eBook Gratuit

APPRENEZ sitecore

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#sitecore

Table des matières

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: sitecore

It is an unofficial and free sitecore ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official sitecore.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapitre 1: Démarrer avec sitecore

Remarques

Cette section fournit une vue d'ensemble de ce qu'est la sitecore et pourquoi un développeur peut vouloir l'utiliser.

Il devrait également mentionner tous les sujets importants dans sitecore, et établir un lien avec les sujets connexes. La documentation de sitecore étant nouvelle, vous devrez peut-être créer des versions initiales de ces rubriques connexes.

Versions

Version	Date de sortie
8.1	2015-01-01
8.0	2015-01-01
7.5	2014-01-01
7.2	2014-01-01
7.1	2013-01-01
7.0	2013-01-01
6.6	2012-01-01
6,5	2011-01-01
6.4	2010-01-01
6.3	2010-01-01
6.2	2009-01-01
6.1	2009-01-01
6,0	2008-01-01
5.3	2006-01-01
5.2	2006-01-01
5.1	2005-01-01
4.3	2004-01-01

Examples

Installation ou configuration

Fichier exécutable

Téléchargez le fichier .exe à l' adresse http://dev.sitecore.net/ et double-cliquez pour démarrer. Ce fichier .exe fera tout pour vous: attachez des bases de données, modifiez le fichier hôte et définissez les autorisations de dossier. Le seul inconvénient est qu'il laisse une entrée dans le registre sous [HKEY_LOCAL_MACHINE \ SOFTWARE \ CMS CMS] et rend votre instance disponible sous «Ajout / Suppression de programmes», ce qui est trompeur. Sitecore est juste une application ASP.NET, pas une application Desktop.

Installation manuelle

Vous pouvez choisir d'installer et de configurer Sitecore manuellement en téléchargeant le fichier zip racine du site. Vous êtes responsable de la connexion des bases de données et de la sécurité de la configuration des autorisations. Vous souhaiterez peut-être utiliser cette méthode dans des environnements qui nécessitent des étapes d'installation personnalisée (les bases de données sont sur un autre serveur, des autorisations personnalisées sont nécessaires). Suivez le guide d'installation (disponible sur http://dev.sitecore.net/ ; assurez-vous de lire le guide correspondant à votre version spécifique) et - si vous effectuez l'installation en production - le Guide de renforcement de la sécurité.

Une fois que vous êtes devenu un développeur certifié Sitecore, vous pourrez télécharger la dernière version de Sitecore et tous les modules associés sur http://dev.sitecore.net/ . Sur dev.sitecore.net, il existe toujours deux formats différents disponibles pour le téléchargement: la racine complète du site ou un fichier exécutable. Vous trouverez ci-dessous une liste de méthodes d'installation de Sitecore.

Pas à pas

1. Téléchargez la version de Sitecore sur le site Web de Sitecore - si vous préférez la méthode d’installation automatique, les étapes suivantes couvrent le processus manuel
2. Décompressez les archives dans un dossier sur votre ordinateur. Vous verrez les sous-dossiers suivants:

- **Données** - Sitecore conserve plusieurs archives liées à l'instance actuellement installée, telles que les journaux, les index, les packages et les fichiers de cache.
- **Databases** - Fichiers de **base de** données de SQL Server et Oracle
- **Site Web** - Tout ce qui sera accessible depuis le navigateur Web.

