FlowProtocol - Anwenderhandbuch

Wolfgang Maier

27. August 2022

Inhaltsverzeichnis

1 Beschreibung und Hintergrund

1.1 Grundidee

FlowProtocol ist eine Anwendung, mit der man Aufgaben-, bzw. Prüflisten zu ausgewählten Themengebieten anhand einfacher Kontrollfragen erstellen kann. Der Name leitet sich ab aus Flow für „Flussdiagramm“ und „Protokoll“ und steht für das Grundkonzept der Anwendung, anhand eines verzweigten Entscheidungsbaums ein Protokoll zusammenzustellen. Die Anwendung funktioniert dabei wie folgt: Nach Auswahl einer Vorlage wird eine Reihe von Multiple-Choice-Fragen gestellt, mit denen die relevanten Aspekte eines Themas Schritt für Schritt eingegrenzt werden, und aus denen am Ende eine genau abgestimmte Auflistung von Punkten zusammengestellt wird, die beispielsweise als Protokoll verwendet werden können.

Die Grundidee der Anwendung besteht darin, ein Medium zu bieten, um Spezialwissen ohne großen Aufwand zu erfassen und es dann in sehr einfacher Form für andere bereitzustellen, ohne dass es dafür vermittelt und erworben werden muss. Die Erstellung der Vorlagen kann und soll direkt durch die Menschen erfolgen, die über das jeweilige Wissen verfügen und die Anwendung für alle möglich sein, die dieses Wissen benötigen.

1.2 Anwendungsbereiche

Die Anwendungsbereiche für diese Anwendung sind sehr vielfältig. Die ursprüngliche Intention war die Aufstellung von Prüflisten für wiederkehrende komplexe Tätigkeiten, die in vielen einzelnen Aspekten variieren können, sodass jeweils nur bestimmte Auswahl von Prüfpunkten relevant ist. Eine Gesamtprüfliste, bei der immer ein Großteil der Punkte hätte ignoriert werden müssen, wäre sowohl aufwendig, als auch fehlerträchtig, und daher nicht effektiv. *FlowProtocol* kann hier durch sein Fragekonzept die für jeden Einzelfall relevanten Aspekte herausfiltern, und auf diese Weise sogar beliebig stark ins Detail gehen. Gleichzeitig hilft dem Anwender die Beantwortung der Fragen auch, sich gedanklich ausführlich mit seiner Aufgabe auseinanderzusetzen. Durch die Vermeidung offener Fragen wird gleichzeitig das Risiko minimiert, dass Dinge vergessen werden. Das Ergebnis ist ein zu 100% relevantes und vollständiges Protokoll.

Ein besonders positiver Nebeneffekt dieser Vorgehensweise besteht darin, dass durch einfaches Auswählen von Antworten am Ende ein Ergebnisdokument entsteht, das nicht mehr manuell erstellt werden muss. Das Anwendungsgebiet dehnt sich damit auf alle Arten von Dokumenten aus, die sich aus einer Aufzählung von Textbausteinen zusammensetzen, deren Zusammenstellung über eine Reihe iterierter Fragen erfolgen kann. Dies umfasst beispielsweise Bestandsaufnahmen, Analyseprotokolle, Zusammenfassungen, Aufgabenlisten und vieles andere mehr. Für kleinere Systeme lassen sich sogar mit überschaubarem Aufwand Vorlagen erstellen, die nicht nur das Ergebnis einer Analyse zusammenfassen, sondern ergänzend dazu auch schon Lösungsansätze und mögliche weitere Schritte. Man kann so vom Prinzip her ein kleines Expertensystem erstellen, das wertvolles Spezialwissen eines Experten in sehr einfacher Form für andere Mitarbeiter verfügbar macht, ähnlich wie ein Chatbot. Das schont wertvolle Unternehmensressourcen und beugt Verfügbarkeitsengpässen vor.

Eine besonders nützliche Anwendung ist die Erstellung von interaktiven Anleitungen. Hierbei werden ebenfalls Rahmenbedigungen und einzelne Textbausteine im Vorfeld abgefragt, und in Abhängigkeit davon die für den vorliegenden Fall relevanten Schritte der Anleitung ausgegeben. Die Möglichkeit, kontextbezogene und durch Textersetzungen angepasste Codepassagen in die Ausgabe einzufügen, macht *FlowProtocol* zu einem besonders nützlichen Werkzeug in der Softwareentwicklung, wo derartige wiederkehrende Muster zum Tagesgeschäft gehören. Die Kombination von Anleitung und Codegenerator erhöht nicht nur die Effizient, sondern auch die Qualität.

Auch kleinere personalisierte Umfragen lassen sich über Vorlagen realisieren, wenn man die Teilnehmer zur Rücksendung der Ergebnisse instruiert. Es gibt sogar einen speziellen Befehl, der bei der Rangfolgenbestimmung von kleineren Listen unterstützt, indem die darin vorkommenden Elemente paarweise gegenübergestellt werden (siehe Abschnitt 4.1). Für wichtige Rangfolgen wie beispielsweise Projektprioritäten ist es natürlich besser, man hat detaillierte und bewährte Kriterien, die sich objektiv auf alle Elemente der Liste anwenden lassen, und die am Ende über einen Zahlenwert zu einer Rangfolge oder Auswahl führen. Auch solche Bewertungen lassen sich mit *FlowProtocol* sehr leicht umsetzen und für alle verfügbar machen.

1.3 Konfiguration

FlowProtocol ist bewusst einfach gehalten und soll es auch bleiben. Als Web-Anwendung kann der Dienst innerhalb einer Einrichtung zentral zur Verfügung gestellt werden. Die einzige notwendige Konfiguration ist die Angabe eines serverseitig verfügbaren Vorlagenordners, in dem sich die Vorlagen befinden (einstellbar über die Eigenschaft TemplatePath in der Datei appsettings.json). Es werden weder Datenbank noch zusätzliche Dienste benötigt. Die Vorlage-Dateien sind normale Textdateien (in UTF-8-Codierung), die die eigentliche Logik enthalten, und die über die Anwendung ausgeführt werden. Die Syntax der Vorlage-Dateien ist ebenfalls sehr einfach und bewusst für die manuelle Erstellung in einem Editor vorgesehen. Der Aufbau ist zeilenbasiert und verwendet Einrückung zur Abbildung der Verschachtelung. Die Erstellung einer ersten eigenen Vorlage ist innerhalb von 2 Minuten möglich. Eine ausführliche Beschreibung des Sprachumfangs wird in der Befehlsreferenz gegeben. Das Ergebnis einer Bearbeitung wird am Ende auf dem Bildschirm ausgegeben, von wo aus es in die Zwischenablage übernommen werden kann. Tatsächlich wird in den meisten Fällen eine Weiterverarbeitung in einem anderen System (E-Mail, CRM, Projektverwaltung) erfolgen, wo noch Begriffe ergänzt und Prozesse angestoßen werden können und eine revisionssichere Verwaltung möglich ist.

Die Organisation der Vorlagen erfolgt auf zwei Ebenen. Auf unterster Ebene ist eine Unterteilung in Anwendergruppen vorgesehen, die jeweils ihre eigenen Vorlagen verwenden. In einem Unternehmen oder einer Einrichtung wird sich diese Aufteilung normalerweise am Mitarbeiterorganigramm orientieren. Ein Beispiel wäre die Aufteilung in Vertrieb, Marketing, Entwicklung, Administration. Jeder Ordner im Vorlagenordner wird in *FlowProtocol* als Anwendergruppe auf der Seite Anwendergruppen angezeigt. Innerhalb dieser Ordner kann man die Vorlagen entweder direkt ablegen oder in weitere Unterordner unterteilen, um fachliche Gruppen zu bilden. Der Pfad einer Vorlage innerhalb dieser Struktur findet sich auch in der URL wie-

der, die innerhalb der Anwendung erzeugt wird, und kann so auch direkt als Link in einer E-Mail, einem Vorgang, im Wiki oder an einer anderen Stelle im Intranet bereitgestellt werden. Der Aufruf führt somit sofort zur Ausführung der gewünschten Vorlage, ohne dass diese über das Menü ausgewählt werden muss.

