlich in einer anderen projektspezifischen Sprache
- mit korrektem Präfix (siehe 2.1.2)
- keine Sonderzeichen, keine Leerzeichen, keine Bindestriche
- Trennung von Wörtern durch Groß-/Kleinschreibung (CamelCase) bzw. Unterstriche

Beispiele:

- pt_title

- st_picture
- ss_showNavigation
- gv_absoluteUrlPrefix

2.1.2 Verwendung von Präfixen

Je nach Verwendungsart und Deklarationsort bekommt eine Variable, die in oder über einer Vorlage definiert wird, einen der folgenden Präfixe:

Vorlagen-Typ	Präfix
Seitenvorlage	pt_
Absatzvorlage	st_
Link-Vorlage	lt_
Format-Vorlage (auch Tabellen-Stilvorlage)	ft_
Projekteinstellungen	ps_
Metadaten-Vorlage	md_
Seitenvorlage einer Global Content Area (GCA)	gc_
Tabellen-Vorlage	tt_
Skript	sc_

Tabelle 1 Übersicht – Variablen-Präfixe in Vorlagen

Des Weiteren gibt es die folgende Variablen-Typen:

- Variablen in der Strukturverwaltung
- Ergebnisse von Funktionsaufrufen
- Temporäre Variablen in CMS_SET-Anweisungen
- Variablen, die innerhalb von „geparsten“ Textdateien in der Medienverwaltung definiert werden
- Laufvariablen in CMS_FOR-Anweisungen
- gv = Global-Value, d.h. Variablen die gesetzt und anschließend irgendwo überschrieben werden (nicht-read-only-Variablen)
- dv = Deployment-Value, d.h. Variablen, die nur für ein Deployment-Ziel definiert wird

Wichtig: Variablen, die z.B. im Site-Store gesetzt werden und im Deployment überschrieben werden, bekommen das Präfix „gv“

Folgende Präfixe werden für die verschiedenen Variablen-Typen verwendet:

Variablen-Typ	Präfix
Struktur-Variable	ss_ **Wichtig:** Variablen in der Struktur-Verwaltung sollten grundsätzlich in der Struktur-Wurzel definiert werden! *Beispiel*: ss_shownavigation
Ergebnisse von Funktionsaufrufen	fr_ **Wichtig:** Es sollte durch einen weiteren Präfix festgelegt werden, in welcher Vorlage das Funktionsergebnis berechnet worden ist. Hierzu werden die oben beschriebenen Vorlagen-Präfixe verwendet. *Beispiele:* fr_st_sidebarmenu, fr_pt_mainnavigation
Temporäre Variablen in CMS_SET	set_ **Wichtig:** Es sollte durch einen weiteren o.a. Vorlagen-Präfix angegeben werden, wo das CMS_SET aufgerufen worden ist. *Beispiele:* set_st_teaserbox set_pt_producthighlight
Medienverwaltung	ms_
Laufvariablen in CMS_FOR-Anweisungen	for_

Tabelle 2 Übersicht – Weitere Variablen-Präfixe

2.1.3 Konfiguration von Eingabekomponenten

Für das „name“-Attribut von Eingabekomponenten (im Formular-Reiter der Vorlagen) gelten die Regeln unter 2.1.1.

Die Anzeigesprachen für das Attribut „label“ und das Attribut „comment“ und deren Befüllung werden fachlich im Projekt definiert und sollten Teil des projektspezifischen Vorlagen-Konzepts sein. Für ein Vorlagen-Konzept bietet sich die folgende Darstellungsweise für die Definition von Eingabekomponenten einer Vorlage an (Beispiel Seitenvorlage):