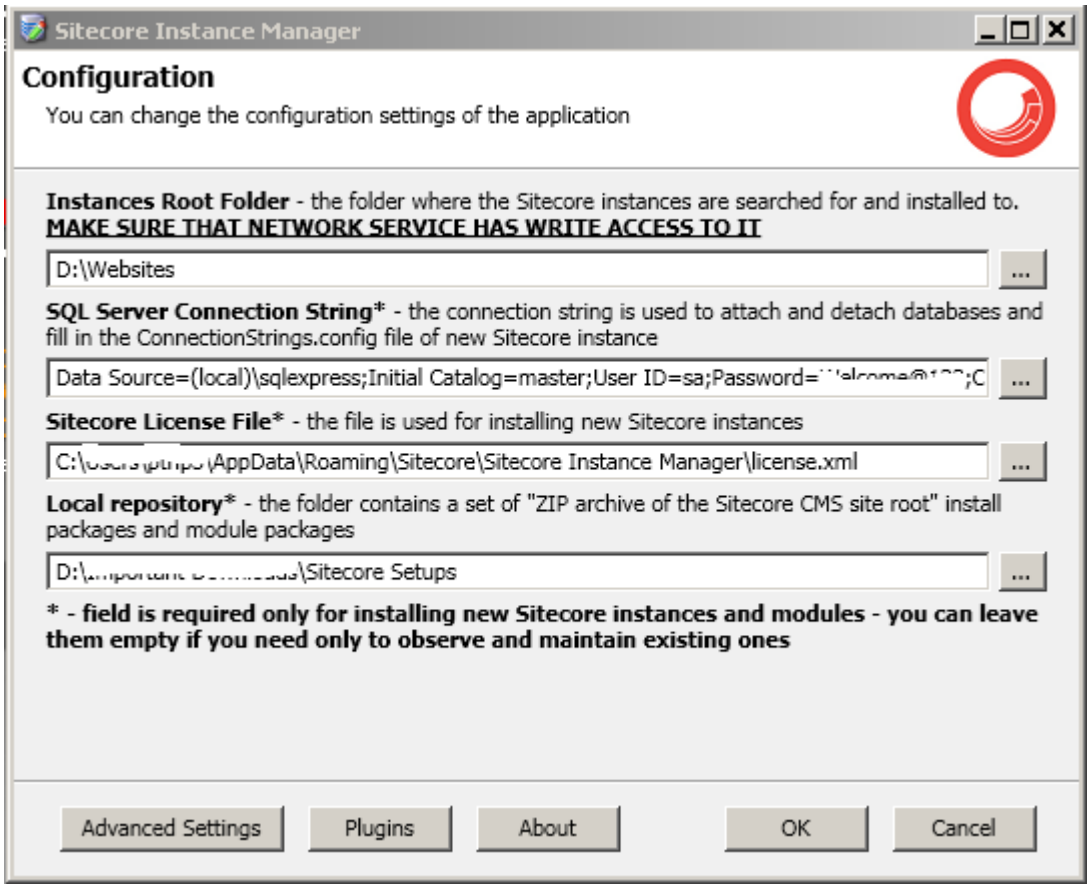
3. Ouvrez IIS et créez un nouveau site Web en lien avec le dossier "Site Web"
4. Définissez le pool d'applications de votre site Web pour utiliser le .NET Framework vers la version 4.5
5. Ajustez votre chemin DataFolder (sur le site web.config) - pour Dev, vous pouvez copier le dossier Data dans le dossier du site Web
6. Copiez le fichier license.xml dans le dossier Data

7. Dans les propriétés du dossier du site Web, décochez la case «Lecture seule». Dans Dev, vous pouvez attribuer des autorisations complètes à l'utilisateur du pool d'applications. En production, vous devez suivre les instructions contenues (ici), sous la rubrique «4.2 Configuration des autorisations de dossier et de registre».
8. Installez toutes les bases de données dans le dossier «Database» et modifiez les configurations dans le fichier ConnectionStrings.config, situé dans le dossier Website / App_Config.
9. Toujours à la ConnectionStrings.config, installez les informations de connexion pour toutes les entrées MongoDB (celles avec connectionString = "mongodb: ...")
10. Si votre site est sur localhost, pointez votre navigateur sur http: // localhost / sitecore . Si vous voyez un écran de connexion, vous avez correctement installé Sitecore.