Eine weitere Aufteilung kann innerhalb einer Vorlage realisiert werden, z.B. indem man dort auf oberster Ebene mit einem Menü oder einer Einstiegsfrage „Was möchten Sie tun?“beginnt. Für jede Auswahl kann wieder praktisch eine eigene Vorlage ausgeführt werden. Auch hier wird die Menüauswahl in der URL verwaltet und kann über Links oder Lesezeichen übermittelt oder gespeichert werden.

1.4 Technische Ergänzungen

FlowProtocol basiert auf den .NET-Framework 6.01 und kann sowohl unter Windows, als auch Linux ausgeführt werden. Der Umgang mit Dateipfaden und Zeilenumbruchzeichen wurde für beide Systeme kompatibel gehalten. Die Anwendung arbeitet vollständig zustandslos in dem Sinne, dass bei der Benutzung keinerlei Daten durch die Anwendung gespeichert werden. Es gibt weder eine Datenbank, noch eine Benutzerverwaltung und der Zugriff auf das Dateisystem erfolgt nur lesend. Die Verwaltung der gegebenen Antworten erfolgt vollständig in der URL, was die Möglichkeit bietet, zurückzuspringen oder einen Zwischenstand als Lesezeichen zu speichern oder zu versenden, allerdings werden Aufrufe von URLs in Unternehmen teilweise durch die IT-Infrastruktur protokolliert, sodass eine Verarbeitung schützenswerter Daten auf jeden Fall dahingehend betrachtet werden sollte.

FlowProtocol steht unter der MIT Lizenz und ist unter

```
https://github.com/maier-san/FlowProtocol
```

frei verfügbar.

Viel Freude beim Erstellen von Vorlagen und deren Anwendung!

2 Befehlsreferenz

2.1 Grundlagen des Dateiaufbaus

Vorlagendateien können in einem beliebigen Texteditor erstellt werden. Die sehr einfache Sprache verlangt weder Syntax-Hervorhebung, noch Auto-Ergänzung, wobei letzteres bei dem von mir für das Windows-Betriebssystem empfohlenen Editor Notepad++ schon in einem sehr komfortablen Maße gegeben ist. Als Dateiendung für eine Vorlagendatei muss „.qfp“ (Quick Flow Protocol) gewählt werden, damit die Datei über die Anwendung aufrufbar ist.

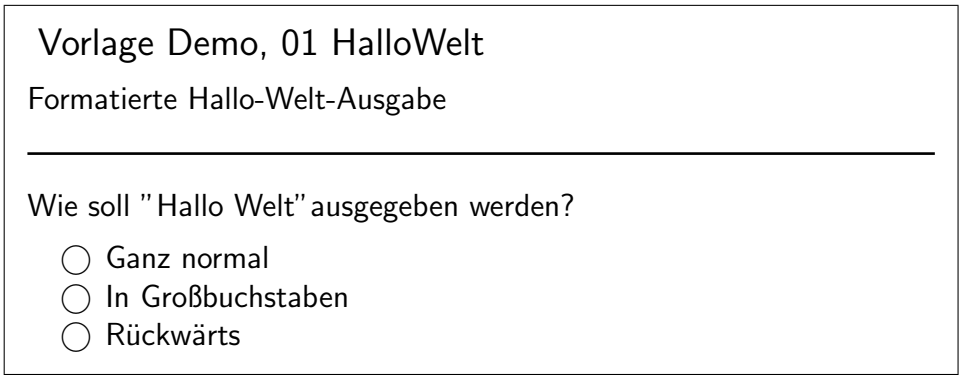
Im Allgemeinen wird die Groß-Klein-Schreibung berücksichtigt, z.B. bei Schlüsseln, Befehlen und Variablen. Bei der Verwendung von Text mit Umlauten sollte darauf geachtet werden, dass diese innerhalb der Anwendung korrekt dargestellt werden. In meiner Umgebung war dies mit der Kodierung UFT-8 der Fall.

Die Verschachtelung der Struktur wird durch Einrückung abgebildet. Hier kann wahlweise mit Tabulator- oder Leerzeichen gearbeitet werden, jedoch sollte dies einheitlich geschehen, da die Struktur bei einer Durchmischung unter Umständen nicht mehr korrekt aufgelöst wird. Die Anwendung ersetzt Tabulatorzeichen durch vier Leerzeichen, unabhängig von ihrer Position. Die Interpretation einer Datei erfolgt zeilenweise, d.h. jeder Zeilenumbruch schließt einen Befehl ab. Der Einsatz von Zeilenumbrüchen zur Formatierung ist damit nicht möglich. Leerzeilen werden bei der Verarbeitung ignoriert und können in beliebiger Menge eingefügt werden.

Beispiel 1

```
/// Formatierte Hallo-Welt-Ausgabe
//  Anwendungsbeispiel für die Befehlsreferenz
?F1: Wie soll "Hallo Welt" ausgegeben werden?
    #a1: Ganz normal
        >> Hallo Welt
    #a2: In Großbuchstaben
        >> HALLO WELT
    #a3: Rückwärts
        >> tleW ollaH
```

Diese Vorlage führt zu folgender Ausgabe:



Der Anwender kann und eine der drei Antwortmöglichkeiten wählen, und die Bearbeitung mit der Weiter-Taste fortsetzen, bzw. in diesem Fall abschließen.

Wenn man zum Beispiel die zweite Antwortmöglichkeit „In Großbuchstaben“, so sieht das Ergebnis wie folgt aus:

2.2 Beschreibungstexte

```
/// <Beschreibungstext>
```

Diese Angabe ist nur in der äußersten Ebene einer Vorlagendatei wirksam. Es können mehrere Zeilen dieser Art angegeben werden.

Der angegebene Beschreibungstext wird bei der Darstellung von Fragen im Kopfbereich ausgegeben. Damit lässt sich einem oder mehreren Sätzen kurz beschreiben, wozu die Vorlage dient, zu welchem Zweck sie eingesetzt werden kann.

2.3 Kommentare

Syntax

```
// <Kommentar>
```

Diese Kommentare werden bei der Verarbeitung durch die Anwendung vollständig ignoriert und dienen ausschließlich zur Kommentierung des Vorlagencodes und werden bei der Ausführung ignoriert.

2.4 Frage, Antworten und Ausgaben

Syntax

```
?<Frageschlüssel>: <Fragetext>
    #<Schlüssel Antwortmöglichkeit 1>: <Text Antwortmöglichkeit 1>
        >> <Text Ausgabe 1>
    #<Schlüssel Antwortmöglichkeit 2>: <Text Antwortmöglichkeit 2>
        >> <Text Ausgabe 2>
    . . .
```

Das Kernelement der Vorlagen sind Fragen. Diese bestehen aus einem Frageschlüssel und einem Frage Text. Die Schlüssel für Fragen und Antwortmöglichkeiten werden verwendet, um die gegebenen Antworten innerhalb der URL zu verwalten. Sie dürfen nur aus Buchstaben und Zahlen bestehen und sollten möglichst kurz gewählt werden. Die Schlüssel der Fragen müssen über die komplette Vorlage eindeutig sein, die der Antworten nur innerhalb einer Frage.