Label	Typ	Reiter	spr. -abh.	Pflicht	Variable	Beschreibung
Browser Titelzeile	TEXT	1	ja	Nein	pt_metaTitle	Attribut für das title-Tag
Meta-Description	TEXTAREA	1	ja	Nein	pt_metaDescription	Befüllung der Meta-Description im HTML der Seite
Meta-Keywords	LIST	1	ja	Nein	pt_metaKeywords	Befüllung der Meta-Keywords im HTML der Seite
Header-Bild	REFERENCE	2	nein	Nein	pt_headerPicture	Seitenbezogenes Header-Bild
Schriftfarbe Introtext / Headline	COMBOBOX	2	nein	ja	pt_fontColor	dunkel / mittel / hell
Headline	TEXT	2	ja	Ja	pt_headline	Headline
Intro-Text	DOM	2	ja	Nein	pt_introText	Formatierbarer Intro-Text als Layer über dem Header-Bild
Link	LINK	2	ja	Nein	pt_link	Interner, externer, Event-, Presse- oder Produkt-Link unter Verwendung spezieller Linkvorlagen
Kontakt anzeigen?	TOGGLE	2	ja	nein	pt_showContact	Soll eine Kontaktbox angezeigt werden?
Kontakt	OBJECTCHOOSER	2	ja	Nein	pt_contact	Kontaktauswahl aus einer Datenquelle

Tabelle 3 Beispiel – Vorlagen-Definition Seitenvorlage

Wichtig: Eingabefelder sollten der Übersicht halber auf verschiedene Reiter durch Verwendung des Formular-Elements <CMS_GROUP> gruppiert werden. Hierzu wird in den Tabellen des Vorlagen-Konzepts die Spalte „Reiter“ vorgesehen.

Für alle Eingabefelder gilt: Alle unnötigen Felder sollten so weit wie möglich ausgeblendet werden.

Beispiel:

Bei CMS_INPUT_PICTURE sollte das Attribut „lean“ auf den Wert „mandatory“ gesetzt werden, damit nur das eigentliche Bild eingegeben werden kann, sofern der Redakteur die anderen Felder nicht befüllen soll.

2.1.4 Namenskonventionen im Bereich der Datenbankschemata

2.1.4.1 Tabellen

Folgende Namenskonventionen gelten für Tabellen innerhalb eines FirstSpirit Datenbank-Schemas:

- englisch
- Pluralformen
- Kleinschreibung
- getrennt durch Unterstriche

2.1.4.2 Spalten

Folgende Namenskonventionen gelten für Spalten innerhalb der Tabellen eines FirstSpirit Datenbank-Schemas:

- englisch
- Singular
- Kleinschreibung
- getrennt durch Unterstriche

Do's and Don'ts

2.2 Templating

2.2.1 Auslagerung von Template-Code

Ein paar Hinweise zur Auslagerung von Template-Code und Vorlagen-Scripten:

- immer wieder kehrender Template-Code sollte wenn möglich in Format-Vorlagen ausgelagert werden
- bevor ein Vorlagen-Skript entwickelt wird, sollte geprüft werden, ob sich die Funktionalität nicht mit Template-Syntax abbilden lässt

2.3 Scripting

Beim Scripting mit BeanShell gibt es einige wenige Regeln, an die sich ein Vorlagenentwickler halten sollte, um Skripte wartungsarm und effizient zu gestalten.

Jedes Script sollte unter den folgenden Gesichtspunkten kritisch beäugt werden:

- werden Connections aufgemacht, obwohl sie nicht gebraucht werden?
- werden Objekte korrekt gelockt? (s. 2.3.2)
- gibt es hardcodierte Werte (Auftrag-Ids, Projekt-Ids) innerhalb von Scripten, die noch (z.B. in die Projekteinstellungen) ausgelagert werden müssen?
- ist das Script so groß, dass man es besser in ein Modul (Executable) auslagert?

In den folgenden Kapiteln werden bestimmte Aspekte genauer betrachtet.

2.3.1 Importieren von Klassen

FirstSpirit- oder Java-Klassen sollten aus Performance-Gründen immer im Header eines Skripts importiert werden.