    Source: http://sitecore-community.github.io/docs/sitecore-basics/installing-sitecore/

Utilisation du gestionnaire SIM pour l'installation

Sitecore Instance Manager est un outil open-source utilisé pour gérer le parc local des instances Sitecore. Vous pouvez installer, localiser, gérer, restaurer ou supprimer des produits Sitecore. Il vous aide également à installer votre instance de sitecore avec n'importe quel paquet sitecore, modules et la seule chose que vous devez faire est de définir le dossier pour le référentiel local où tous les fichiers d'installation sont conservés. Voir la capture d'écran ci-dessous pour les paramètres.



Il dispose également d'une API et d'un moteur de plug-in pour vous permettre de l'étendre à vos besoins.

Pour l'installation, cliquez ici

Pour plus d'informations sur SIM, cliquez ici.

Lire Démarrer avec sitecore en ligne: https://riptutorial.com/fr/sitecore/topic/1007/demarrer-avec-sitecore

Chapitre 2: Articles

Syntaxe

- Database.GetItem (ID itemId)
- Database.GetItem (ID itemId, langue)
- Database.GetItem (ID itemId, langue, version)
- Database.GetItem (chemin de la chaîne)
- Database.GetItem (chemin de la chaîne, langage)
- Database.GetItem (chemin de la chaîne, langue, version)

Remarques

La plupart des exemples ci-dessus utilisent `Sitecore.Context.Database` pour extraire des éléments. Gardez à l'esprit que la plupart des implémentations Sitecore ont plusieurs bases de données de contenu. Il est donc important d'utiliser la base de données appropriée pour récupérer votre élément.

Examples

Obtenir l'article par ID

Pour récupérer la dernière version d'un article dans la langue actuelle:

```
Sitecore.Context.Database.GetItem(new ID("{11111111-1111-1111-1111-111111111111}"));
```

Obtenir une version spécifique de l'article

Si vous avez besoin d'une langue ou d'une version spécifique d'un élément, vous pouvez utiliser ces surcharges de `GetItem()`

```
Sitecore.Context.Database.GetItem("/sitecore/content/Sitecore", Language.Current, new
Version(5));
```

Obtenir un objet par chemin

Pour récupérer la dernière version d'un article dans la langue actuelle:

```
Sitecore.Context.Database.GetItem("/sitecore/content/Sitecore")
```

Obtenir la valeur brute du champ sur un élément Sitecore

Pour obtenir la valeur brute d'un champ dans l'élément de contexte:

```
Sitecore.Context.Item["Field Name"];
```

Pour obtenir la valeur brute d'un champ sur un élément donné, `item` :

```
item["Field Name"];
```

Publier un élément Sitecore par programmation

Lors de la publication d'un élément Sitecore par programmation, le développeur doit garder à l'esprit que Sitecore peut être configuré pour plusieurs cibles de publication, de même que plusieurs langues peuvent être définies pour un élément.

```
ID targetDatabaseFieldId = ID.Parse("{39ECFD90-55D2-49D8-B513-99D15573DE41}");

     var publishingDatabases =
                                PublishManager.GetPublishingTargets(mediaItem.Database)
                                .Select(i => i[targetDatabaseFieldId]) //Get Target Database
value
                                .Where (i => i != null)
                                .Select(i => Database.GetDatabase(i))
                                .ToArray();

                        PublishManager.PublishItem(mediaItem, publishingDatabases,

LanguageManager.GetLanguages(StaticSettings.WebDatabase).ToArray(), false, false);
```

Lire Articles en ligne: https://riptutorial.com/fr/sitecore/topic/2035/articles

Chapitre 3: Automatisation de l'engagement

Remarques

L'automatisation de Sitecore donne aux marketeurs la possibilité de créer des workflows marketing qui permettent à l'utilisateur d'accéder à différents états du site.