An der Oberfläche dargestellt wird nur der Fragetext und die Texte der Antwortmöglichkeiten.

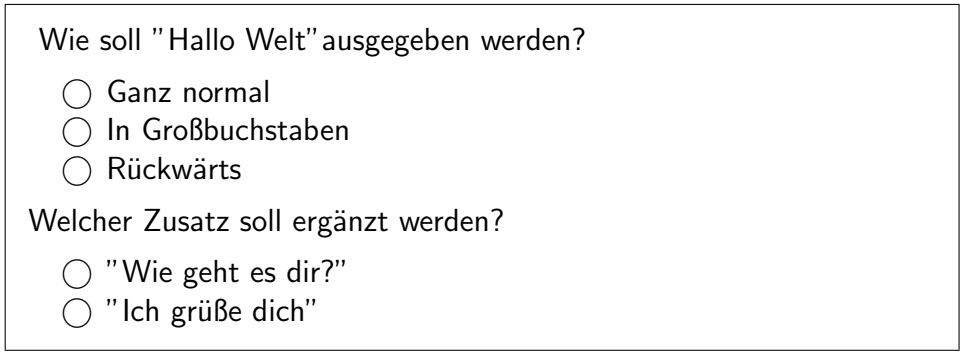
Durch `>>` wird ein Ausgabeeintrag erzeugt, wenn die entsprechende Antwortmöglichkeit gewählt wurde. Die Ausgaben werden als nummerierte Liste zu einem Ergebnis zusammengefasst.

2.5 Sequenzen

Mehrere Fragen können hintereinander aufgelistet werden. Diese werden standardmäßig untereinander auf einer Seite aufgelistet und können zusammen bearbeitet werden.

Beispiel 2

```
/// Formatierte Hallo-Welt-Ausgabe
//  Anwendungsbeispiel für die Befehlsreferenz
?F1: Wie soll "Hallo Welt" ausgegeben werden?
    #a1: Ganz normal
        >> Hallo Welt
    #a2: In Großbuchstaben
        >> HALLO WELT
    #a3: Rückwärts
        >> tleW ollaH
?F2: Welcher Zusatz soll ergänzt werden?
    #z1: "Wie geht es dir?"
        >> Wie geht es dir?
    #z2: "Ich grüße dich!"
        >> Ich grüße dich!
```



Die Benutzerführung erlaubt es, auch nur einen Teil der angezeigten Fragen zu beantworten und dann die Weiter-Schaltfläche zu betätigen. In diesem Fall werden die nicht beantworteten Fragen einfach auf der Folgeseite erneut angezeigt.

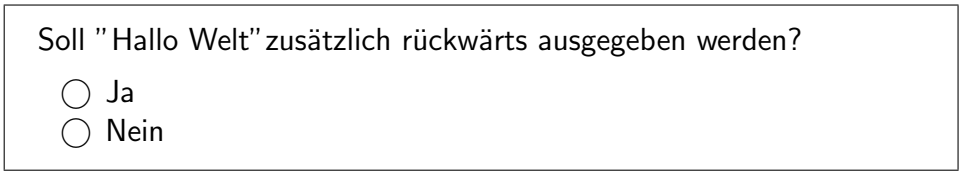
2.6 Verschachtelungen

Beispiel 3

```
/// Formatierte Hallo-Welt-Ausgabe
//  Anwendungsbeispiel für die Befehlsreferenz
?F1: Wie soll "Hallo Welt" ausgegeben werden?
    #a1: Ganz normal
        >> Hallo Welt
    #a2: In Großbuchstaben
        ?F1a: Text zusätzlich rückwärts ausgeben?
            #j: Ja
```

```
            >> TLEW OLLAH
        #n: Nein
            >> HALLO WELT
#a3: Rückwärts
    >> tleW ollaH
```

Die Darstellung ist zunächst identisch mit der des Ausgangsbeispiels. Die innere Frage wird erst und nur dann angezeigt, wenn in der ersten Ebene die Antwortmöglichkeit „In Großbuchstaben“ gewählt wird.



2.7 Gruppierungen

Das Ergebnis einer Bearbeitung kann in vielen Fällen mehr als eine Liste sein. Eine Vorlage zur Analyse kann beispielsweise zum einen den ist-Stand erfassen, mögliche Lösungsansätze formulieren und die noch ausstehenden Analyse-Maßnahmen Auflisten. Zu diesem Zweck können die Ausgaben in einer sehr einfachen Form kopiert werden.

Syntax

```
>> <Gruppe> >> <Text Ausgabe>
```

Beispiel 4

```
/// Formatierte Hallo-Welt-Ausgabe
//  Anwendungsbeispiel für die Befehlsreferenz
?F1: Wie soll "Hallo Welt" ausgegeben werden?
    #a1: Ganz normal
        >> Deine Ausgabe >> Hallo Welt
        >> Die Alternativen >> HALLO WELT
        >> Die Alternativen >> tleW ollaH
    #a2: In Großbuchstaben
        >> Deine Ausgabe >> HALLO WELT
        >> Die Alternativen >> Hallo Welt
        >> Die Alternativen >> tleW ollaH
    #a3: Rückwärts
        >> Deine Ausgabe >> tleW ollaH
        >> Die Alternativen >> Hallo Welt
        >> Die Alternativen >> HALLO WELT
```

Wählt man die Antwortmöglichkeit „In Großbuchstaben“, so erhält man die folgende Ausgabe:

Ergebnisliste (Demo, 05 Gruppierungen)

Deine Ausgabe

1 HALLO WELT

Die Alternativen

1 Hallo Welt
2 tleW ollaH

2.8 Unterpunkte

Ausgaben können je nach Vorlage selbst wieder Tätigkeiten oder Aufgaben beschreiben, die sich aus verschiedenen Schritten zusammensetzen, oder bei denen mehrere Teilaspekte zu berücksichtigen sind. Diese lassen sich in Form von Unterpunkten Auflisten. Ein Unterpunkt, der mit der Protokollkennung „https://“ beginnt und als Link erkannt wird, wird dabei in der Ausgabe als ausführbarer Link dargestellt.

Syntax

```
>> <Ausgabe Text>
    > <Unterpunkt Text 1>
    > <Unterpunkt Text 2>
    . . .
```

Beispiel 5

```
/// Formatierte Hallo-Welt-Ausgabe
//  Anwendungsbeispiel für die Befehlsreferenz
?F1: Wie soll "Hallo Welt" ausgegeben werden?
    #a1: Ganz normal
        >> Hallo Welt
    #a2: In Großbuchstaben
        >> HALLO WELT
            > Mehr Infos findest du hier:
            > https://github.com/maier-san/FlowProtocol
    #a3: Rückwärts
        >> tleW ollaH
```

Wählt man die Antwortmöglichkeit „In Großbuchstaben“, so erhält man die folgende Ausgabe:

Ergebnisliste (Demo, 06 Unterpunkte)

1 HALLO WELT

- Mehr Infos findest du hier:
- `https://github.com/maier-san/FlowProtocol`

2.9 Wiederholungen

Eine Frage kann an anderer Stelle wiederholt gestellt werden. Ausschlaggebend dafür ist, dass der Schlüssel der Frage identisch mit einem bereits verwendeten Schlüssel ist. Ist die Antwort auf die Frage bei der Bearbeitung zu diesem Zeitpunkt bereits gegeben, so wird die Fragestellung nicht mehr angezeigt und direkt der jeweilige Antwortzweig verarbeitet. Wenn das aufgrund der Anordnung zwingend der Fall ist, können die Texte abgekürzt werden, da sie ja nicht ausgegeben werden. In diesem Fall empfiehlt es sich zur Verbesserung der Übersichtlichkeit, in den Texten kenntlich zu machen, dass es sich um eine Wiederholung handelt.

