Inhaltsverzeichnis

1. Bereich 1
1.1. Beschreibung 1
1.2. Schema 1
1.3. Konfiguration 1
2. Inhalt 2
2.1. Dokumenttyp 2
2.2. Attribute 2
2.3. Dynamische Attribute 3
2.4. Renderer 3
2.5. Schema 3
3. Unterbereich 4
4. Route 4
5. Pfad 5

1. Bereich

1.1. Beschreibung

Als Bereich wird eine Sammlung von Inhalten betrachtet, die unter einer definierten Route innerhalb der Dokumentationsplattform erreichbar. Die Inhalte gehören logisch zusammen und werden unter Verwendung eines definierten Layouts dargestellt.

Ein Beispiel für einen Bereich ist die Homepage, also die Seite die angezeigt wird, wenn die URL der Dokumentationsplattform aufgerufen wird. Die definierten Inhalte (hier Verweise auf andere Bereiche) werden in Form von Kacheln dargestellt.

Bereiche werden durch XML-Dateien definiert, die `inhalte.xml` heißen.Dies können in beliebigen Ordnern im Dokumentationsrepository liegen. Sie müssen dem Dokumentationsplattform-Schema entsprechen.

1.2. Schema

Name	Pflicht	Beschreibung
name	ja	Der Name des Bereichs. Wird hauptsächlich intern verwendet um
route	nein	die Route des Bereichs

1.3. Konfiguration

Über die Konfiguration des Bereichs lassen sich weitere Einstellungen zur Darstellung konfigurieren.

Name(*=Pflichtfeld)	Datentyp	Default	Beschreibung
layout*	String	-	Die Layout-Implementierung die verwendet werden soll um diesen Bereich darzustellen. Der Autor legt damit die grundsätzliche Darstellung dieses Bereichs fest
statische-ressourcen-scan	boolean	false	Aktiviert für diesen Bereich ein Scanning nach zusätzlichen Ressourcen/Inhalten, die nicht in der `inhalte.xml` definiert sind. Gefunde Ressourcen stehen dann wie andere Inhalte zum Abruf zur Verfügung, sind aber nicht automatisch im Bereich sichtbar. Dies kann genutzt werden, um zusätzliche statische Ressourcen bereitzustellen, bspw. Schema-Dateien oder andere Dateien.
sichtbar-in-sitemap	boolean	true	Mit diesem Schalter kann die Sichtbarkeit sämtlicher Inhalte des Bereichs in der Sitemap (Nutzung durch Suchmaschinen) beeinflusst werden.

2. Inhalt

[question] Sollten wir das nicht lieber Dokument nennen?

Ein Inhalt ist eine Sammlung verschiedener Attribute, die zusammen eine Dokumentationsseite abbilden. Ein Inhalt existiert innerhalb eines Bereichs und ist über eine Route unterhalb der Bereichsroute erreichbar. Ein Inhalt muss immer einem bekannten/definiertem Dokuemnttypen entsprechen.

Diese sind im Dokumentationsplattform-Schema definiert.

2.1. Dokumenttyp

Ein Dokumenttyp definiert eine Zusammnstellung von Attributen die für einen Inhalt existieren müssen. Jeder einzelne Typ muss im System bekannt sein, um ihn korrekt darstellen und verwenden zu können. Sie sind durch das Dokumentationsplattform-Schema definiert.

2.2. Attribute

Attribute sind Werte und Eigenschaften eines Inhalts. Diese werden durch einen passenden Renderer ausgewertet und können sowohl für den eigentlichen Inhalt als auch für die Konfiguration und Anpassung der Darstellung genutzt werden.

Die konkrete Verwendung definieren der Dokumenttypen und der passende Renderer. Hier besteht eine direkte Abhängigkeit. Für den Autor dient das Dokumentationsplattform-Schema als Orientierung.

2.3. Dynamische Attribute

Alle Inhalte können über eine dynamische Attribut-Definition erweitert werden. Dazu steht im Schema das Element `attribute` zur Verfügung.

Die Definition sieht folgendermaßen aus:

```
<meinInhaltsTyp>
   //Inhalts-Elemente
   <attribut name="{{name das Attributes}}" wert="{{wert des Attrubutes}}"/>
</meinInhaltsTyp>
```

Dynamische Attribute können für verschiedene Zwecke verwendet. Beispiele hierfür sind:

1. Erweiterungen, die nicht direkt im Schema dokumentiert werden sollen und damit etwas versteckt werden sollen
2. Erweiterungen, die nur für bestimmte Teile der Anwendung benutzt werden, aber nicht generell für alle Anwendungend es Dokumenttyps relevant sind.

2.4. Renderer

Renderer sind technische Komponenten innerhalb der Dokumentationsplattform, der Dokumenttypen bzw. ein Layout kennen und darstellen können.Der Renderer prüft die Vorgaben des Autors und stellt Inhalte gemäß seiner Aufgabe dar.

Für jedes Layout MUSS ein Renderer existieren, der die prinzipielle Einrichtung und Darstellung des Bereichs übernimmt. Die verschiedenen Dokumenttypen können über eigene Renderer dargestellt werden oder aber über den Bereichsrenderer ausgewertet werden.Bspw. werden die Links direkt im Homepage-Renderer ausgewertet.