L'exemple d'utilisation de l'automatisation peut être le flux de travail d'enregistrement (enregistré, confirmé, connecté) ou le flux de travail d'achat (nouvelle commande, produits ajoutés, détails de paiement, achat terminé).

Examples

Obtenir le contact par nom d'utilisateur

```
ContactManager contactManager = Factory.CreateObject("tracking/contactManager", true) as
ContactManager;

Contact contact = contactManager.LoadContactReadOnly(userName);
return contact;
```

Changer l'état d'automatisation du contact

Cette méthode ne nécessite pas d'initialisation du tracker, ce qui est pratique si l'état doit être modifié en dehors du contexte du site (par exemple dans le shell).

```
var stateManager = AutomationStateManager.Create(contact);
automationStateManager.MoveToEngagementState(stateItem.ParentID, stateId);
stateManager.SaveChanges(AutomationManager.Provider);
```

Inscrire un contact dans le plan d'engagement par programmation

```
 AutomationMetadataProvider automationMetadataProvider =
Assert.ResultNotNull(Factory.CreateObject("automation/metadataProvider", true) as
AutomationMetadataProvider);
 var context = AutomationManager.Provider.GetAutomationContext(ID.Parse(contact.ContactId));

 context.Enroll(fromStateItem.ParentID, ID.Parse(fromState),
automationMetadataProvider.CalculateWakeUpDateTime(ID.Parse(fromState), DateTime.UtcNow),
null);
 AutomationManager.Provider.SaveAutomationContext(context);
```

Lire Automatisation de l'engagement en ligne:
https://riptutorial.com/fr/sitecore/topic/6879/automatisation-de-l-engagement

Chapitre 4: Chercher

Remarques

La recherche Sitecore est basée sur Lucene, offrant la possibilité de créer des capacités de recherche très rapides pour votre site. Au lieu d'interroger une base de données centralisée (telle que la base de données SQL de Sitecore), elle interroge les fichiers d'index Lucene stockés sur le système de fichiers physique du serveur Web. Sitecore fournit une couche d'abstraction par rapport à l'API Lucene.NET, y compris un fournisseur LINQ qui permet d'écrire Lucene pour un processus simple et familier pour les développeurs .NET. Sitecore est livré avec des index standard configurés que vous pouvez étendre ou définir vous-même. Vous pouvez également choisir d'utiliser SOLR; une plateforme centralisée et évolutive construite sur Lucene.

Examples

Configuration

Sitecore est livré avec un ensemble d'index standard préconfigurés que vous pouvez étendre ou que vous pouvez définir vous-même. `sitecore_master_index` & `sitecore_web_index` sont les plus intéressants pour votre recherche de site. Ce sont les index prédéfinis pour tous vos éléments Sitecore dans l'arborescence de vos bases de données master et web, respectivement. Ils sont configurés pour stocker tous les champs standard d'un élément Sitecore commun à tous les modèles.

Vous pouvez consulter cette configuration de l'index Web standard à cet emplacement: `<Your Site>\App_Config\Include\Sitecore.ContentSearch.Lucene.Index.Web.config`

Les principaux domaines d'importance de la configuration d'un index sont les suivants:

- *La configuration du champ* - Quels champs doivent être stockés dans l'index et comment ils doivent être stockés.
- *La stratégie* - Comment et quand l'index doit-il être mis à jour?
- *The Crawler* - L'emplacement où l'index peut obtenir ses données Sitecore

Configuration du champ

En regardant dans la configuration `sitecore_web_index` , vous pouvez voir la référence suivante: `<configuration ref="contentSearch/indexConfigurations/defaultLuceneIndexConfiguration" />` . Cela fait référence à un fichier de configuration d'index partagé trouvé ici: `<Your Site>\App_Config\Include\Sitecore.ContentSearch.Lucene.DefaultIndexConfiguration.config` . Ici, vous pouvez voir tous les champs inclus dans la configuration standard.