Beispiel 6

```
/// Formatierte Hallo-Welt-Ausgabe
//  Anwendungsbeispiel für die Befehlsreferenz
?F1: Wie soll "Hallo Welt" ausgegeben werden?
    #a1: Ganz normal
        >> Hallo Welt
    #a2: In Großbuchstaben
        >> HALLO WELT
    #a3: Rückwärts
        >> tleW ollaH
?F2: Welcher Zusatz soll ergänzt werden?
    #z1: "Wie geht es dir?"
        ?F1: Wdh. Wie soll "Hallo Welt" ausgegeben werden?
            #a2: In Großbuchstaben
                >> WIE GEHT ES DIR?
            #x: sonst
                >> Wie geht es dir?
    #z2: "Ich grüße dich!"
        >> Ich grüße dich!
```

Bei der Wiederholung einer Frage können auch nur Teile der ursprünglichen Antwortmöglichkeiten gegeben werden. Wird darunter die zuerst gegebene Antwort nicht gefunden, so wird keiner der Antwortzweige verarbeitet. Zur Vereinfachung kann jedoch mit dem Antwortschlüssel `#x` eine Antwortmöglichkeit gegeben werden, die stellvertretend für alle anderen Antwortmöglichkeiten steht. Wird im obigen Beispiel also für F1 die Antwortmöglichkeit `#a3` gewählt, so wird diese mit der Wiederholung der Frage dem Antwortschlüssel `#x` zugeordnet.

2.10 Implikationen

Die Antwort auf eine Frage kann in manchen Fällen die Antwort auf eine andere implizieren. Dies kann mit dem `~Implies`-Befehl abgebildet werden.

Syntax

```
~Implies <Frageschlüssel 1>=<Antwortschlüssel 1>;  . . .
```

Beispiel 7

```
/// Formatierte Hallo-Welt-Ausgabe
//  Anwendungsbeispiel für die Befehlsreferenz
?F1: Wie soll "Hallo Welt" ausgegeben werden?
    #a1: Ganz normal
        >> Hallo Welt
    #a2: In Großbuchstaben
        >> HALLO WELT
        ~Implies F2a=j
    #a3: Rückwärts
        >> tleW ollaH
?F2: Welcher Zusatz soll ergänzt werden?
    #z1: "Wie geht es dir?"
        ?F2a: Soll der Zusatz mit Sternchen ausgegeben werden?
            #j: Ja
                >> *** Wie geht es dir? ***
            #n: Nein
                >> Wie geht es dir?
    #z2: "Ich grüße dich!"
        >> Ich grüße dich!
```

Bei Auswahl der Antwort „In Großbuchstaben“ Frage `F1` wird für die Frage `F2a` die Antwort „Ja“ impliziert. Diese wird dann nicht mehr gestellt, sondern automatisch beantwortet. Im Falle der anderen Antwortmöglichkeiten wird keine Antwort impliziert, sodass die Frage `F2a` in diesem Fall ganz normal gestellt und verarbeitet wird.

2.11 Autonummerierung

Das Durchnummerieren der Frageschlüssel kann in umfangreichen Dateien sehr mühsam werden, da diese zumeist nicht von oben nach unten aufgebaut werden, sondern in die Tiefe. Die Kontrolle über die Schlüssel benötigt man aber nur, wenn man mit Implikationen oder Wiederholungen arbeitet, oder die Gültigkeit von Links auch nach Erweiterungen der Vorlage sicherstellen möchte. Für alle anderen Fälle kann man die Nummerierung automatisch vergeben lassen.

Syntax

```
?<Frageschlüsselanfang>': <Fragetext>
```

Beispiel 8

```
/// Formatierte Hallo-Welt-Ausgabe
//  Anwendungsbeispiel für die Befehlsreferenz
?F': Wie soll "Hallo Welt" ausgegeben werden?
    #a1: Ganz normal
        >> Hallo Welt
```

```
    #a2: In Großbuchstaben
        >> HALLO WELT
?F': Welcher Zusatz soll ergänzt werden?
    #z1: "Wie geht es dir?"
        >> Wie geht es dir?
    #z2: "Ich grüße dich!"
        >> Ich grüße dich!
```

Die Apostrophzeichen in den Schlüsseln werden pro Datei durchnummeriert und im obigen Beispiel zu `F_1` und `F_2` ersetzt. Schlüssel ohne das `'`-Zeichen bleiben unverändert und können für Implikationen oder Wiederholungen genutzt werden. Implikationen für automatisch nummerierte Schlüssel sind zwar ebenfalls möglich, aber aufgrund der fehlenden Transparenz keine gute Idee. Die Nummerierung erfolgt pro Datei, was bei der Verwendung von Funktionen (siehe Abschnitt 2.13) berücksichtigt werden muss.

2.12 Ausführungen

Bei längeren Sequenzen und insbesondere bei der Verwendung von Wiederholungen und Implikationen ist es wünschenswert, zunächst einen Teil der Fragen vollständig abzuarbeiten, bevor die Fragen aus dem nachfolgenden Teil angezeigt werden. Dies ist mit dem `~Execute`-Befehl möglich.

Syntax

```
~Execute
```

Beispiel 9

```
/// Formatierte Hallo-Welt-Ausgabe
//  Anwendungsbeispiel für die Befehlsreferenz
?F1: Wie soll "Hallo Welt" ausgegeben werden?
    #a1: Ganz normal
        >> Hallo Welt
    #a2: In Großbuchstaben
        >> HALLO WELT
    #a3: Rückwärts
        >> tleW ollaH
~Execute
?F2: Welcher Zusatz soll ergänzt werden?
    #z1: "Wie geht es dir?"
        >> Wie geht es dir?
    #z2: "Ich grüße dich!"
        >> Ich grüße dich!
```

Die beiden Fragen aus dem Sequenzen-Beispiel werden in diesem Fall einzelnen angezeigt.

2.13 Funktionen

Vorlagen können mitunter sehr umfangreich und tief verschachtelt sein. Verwaltet man solche Vorlagen in einer einzigen Datei, kann das insbesondere aufgrund der Einrückungen sehr unübersichtlich werden, sodass sich Anpassungen und Erweiterungen schwierig gestalten. Hier ist es empfehlenswert Teile des Codes als Funktionen in jeweils eigene Dateien auszulagern. Eine Funktion ist vom Prinzip und vom Aufbau her das gleiche wie eine Vorlage und wird über eine Datei der Endung „*.qff“ (Quick Flow Function) bereitgestellt. Aufgrund der abweichenden Dateiendung werden Funktionen nicht als Vorlagen in der Anwendung zur Auswahl angeboten.

Die Funktionsdatei muss auf der gleichen Ebene liegen wie die Vorlage, die sie aufruft und der Name darf nur aus Buchstaben und Ziffern bestehen.