Weitere Details zu einzelnen Renderern ist im Bereich Dokumenttypen zu finden.

2.5. Schema

Basis-Schema "Inhalt"

Nachfolgend ist das Basis-Schema von Inhalten dargestellt. Die einzelnen Dokumenttypen erweitern dieses um spezifische Attribute.

Name(*=Pflichtfeld)	Datentyp	Default	Beschreibung
name*	String	-	Der logische Name eines Inhalts. Dieser dient einerseits zur Unterscheidung, kann von den Renderern auch für spezifisch Verwendet werden, bspw. als Titel etc.
pfad	String	-	Pfad zur Hauptinhaltsdatei, relativ zur Inhaltsdefinition (inhalte.xml).

Name(*=Pflichtfeld)	Datentyp	Default	Beschreibung
attribut	Liste von Name/Wert-Paaren	-	Generische/dynamische Erweiterbarkeit der Inhalte. Hierüber können Erweiterungen für die einzelnen Dokumenttypen umgesetzt werden, die nur für einzelne Anwendungsfälle benötigt werden, für die das allgemeine Schema nicht erweitert werden soll

Unterstützte generische Attribute

Die nachfolgende Tabelle listet dynamische Attribute, die allgemein (durch alle Inhalte) unterstützt werden, die jedoch NICHT als spezifische Konfiguration umgesetzt sind, da sie nur selten benötigt werden

Name	Datentyp	Default	Beschreibung
sichtbar-in-sitemap	boolean	true	Dieser Schalter erlaubt das Herausfiltern des Inhalts aus der Sitemap. Wenn Bots die Sitemap anfragen, dann wird der Inhalt herausgefiltert. Dies ist keine Garantie, dass die Inhalte nicht doch in Suchmaschinen auftauchen, bspw. wenn sie anderweitig verlinkt sind.

3. Unterbereich

Ein Unterbereich ist ein spezieller Inhalt, der ander Inhalte aggregiert und stellt damit ein Mittel zur Strukturierung dar. Diese kann zur Erhöhung der Übersicht bei der Erstellung der Dokumentation dienen. Daneben kann diese Strukturierung auch durch den jeweiligen Bereichsrenderer genutzt werden, um spezifische Aspekte des Layouts abzubilden.

Ein Beispiel hierfür ist das Layout Homepage, welches die Unterbereiche zur Gruppierung der Verweise nutzt.

4. Route

Die Route eines Inhalts definiert die Browser-Location unter der der Inhalt erreichbar ist.Ein Inhalt mit Route ist somit im Browser adressierbar.

Eine Route MUSS eindeutig sein.Haben in der Dokumentationsplattform mehrere Inhalte dieselbe Route so ist fehlerhaft und das Verhalten undefiniert.Der Autor muss dies korrigieren.

Es gibt **implizite** Routen und **explizite** Routen. **Implizite** Routen sind in der Inhaltsdefinition nicht extra aufgeführt und konfiguriert (der Inhalt hat kein eigenes `<route>`-Element).Der Inhalt hat dann eine Route, die je nach Dokumenttyp definiert ist und die Route des Eltern-Inhalts bzw. des Bereichs berücksichtigt.

Bei **expliziten** Routen ist die Route im Inhalt in Form des `<route>`-Elements angegeben.Es werden

sowohl **absolute** als auch **relative** Routen unterstützt.

Relative Routen berücksichtigen die Route des Eltern-Inhalts und erweitern diese für den Inhalt.Der Autor muss die hierarchischen Grenzen berücksichtigen.Die folgende Tabelle listet exemplarische Verwendungen von relativen Routen:

Eltern-Route	`route`**-Konfiguration**	**Route des Inhalts**
/basis	sub-route	/basis/sub-route
/basis	./sub-route	/basis/sub-route
/basis	../sub-route	/sub-route
/basis	sub1/sub2/sub3	/basis/sub1/sub2/sub3
/basis	sub1/../sub2	/basis/sub2

Neben den relativen Routen kann der Autor auch **absolute** Routen vergeben. Absolute Routen beginnen mit einem `/` und ignorieren die Eltern-Routen. Jeder Inhalt mit Unterstützung für das `route`-Element kann mit einer absoluten Route konfiguriert werden. Bspw. kann der Autor für einen Unterbereich eine absolute Route konfiguriere. Dessen Kind-Inhalte erben dann diese absolute und nutzen sie, wenn relative oder implizite Routen verwendet werden.

Der Autor kann darüber eigene Routen für bestimmte Inhalte in einem Bereich vergeben.

5. Pfad

Inhalte, deren Daten in Form von Dateien im Repository vorliegen oder eine andere Notwendigkeit haben, verwenden den Pfad zur Adressierung. Nicht jeder Inhalt hat somit einen Pfad.

Aggregierende Inhalte (Bereich, Unterbereich) haben einen Pfad, der auf ein Verzeichnis im Repository verweisen sollte. Unterbereiche erben **implizit** den Pfad des Eltern-Elements direkt, d.h. haben denselben Pfad wie das Eltern-Element. Bei **expliziter** Konfigurations via `<pfad>`-Element wir der Elternpfad erweitert (wenn es sich um eine relative Pfadangabe handelt)

Für die Adressierung gelten ähnliche Regeln, wie für die Routen und es können **relative** und **absolute** Pfade genutzt werden.