Schlechtes Beispiel:

```
//!Beanshell
// Schlecht
if (myVar instanceof java.lang.String) {
   print("its a string!");
}
```

Gutes Beispiel:

```
//!Beanshell
// Gut!
import java.lang.String;
if (myVar instanceof String) {
   print("its a string!");
}
```

2.3.2 Sperren von Objekten (Locking)

Das Setzen von Sperren (Locks) sollte immer nach folgendem Schema erfolgen:

```
//!Beanshell
import de.espirit.firstspirit.access.store.LockException;

try {
  elm.setLock(true, false);
  try {
    ...
    elm.save("some comment", false);
  } catch (Exception e) {
    // catch handling
  } finally {
    elm.setLock(false, false);
  }
} catch (LockException e) {
  // Element locked, catch handling
}
```

Nur damit ist ein sicheres Entfernen der Sperre garantiert.

2.3.3 Iteration über die Kinder eines Elementes

Es kommt häufig vor, dass man über alle Kinder eines bestimmten Elementes (z.B. Ordner) iterieren möchte.

Wichtig ist dabei zu beachten, dass die Liste (insbesondere wenn rekursive alle Kinder betrachtet werden) sehr lang sein kann und daher immer mit Iteratoren (niemals mit Arrays oder kompletten Listen) gearbeitet werden sollte.

Hier ein Code-Beispiel für die korrekte Iteration über die Kinder eines Elementes im Baum, wobei die Kinder nach einer bestimmten Klasse (hier "MyClass") gefiltert werden.

```
//!Beanshell

// ..Ermittlung der Variable folder...

for (elem : folder.getChildren(MyClass.class, true).iterator()) {
    print(elm.getSomeValue(...));
}
```

2.3.4 Logausgaben

Die Ausgabe eines Ausdrucks oder einer Variablen erfolgt in BeanShell über das Kommando "print()", was allerdings ein schlechtes Beispiel für eine Logausgabe ist:

```
//!Beanshell
// Schlecht
print();
```

Diese Ausgabe sollte durch eine differenzierte Ausgabe über die drei Logging-Methoden:

- logInfo
- logError
- logDebug

ersetzt werden.

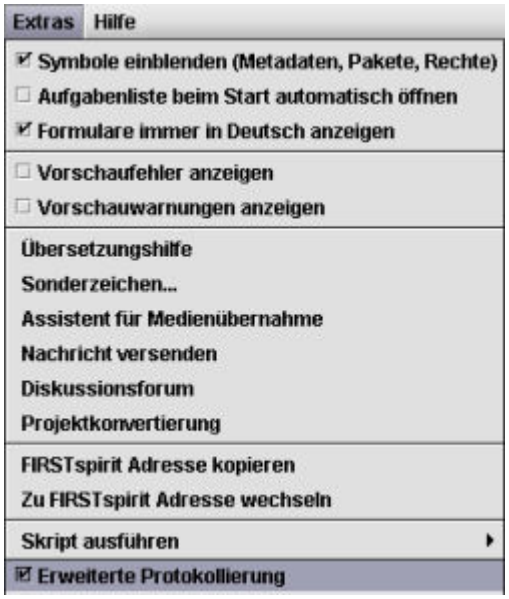
Die Log-Methoden stehen im jeweiligen Skript-Kontext zur Verfügung.

Beispiel:

```
//!Beanshell
// gut
context.logDebug(...);
```

Der Vorteil ist: die Protokollierung von Skripten kann auf einfache Art beeinflusst werden. So kann für Skripte beispielsweise eine "erweiterte Protokollierung" im FIRSTspirit-JAVAclient aktiviert werden (Checkbox im Menü "Extras"/"Erweiterte Protokollierung" aktivieren).

In Modulen sollte in den Klassen log4J benutzt werden. Hierzu gibt es in Firstspirit eigene Loggingklassen: de.espirit.common.base.Logging (Developer-Api)



Bestimmte Informationen werden nur in der "erweiterten Protokollierung" angezeigt bzw. ausgegeben und können zum Debugging eines Skripts genutzt werden. Ist das nicht mehr erwünscht, kann die Protokollierung einfach wieder deaktiviert werden.

Tabellen-Verzeichnis

3 Referenzierte Dokumente

Nr.	Dokument	Beschreibung
[1]	DEVB4xDE_FirstSpirit_DeveloperDocumentationBasics.pdf	FirstSpirit Basis Dokumentation für Entwickler

Tabelle 4 Referenzierte Dokumente