Syntax

```
~Include <Name der Funktion>
```

Beispiel 10

```
/// Formatierte Hallo-Welt-Ausgabe
//  Anwendungsbeispiel für die Befehlsreferenz
~Include HalloWelt1
```

Mit der Funktionsdatei HalloWelt1.qff

```
//  Hallo-Welt-Funktion
?F1: Wie soll "Hallo Welt" ausgegeben werden?
    #a1: Ganz normal
        >> Hallo Welt
    #a2: In Großbuchstaben
        >> HALLO WELT
    #a3: Rückwärts
        >> tleW ollaH
```

Neben der Erhöhung der Übersichtlichkeit hat die Verwendung von Funktionen viele weitere Vorteile. Auf technischer Ebene vereinfacht es die Verarbeitung, da eine Funktion erst dann geladen wird, wenn der entsprechende Pfad auch tatsächlich erreicht wird. Aus Anwendersicht ergibt sich der große Vorteil, dass man eine Funktion in verschiedenen Antwortzweigen aufrufen kann, und damit keinen doppelten Code pflegen muss. Entsprechend lassen sich Funktionen auch aus verschiedenen Vorlagen heraus aufrufen, sodass fachliche Einheiten ausgelagert und mehrfach wieder verwendet werden können.

2.14 Parametrisierte Funktionen

Möchte man eine Funktion innerhalb einer Vorlage mehrfach ausführen, so stößt man auf das Problem, dass die dort eingetragenen Frage-Schlüssel für beide Aufrufe identisch sind, und so der zweite Aufruf automatisch die Antworten des ersten Aufrufs übernehmen würde. Zudem wäre die Anzeige der Fragen identisch, sodass man

diese beim Beantworten nicht den jeweiligen Aufrufen zuordnen könnte. Und dieses Problem zu lösen, lassen sich Aufrufe parametrisieren.

Syntax

```
~Include <Name der Funktion> <Parameter 1>=<Wert 1>; . . .
```

Beispiel 11

```
/// Formatierte Hallo-Welt-Ausgabe
//  Anwendungsbeispiel für die Befehlsreferenz
~Include HalloWelt2 weltindex=1; weltbezeichnung=Meine Welt
~Include HalloWelt2 weltindex=2; weltbezeichnung=Deine Welt
```

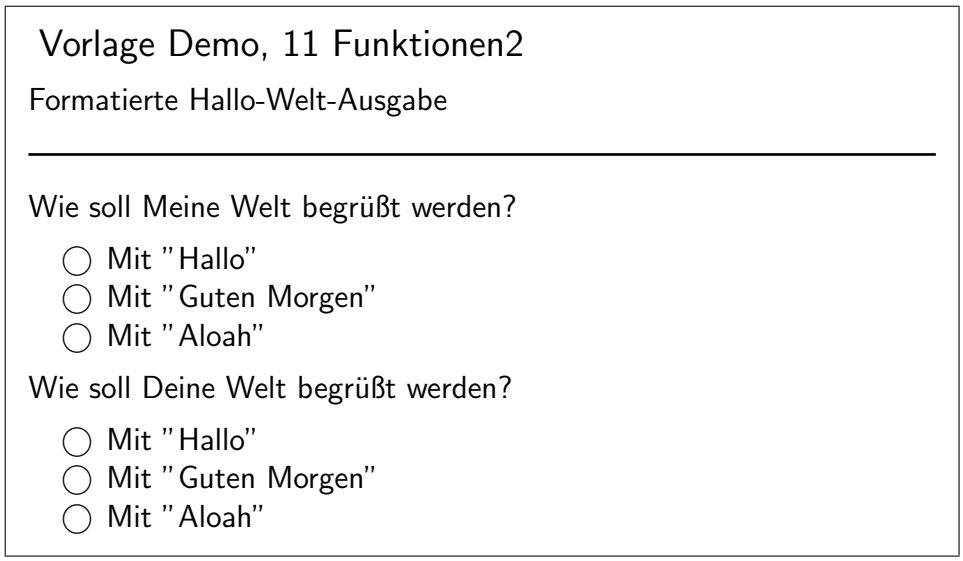
Mit der Funktionsdatei HalloWelt2.qff

```
// Hallo-Welt-Funktion
// Parameter $weltindex: Schlüsselindex
// Parameter $weltbezeichnung: Weltbezeichnung
?F$weltindex1: Wie soll $weltbezeichnung begrüßt werden?
    #a1: Mit "Hallo"
        >> Hallo $weltbezeichnung
    #a2: Mit "Guten Morgen"
        >> Guten Morgen $weltbezeichnung
    #a3: Mit Aloah
        >> Aloah $weltbezeichnung
```

Beim Aufrufen der Funktion HalloWert2 werden in jeder Zeile die mit dem $-Zeichen gekennzeichneten Variablen durch die beim Aufruf zugeordneten Werte ersetzt. Da auch in den Frageschlüsseln die Variable `$weltindex` integriert wurde, ergeben sich bei den beiden aufrufen unterschiedliche Frageschlüssel `F11` und `F21`, über die jeweils eigene Antworten verwaltet werden können. Die Texte der Fragen enthalten ebenfalls Variablen und sind damit klar zuzuordnen.

Es wird empfohlen die Parameter einer Funktion in Form von Kommentaren im Kopfbereich zu beschreiben.

Das obige Beispiel wird wie folgt angezeigt:



Eine eindeutige Benennung der Schlüssel wäre auch automatisch möglich gewesen, jedoch auf Kosten der Möglichkeit, innerhalb einer Funktion wahlweise auch auf Fragen der aufrufenden Stelle zuzugreifen. Deshalb wurde hierauf verzichtet.

2.15 Variablen

Variablen können auch unabhängig von Funktionsparametern gesetzt und in Fragen, Antworten und Ausgaben verwendet werden.

Syntax

```
~Set <Variable 1>=<Wert 1>; <Variable 2>=<Wert 2>; . . .
```

Beispiel 12

```
/// Formatierte Hallo-Welt-Ausgabe
//  Anwendungsbeispiel für die Befehlsreferenz
?F1: Wie soll "Hallo Welt" ausgegeben werden?
    #a1: Ganz normal
        >> Hallo Welt
        ~Set Preis=5 Euro
    #a2: In Großbuchstaben
        >> HALLO WELT
        ~Set Preis=8 Euro
    #a3: Rückwärts
        >> tleW ollaH
        ~Set Preis=12 Euro
~Execute
?F2: Soll der Preis von $Preis ausgegeben werden?
    #j: Ja
        >> Der Preis beträgt $Preis.
    #n: Nein
```

Wird eine Variable in einer Frage oder Antwort verwendet, sollte durch die Struktur sichergestellt werden, dass die Frage, die für das Setzen der Variablen zuständig ist, zuvor ausgeführt wurde. Dies kann z.B. mit dem oben beschriebenen `~Execute`-Befehl geschehen.

2.16 Additionen

Über Variablen lassen sich auch einfache Additionen durchführen, beispielsweise um Kosten, Aufwände oder Komplexität zu ermitteln, die sich aus den gewählten Antwortmöglichkeiten zusammensetzen. Dies ist ebenfalls über den `~Set`-Befehl möglich.

Syntax

```
~Set <Variable 1>+=<Inkrement 1>;  . . .
```

Additionen und Variablenzuweisungen können auch im gleichen `~Set`-Befehl gemischt werden. Bei ihrer ersten Verwendung in einer Addition wird der Variablen zuvor ein Wert von 0 zugewiesen. Hat eine Variable bei der Verwendung in einer Addition bereits einen Wert, der nicht als natürliche Zahl interpretierbar ist, so wird die Addition ignoriert.