Il existe essentiellement deux manières de définir un champ: soit le champ provient directement d'un champ d'élément Sitecore, soit il s'agit d'un champ calculé. Un champ calculé vous permet d'écrire du code pour effectuer des calculs et stocker le résultat dans le champ. Ce code sera exécuté lorsque l'index est créé / mis à jour, *pas* lorsque l'index est interrogé. Ceci est

particulièrement utile si le champ a besoin de stocker des données agrégées, telles que des comptes, etc.

Dans l'élément `<fieldMap>` , vous verrez les éléments `<fieldNames hint="raw:AddFieldByFieldName">` & `<fields hint="raw:AddComputedIndexField">` , qui contiennent respectivement les champs directement issus et les champs calculés.

Stratégie

La stratégie de votre index détermine à quel moment votre index est mis à jour. Vous pouvez choisir parmi les options suivantes:

- **OnPublishEndAsynchronousStrategy (onPublishEndAsync)** - Lorsqu'un article est publié, l'index sera mis à jour de manière asynchrone.
- **SynchronousStrategy (syncMaster)** - Lorsqu'un élément est enregistré, l'index sera mis à jour instantanément et de manière synchrone.
- **IntervalAsynchronousStrategy (intervalleAsyncCore / intervalAsyncMaster)** - Vérifier périodiquement les mises à jour d'éléments et mettre à jour l'index de manière asynchrone
- **ManualStrategy** - Aucune mise à jour automatique des index. Les index ne seront mis à jour que manuellement (via le panneau de contrôle ou par programme)
- **RebuildAfterFullPublishStrategy (rebuildAfterFullPublish)** - Après une publication, l'index sera entièrement reconstruit
- **RemoteRebuildStrategy (remoteRebuild)** - Cette stratégie concerne plusieurs instances de Sitecore. Par exemple, si une reconstruction est demandée à partir du serveur de gestion de contenu, les serveurs de diffusion de contenu distants seront abonnés à cet événement et reconstruiront leurs propres index.

Par défaut, l'index principal est configuré en tant que `syncMaster` . En effet, si vous enregistrez des éléments dans l'éditeur d'expérience et que le rendu de la page affiche les résultats d'un index, vous souhaiterez voir les modifications apportées aux éléments immédiatement dans les résultats. L'index Web est configuré en tant que «onPublishEndAsync», car les index de votre base de données Web ont uniquement besoin d'être mis à jour lorsque des éléments sont publiés à partir de la base de données master sur le Web.

Vous pouvez également combiner plusieurs stratégies. Par exemple, si vous avez des instances Sitecore distinctes pour votre gestion de contenu (CM) et votre diffusion de contenu (CD), il serait judicieux de combiner `onPublishEndAsync` avec `remoteRebuild` , afin que les index de CD soient mis à jour lors de la publication un utilisateur déclenche une reconstruction à partir du panneau de commande du serveur CM.

Vous pouvez choisir votre stratégie en utilisant la configuration suivante:

```
<strategies hint="list:AddStrategy">
    <strategy ref="contentSearch/indexConfigurations/indexUpdateStrategies/onPublishEndAsync"
/>
</strategies>
```

Le crawler

Cela vous permet de spécifier l'emplacement des données Sitecore que vous souhaitez indexer. L'index Web a la configuration par défaut suivante:

```
<locations hint="list:AddCrawler">
    <crawler type="Sitecore.ContentSearch.SitecoreItemCrawler, Sitecore.ContentSearch">
        <Database>web</Database>
        <Root>/sitecore</Root>
    </crawler>
</locations>
```

Les deux bits importants sont les éléments `<Database>` et `<Root>` . L'élément `<Root>` vous permet de spécifier la position de départ dans votre arborescence Sitecore que l'index doit indexer. En réalité, vous avez probablement un nœud 'Home' sous le nœud de contenu auquel vous devez faire référence pour indexer uniquement le contenu / les pages plutôt que vos modèles, etc.