Beispiel 13

```
/// Formatierte Hallo-Welt-Ausgabe
//  Anwendungsbeispiel für die Befehlsreferenz
?F1: Wie soll "Hallo Welt" ausgegeben werden?
    #a1: Ganz normal
        >> Hallo Welt
        ~Set Preis=5
    #a2: In Großbuchstaben
        >> HALLO WELT
        ~Set Preis=8
    #a3: Rückwärts
        >> tleW ollaH
        ~Set Preis=12
?F2: Welcher Zusatz soll ergänzt werden?
    #z1: "Wie geht es dir?"
        >> Wie geht es dir?
        ~Set Preis+=3
    #z2: "Ich grüße dich!"
        >> Ich grüße dich!
        ~Set Preis+=7
?F3: Soll der Preis ausgegeben werden?
    #j: Ja
        >> Der Preis beträgt $Preis Euro.
    #n: Nein
```

2.17 Schleifen

In machen Fällen möchte man eine Funktion wiederholt für eine beliebige Menge an Elementen aufrufen. Dies ist möglich, indem man das Hochzählen von Variablen mit den Parameterzuweisungen eines rekursiven Funktionsaufrufes kombiniert.

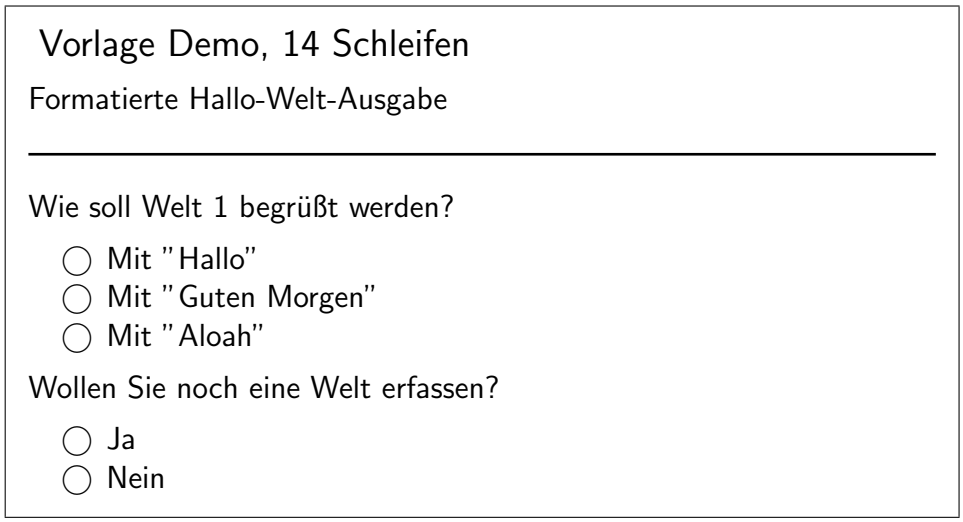
Beispiel 14

```
/// Formatierte Hallo-Welt-Ausgabe
//  Anwendungsbeispiel für die Befehlsreferenz
~Include HalloWelt3 weltindex=1
```

Mit der Funktionsdatei HalloWelt3.qff

```
// Hallo-Welt-Funktion
// Parameter $weltindex: Schlüsselindex
?F$weltindex1: Wie soll Welt $weltindex begrüßt werden?
    #a1: Mit "Hallo"
        >> Hallo Welt $weltindex
    #a2: Mit "Guten Morgen"
        >> Guten Morgen Welt $weltindex
    #a3: Mit Aloah
        >> Aloah Welt $weltindex
?F$weltindex2: Wollen Sie noch eine Welt erfassen?
    #j: Ja
        ~Set dummy=$weltindex; dummy+=1
        ~Include HalloWelt3 weltindex=$dummy
    #n: Nein
```

Man beachte, dass Additionen wie im `~Set`-Befehl in den Zuweisungen des `~Include`-Befehls selbst nicht möglich sind, da Include-Parameter anders verarbeitet werden als Variablen.
Das obige Beispiel wird wie folgt angezeigt:



Wählt man nacheinander Antworten und bestätigt die zweite Frage in den ersten beiden Durchläufen mit ja, bekommt man ein Ergebnis wie folgt:

Ergebnisliste (Demo, 13 Schleifen)

1 Guten Morgen Welt 1
2 Hallo Welt 2
3 Aloah Welt 3

2.18 Eingaben

In manchen Fällen kann es sinnvoll sein, kurze Texte oder einzelne Begriffe abzufragen, um diese als Bestandteil der Ausgabe zu verwenden. Insbesondere wenn man über eine Schleife eine Sequenz von Elementen abfragt, die man in der Ausgabe nicht nur als nummerierte, sondern benannte Aufzählung haben möchte. Dafür gibt es den `~Input`-Befehl.

Syntax

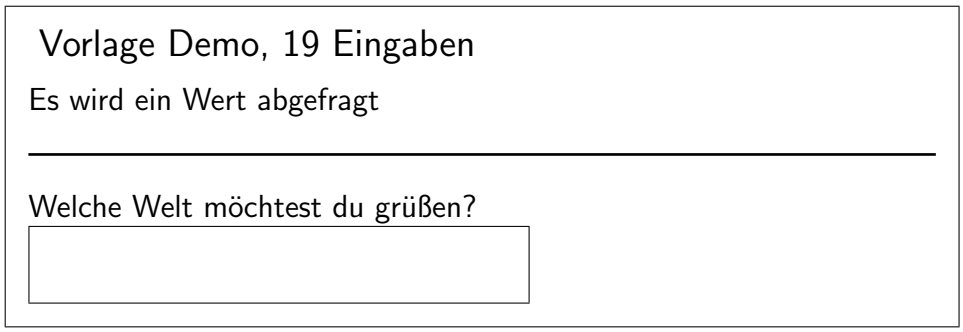
```
/// Formatierte Hallo-Welt-Ausgabe
//  Anwendungsbeispiel für die Befehlsreferenz
~Input  <Eingabeschlüssel>: <Fragetext>
```

Der Schlüssel dient wie bei den Fragen zur Verwaltung der Eingabe in der URL, und der Fragetext wird an der Oberfläche angezeigt. Der Zugriff auf die Eingabe erfolgt wie bei Variablen über den Eingabeschlüssel, indem man das $-Zeichen voranstellt. Eine Eingabe kann ebenso wie die Antwort auf eine Frage mit dem `~Implies`-Befehl impliziert werden. Die Zeichenzahl ist pro Antwort auf maximal 50 Zeichen beschränkt, und aufgrund der beschränkten Zeichenzahl in der URL sollte diese Funktion nicht überstrapaziert werden.

Beispiel 15

```
/// Es wird ein Wert abgefragt
//  Anwendungsbeispiel für die Befehlsreferenz
?F1: Möchtest du eine besondere Welt grüßen?
    #j: Ja
        ~Input F2: Welche Welt möchtest du grüßen?
    #n: Nein
        ~Implies F2=Welt
~Execute
>> Hallo $F2!
```

Wird im obigen Beispiel die Antwort Ja gewählt, so wird im nächsten Schritt die Frage gestellt, welche Welt man grüßen möchte, mit der Möglichkeit einen Text einzugeben. Wählt man Nein, so wird die Frage implizit mit „Welt“ beantwortet.



2.19 Codesequenzen

In der Softwareentwicklung und insbesondere bei der Pflege und Weiterentwicklung großer Systeme gibt es zahlreiche Komponenten und Bausteine, die immer nach dem grundsätzlich selben Muster auf-, oder eingebaut werden. Oft gibt es dabei Varianten, bei denen man den einen oder anderen Aufruf mehr oder weniger benötigt, und die Benennung der Elemente muss ebenfalls angepasst werden. Es kann die Arbeit deutlich vereinfachen, wenn man mit *FlowProtocol* eine Vorlage für eine Anleitung erstellt, die die relevanten Abhängigkeiten abfragt, und dem Entwickler die notwendigen Implementierungsschritte beschreibt. Noch besser ist es, wenn man eine Stufe weiter geht, und die einzelnen Codefragmente direkt durch die Vorlage erstellen lässt, so dass man diese ausschneiden und in den Produktivcode einfügen kann. Hierfür gibt es die Möglichkeit, ganze Codesequenzen ausgeben zu lassen.