Créer un filtre entrant pour la recherche

Le filtre entrant est appliqué lorsque l'élément est ajouté à l'index de recherche et permet de spécifier si l'élément est inclus ou non dans l'index.

Exemples d'utilisation du filtre entrant - n'incluez pas les valeurs standard et les versions précédentes de l'élément dans l'index.

Les filtres entrants sont définis dans la configuration:

```
<indexing.filterIndex.inbound>
 <processor type="Sitecore.ContentSearch.Pipelines.IndexingFilters.ApplyInboundIndexFilter,
Sitecore.ContentSearch"></processor>
</indexing.filterIndex.inbound>
```

Mise en œuvre du code:

```
public class ApplyInboundIndexVersionFilter : InboundIndexFilterProcessor
{
    public override void Process(InboundIndexFilterArgs args)
    {
        var item = args.IndexableToIndex as SitecoreIndexableItem;

        if (!item.Item.Versions.IsLatestVersion())
        {
            args.IsExcluded = true;
        }
    }
}
```

Plus d'exemples et d'informations peuvent être trouvés sur http://www.sitecore.net/learn/blogs/technical-blogs/sitecore-7-development-team/posts/2013/04/sitecore-7-inbound-and-outbound- filtre-pipelines.aspx

Créer un filtre sortant pour la recherche

Le filtre sortant peut être utilisé pour filtrer les résultats de la recherche.

L'un des exemples d'utilisation du filtre sortant consiste à supprimer les éléments auxquels l'utilisateur n'a pas accès à partir des résultats de recherche.

Les filtres sortants sont définis dans la configuration:

```
<indexing.filterIndex.outbound>
 <processor
type="Sitecore.ContentSearch.Pipelines.IndexingFilters.ApplyOutboundSecurityFilter,
Sitecore.ContentSearch"></processor>
</indexing.filterIndex.outbound>
```

Exemple d'implémentation de filtre sortant:

```
public class ApplyOutboundIndexWorkflowFilter : OutboundIndexFilterProcessor
{
    public override void Process(OutboundIndexFilterArgs args)
    {
        //You can use args.IsExcluded to remove items from the search results here
    }
}
```

Plus d'exemples et d'informations peuvent être trouvés sur
http://www.sitecore.net/learn/blogs/technical-blogs/sitecore-7-development-team/posts/2013/04/sitecore-7-inbound-and-outbound- filtre-pipelines.aspx

Supprimer toutes les versions précédentes de l'élément dans l'index lors de l'ajout d'une nouvelle version

Par défaut, Sitecore ajoute toutes les versions de l'élément au sitecore_master_index. L'inconvénient est que si les utilisateurs utilisent des flux de travail et ajoutent beaucoup de versions, ils seront tous ajoutés aux résultats de recherche dans l'éditeur de contenu.

Configuration:

```
<event name="item:versionAdded" >
              <handler type="FilterPatch.Library.ContentSearch.EventHandler, AssemblyName"
method="Execute" />
       </event>
```

Implémentation du gestionnaire

```
public class EventHandler
    {
        public void Execute(object sender, EventArgs eventArgs)
        {
            var item = Event.ExtractParameter(eventArgs, 0) as Item;

            //If item has less than 2 versions - then skip
            if(item.Versions.Count < 2)
            {
                return;
            }
```

```
            var indexableItem = new SitecoreIndexableItem(item);

            var index = ContentSearchManager.GetIndex(indexableItem);

            using (var context = index.CreateDeleteContext())
            {
                foreach(var version in item.Versions.GetVersions(true))
                {
                    if(!version.Versions.IsLatestVersion())
                    {
                        var indexableItemVersion = new SitecoreIndexableItem(version);
                        context.Delete(indexableItemVersion.UniqueId);
                    }
                }
                context.Commit();
            }
        }
    }
```