Die Syntax ist ähnlich zu der der Unterpunkte (siehe Abschnitt 2.8). Codesequenzen und Unterpunkte lassen sich auch innerhalb desselben Ausgabeeintrags kombinieren, die Ausgabe erfolgt jedoch blockweise, d.h. zuerst die Unterpunkte und dann der Codeblock.

Syntax

```
>> <Ausgabe Text>
    >| <Codezeile 1>
    >| <Codezeile 2>
    . . .
```

Die Ausgabe erfolgt in einer Festbreitenschriftart und es werden auch führende Leerzeichen mit ausgegeben.

Beispiel 16

```
/// Programmierunterstützung
/// Erstellungsroutine für einen Entitätsdatensatz
//  Anwendungsbeispiel für die Befehlsreferenz
~Input Entity: Wie wird die Entität benannt?
?F1: Von welchem Typ ist der Primärschlüssel?
    #i: Int
        ~Set PTyp=int;
    #G: Guid
```

```
        ~Set PTyp=Guid;
~Execute
>> Anleitung >> Implementiere die Funktion Erstelle$Entity()
    >|public $PTyp Erstelle$Entity()
    >|{
    >|   // Rufe generische Funktion auf
    >|   return ErstelleElement<$PTyp>()
    >|}
>> Anleitung >> Ergänze den Aufruf in der Geschäftslogik
    >|   $PTyp id = Erstelle$Entity()
```

Wird im obigen Beispiel als Entitätsbennung „Fahrzeug“ eingegeben und als Typ des Primärschlüssels „int“ gewählt, so bekommt man folgende Ausgabe:

Ergebnisliste (Demo, 21 Codegen)

Anleitung

1 Implementiere die Funktion ErstelleFahrzeug()

```
public int ErstelleFahrzeug()
{
  // Rufe generische Funktion auf
  return ErstelleElement<int>()
}
```

2 Ergänze den Aufruf in der Geschäftslogik

```
int id = ErstelleFahrzeug()
```

3 Fehlermeldungen

Die Möglichkeit, Vorlagen von Hand zu schreiben birgt ein gewisses Fehlerrisiko. Die Fehler werden bei der Ausführung weitestgehend erkannt und im Kopfbereich mit allen notwendigen Informationen angezeigt.

Beispiel 17

```
/// Fehler-Beispiel
//  Anwendungsbeispiel für die Befehlsreferenz
#d: Hallo Fehler!
~Tralala abc
?F1: Ist ein Fehler schlimm?
    #j: Ja
        >> Stimmt, muss man korrigieren.
    #n: Nein
        >> Stimmt, Ausführung geht weiter.
```

Der Aufruf dieser Vorlage führt direkt zur Anzeige der beiden Fehler:

Vorlage Demo, 15 HalloFehler

Fehler-Beispiel

Hinweis: Beim Ausführen der Vorlage sind Fehler aufgetreten. Bitte wenden Sie sich an den Autoren der Vorlage oder einen Administrator, um die Korrekturen zu veranlassen:

1. **Fehler R04, Antwort kann keinem Fragekontext zugeordnet werden.**
   Zeile 3 *FlowProtocol/Templates/Demo/14 HalloFehler.qfp*
   `#d: Hallo Fehler!`
2. **Fehler C02,Der Befehl Tralala ist nicht bekannt und kann nicht ausgeführt werden.**
   Zeile 4 *FlowProtocol/Templates/Demo/14 HalloFehler.qfp*
   `~Tralala abc`

Ist ein Fehler schlimm?

- ◯ Ja
- ◯ Nein

Wie man erkennt, wird der nach dem Fehler stehende Vorlagencode trotz der Fehler interpretiert und ausgeführt, was die Notwendigkeit einer Korrektur natürlich nicht mindert.

3.1 Einlesefehler

Einlesefehler treten direkt beim Einlesen einer Vorlage- oder Funktionsdatei auf. Da Funktionsdateien erst bei Bedarf eingelesen werden, werden auch die darin enthaltenen Fehler erst angezeigt, wenn der entsprechende Zweig durchlaufen wird.

R01 *Vorlagendatei nicht gefunden.*
Kann auftreten, wenn eine als Lesezeichen gespeicherte URL auf eine nicht (mehr) vorhandene Vorlagendatei verweist, oder kein Lesezugriff auf das Verzeichnis besteht.

R02 *Beschreibungskommentar auf untergeordneter Ebene wird ignoriert.*
Tritt auf, wenn ein Beschreibungskommentar nicht auf oberster Ebene der Vorlage angegeben ist. Dieser wird ignoriert.

R03 *Frage kann keinem Kontext zugeordnet werden.*
Tritt auf, wenn eine Frage keinem Vorlagenkontext zugeordnet werden kann, wenn also die Struktur aus Fragen und Antworten unstimmig ist. Ursache kann eine fehlerhafte Einrückung sein, z.B. weil dabei Leerzeichen und Tabulatorzeichen gemischt wurden.

R04 *Antwort kann keinem Fragekontext zugeordnet werden.*
Tritt auf, wenn eine Antwort keinem Fragekontext zugeordnet werden kann, wenn also die Struktur aus Fragen und Antworten unstimmig ist. Ursache kann eine fehlerhafte Einrückung sein, z.B. weil dabei Leerzeichen und Tabulatorzeichen gemischt wurden.

R05 *Gruppierter Ausgabeeintrag kann keinem Kontext zugeordnet werden.*
Tritt auf, wenn ein gruppierter Ausgabeeintrag keinem Vorlagenkontext zugeordnet werden kann. Mögliche Ursachen siehe Fehler R03.

R06 *Ausgabeeintrag kann keinem Kontext zugeordnet werden.*
Tritt auf, wenn ein nicht gruppierter Ausgabeeintrag keinem Vorlagenkontext zugeordnet werden kann. Mögliche Ursachen siehe Fehler R03.

R07 *Unterpunkt kann keinem Ausgabeeintrag zugeordnet werden.*
Tritt auf, wenn ein Unterpunkt keinem Ausgabeeintrag zugeordnet werden kann. Unterpunkte müssen immer unmittelbar auf Ausgabeeinträge oder andere Unterpunkte oder Codezeilen folgen und dürfen nicht durch Fragen und Antworten unterbrochen werden.

R08 *Execute-Befehl kann keinem Kontext zugeordnet werden.*
Tritt auf, wenn der Execute-Befehl keinem Vorlagenkontext zugeordnet werden kann. Mögliche Ursachen siehe Fehler R03.

R09 *Befehl kann keinem Kontext zugeordnet werden.*
Tritt auf, wenn ein Befehl keinem Vorlagenkontext zugeordnet werden kann. Mögliche Ursachen siehe Fehler R03.

R10 *Zeile nicht interpretierbar*
Tritt auf, wenn eine Zeile nicht als Anweisung interpretiert werden kann, sie also keiner innerhalb einer Vorlage bekannten Syntax entspricht.

R11 *Input-Befehl kann keinem Kontext zugeordnet werden.*
Tritt auf, wenn ein Input-Befehl keinem Vorlagenkontext zugeordnet werden kann. Mögliche Ursachen siehe Fehler R03.

R12 *Codezeile kann keinem Ausgabeeintrag zugeordnet werden.*
Tritt auf, wenn eine Codezeile keinem Ausgabeeintrag zugeordnet werden kann. Codezeilen müssen immer unmittelbar auf Ausgabeeinträge oder Unterpunkte oder andere Codezeilen folgen und dürfen nicht durch Fragen und Antworten unterbrochen werden.

3.2 Ausführungsfehler

Ausführungsfehler treten im Zusammenhang mit der Ausführung von Befehlen auf und sind teilweise vom Ausführungsverlauf abhängig. Auch sie werden erst dann angezeigt, wenn der entsprechende Zweig durchlaufen wird.

C00 *Beim Ausführen eines Befehls ist ein unbehandelter Fehler aufgetreten.*
Tritt auf, wenn bei der Ausführung ein Fehler auftritt, der durch die Anwendung nicht weiter eingegrenzt werden konnte.

C02 *Der Befehl ... ist nicht bekannt und kann nicht ausgeführt werden.*
Tritt auf, wenn eine Befehlsanweisung keinem bekannten Befehl entspricht. Eine Abweichung in der Groß-Klein-Schreibung kann als Ursache ausgeschlossen werden.

C03 *Die Funktionsdatei ... konnte nicht geladen werden.*
Tritt auf, wenn in einem Include-Befehl angegebene Funktionsdatei nicht gefunden werden konnte.

C04 *Der Wert der Variablen ... konnte nicht als ganze Zahl interpretiert werden.*
Tritt auf, wenn für eine Variable im Set-Befehl ein Wert addiert werden soll, und diese zum entsprechenden Zeitpunkt einen nichtnumerischen Wert enthält.

C05 *Der Aufruf der Funktionsdatei ... überschreitet das Rekursionsmaximum von 100.*
Tritt auf, wenn mehr als 100 Mal eine Funktionsdatei aufgerufen wird. Diese einfache Sicherung verhindert Endlosrekursionen, die zum Absturz der Anwendung führen würden.

4 Spezialbefehle

Spezialbefehle sind spezielle Befehle, die die Struktur einer Vorlage verändern, und die darauf erfolgten Eingaben in eigener Weise interpretieren und zu einem Ergebnis umsetzen können. Diese Befehle wirken innerhalb des Vorlageteils, wo sie stehen, also auf einer Verschachtelungsebene bis zu den jeweils angrenzenden `~Execute`-Befehlen.

Befehlsübergreifende Fehlermeldungen

S01 *Befehl ohne ...-Argument.*
Tritt auf, wenn das zwingende ...-Argument nicht angegeben wurde.

S02 *Frageschlüssel ... (...-Argument) nicht gefunden.*
Tritt auf, wenn der über das ...-Argument angegebene Frageschlüssel nicht gefunden wurde.

S03 *Befehl ohne gültiges ...-Argument.*
Tritt auf, wenn ein ungültiger Parameterwert angegeben wurde, z.B. eine `GroupSize`-Angabe, die nicht als Zahl interpretiert werden kann.

4.1 Rangfolgenbestimmung

Die Rangfolgenbestimmung ist die Implementierung einer Methode, bei der man eine Rangfolge von Dingen wie z.B. Projekten dadurch bestimmt, dass man diese paarweise vergleicht. Bei jedem Vergleich bekommt der Gewinner einen Punkt und am Ende sortiert man die Einträge entsprechend ihrer Punktezahl. Die Reduzierung auf 2er-Vergleiche vereinfacht zwar die Entscheidung pro Frage, macht die Beantwortung jedoch auch deutlich umfangreicher. Für n Elemente ergeben sich $\frac{n(n-1)}{2}$ Einzelfragen, also für $n = 8$ also schon 28 Fragen. *FlowProtocol* unterteilt dabei die Einzelfragen so in Gruppen, dass jedes Element maximal einmal pro Gruppe vorkommt. Für eine gerade Anzahl von n Elementen werden auf diese Weise $n - 1$ Gruppen mit $\frac{n}{2}$ Vergleichen gebildet und für eine ungerade Anzahl n sind es n Gruppen mit je $\frac{n-1}{2}$ Vergleichen.

Syntax

```
~Vote Key=<F-Schlüssel> [;GroupName=<G-Name>]
                        [;DrawOption=<Unentschiedenoption>]
```

Beispiel 18

```
/// Es wird eine Rangfolgenbestimmung durchgeführt
//  Anwendungsbeispiel für die Befehlsreferenz
~Vote Key=V; GroupName=Die besten Welten; DrawOption=egal
?V: Welche Welt gefällt dir besser?
    #w1: Waldwelt
    #w2: Wasserwelt
    #w3: Bergwelt
    #w4: Wichtelwelt
```

Der Aufruf dieser Vorlage führt dazu, dass die Frage mit dem Schlüssel `V` zu einer Reihe von Einzelfragen umgestaltet wird. Genauer: Die Frage wird für jede der drei 2er-Kombinationen abgefragt. Die Fragen werden zur 3er-Gruppen zusammengefasst, was hier eine Gruppe ergibt. Das Ergebnis wird unter der Gruppenüberschrift *Die besten Welten* aufgelistet. Die Option `DrawOption` ermöglicht es, für jede Paarung auch mit unentschieden zu stimmen, wobei der Text für diese Option frei gewählt werden kann. Im Beispiel oben wird als dritte Option jeweils *egal* angezeigt. Die Punktevergabe wird in diesem Fall so angepasst, dass der Sieger einer Paarung 2 Punkte bekommt und im Fall eines Unentschiedens jeder einen Punkt, so dass auch hier die Gesamtpunktzahl immer gleich bleibt.

Vorlage Demo, 16 Vote

Es wird eine Abstimmungsbewertung durchgeführt

Welche Welt gefällt dir besser?

- ◯ Waldwelt
- ◯ Wasserwelt
- ◯ egal

Welche Welt gefällt dir besser?

- ◯ Bergwelt
- ◯ Wichtelnwelt
- ◯ egal

Das Ergebnis wird dann wie folgt präsentiert:

Ergebnisliste (Demo, 16 Vote)

Die besten Welten

1. Platz 1 (6 Punkte) Bergwelt
2. Platz 2 (4 Punkte) Waldwelt
3. Platz 3 (2 Punkte) Wasserwelt
4. Platz 4 (0 Punkte) Wichtelwelt

Man beachte, es kann auch Punktegleichstand geben. Dann teilen sich mehrere Elemente einen Platz.

4.2 Zitatmodus

Sofern es bei einer Vorlage nur darum geht, die gewählten Antworten auf die Fragen zu notieren, kann man sich die Arbeit stark vereinfachen. Mit den standardmäßigen Sprachmitteln müsste man in jedem Antwortzweig Ausgaben erzeugen, die sowohl Frage, als auch Antwort beinhalten, was eine prinzipiell unnötige Wiederholung einer schon vorhandenen Information ist. Der Zitatmodus erledigt genau das für einen kompletten Vorlageteil.

Syntax

```
~Cite [GroupName=<Gruppe>]
```

Beispiel 19

```
/// Es wird gefragt, wie  "Hallo Welt" ausgegeben werden soll
//  Anwendungsbeispiel für die Befehlsreferenz
~Cite GroupName=Antwortprotokoll
?F1: Wie soll "Hallo Welt" ausgegeben werden?
    #a1: Ganz normal
    #a2: In Großbuchstaben
    #a3: Rückwärts
```

Der Aufruf dieser Vorlage führt dazu, dass die Frage als Ausgabe mit der gewählten Antwort als Unterpunkt ausgegeben wird. Das Ergebnis wird unter der Gruppenüberschrift *Antwortprotokoll* aufgelistet.

Ergebnisliste (Demo, 17 Cite)

Antwortprotokoll

1 Wie soll „Hallo Welt“ausgegeben werden?

- In Großbuchstaben