Lire Chercher en ligne: https://riptutorial.com/fr/sitecore/topic/2440/chercher

Chapitre 5: Configuration de l'index

Examples

Configuration du champ

Lorsque vous ajoutez des champs personnalisés à un index Lucene, vous pouvez ajouter de nouveaux champs dans l'index en utilisant la configuration suivante:

```
 <configuration ref="contentSearch/indexConfigurations/defaultLuceneIndexConfiguration">
  <indexAllfields>false</indexAllfields>
  <fieldNames hint="raw:AddFieldByFieldName">
      <field fieldName="title" storageType="YES" indexType="TOKENIZED"   vectorType="NO"
boost="1f" type="System.String"

settingType="Sitecore.ContentSearch.LuceneProvider.LuceneSearchFieldConfiguration,
Sitecore.ContentSearch.LuceneProvider"/>
  </fieldNames>
</configuration>
```

Un champ a quelques propriétés possibles:

- `storageType`
- `indexType`
- `vectorType`
- `boost`

Ces champs sont directement liés au contenu de la classe `Sitecore.ContentSearch.LuceneProvider.LuceneSearchFieldConfiguration` . Si nous reflétons les valeurs hors de cette classe, nous pouvons voir leurs valeurs possibles, etc.

Type de stockage

```
/// <summary>Specifies whether and how a field should be stored. </summary>
public enum Store
{
    /// <summary>Store the original field value in the index. This is useful for short texts
    /// like a document's title which should be displayed with the results. The
    /// value is stored in its original form, i.e. no analyzer is used before it is
    /// stored.
    /// </summary>
    YES,
    /// <summary>Do not store the field value in the index. </summary>
    NO
}
```

Type d'index

```
/// <summary>Specifies whether and how a field should be indexed. </summary>
```

```
public enum Index
{
    /// <summary>Do not index the field value. This field can thus not be searched,
    /// but one can still access its contents provided it is
    /// <see cref="T:Lucene.Net.Documents.Field.Store">stored</see>.
    /// </summary>
    NO,
    /// <summary>Index the tokens produced by running the field's
    /// value through an Analyzer.  This is useful for
    /// common text.
    /// </summary>
    ANALYZED,
    /// <summary>Index the field's value without using an Analyzer, so it can be searched.
    /// As no analyzer is used the value will be stored as a single term. This is
    /// useful for unique Ids like product numbers.
    /// </summary>
    NOT_ANALYZED,
    /// <summary>Expert: Index the field's value without an Analyzer,
    /// and also disable the storing of norms.  Note that you
    /// can also separately enable/disable norms by calling
    /// <see cref="!:AbstractField.SetOmitNorms" />.  No norms means that
    /// index-time field and document boosting and field
    /// length normalization are disabled.  The benefit is
    /// less memory usage as norms take up one byte of RAM
    /// per indexed field for every document in the index,
    /// during searching.  Note that once you index a given
    /// field <i>with</i> norms enabled, disabling norms will
    /// have no effect.  In other words, for this to have the
    /// above described effect on a field, all instances of
    /// that field must be indexed with NOT_ANALYZED_NO_NORMS
    /// from the beginning.
    /// </summary>
    NOT_ANALYZED_NO_NORMS,
    /// <summary>Expert: Index the tokens produced by running the
    /// field's value through an Analyzer, and also
    /// separately disable the storing of norms.  See
    /// <see cref="F:Lucene.Net.Documents.Field.Index.NOT_ANALYZED_NO_NORMS" /> for what norms
are
    /// and why you may want to disable them.
    /// </summary>
    ANALYZED_NO_NORMS
}
```

VectorType

```
/// <summary>Specifies whether and how a field should have term vectors. </summary>
public enum TermVector
{
    /// <summary>Do not store term vectors. </summary>
    NO,
    /// <summary>Store the term vectors of each document. A term vector is a list
    /// of the document's terms and their number of occurrences in that document.
    /// </summary>
    YES,
    /// <summary> Store the term vector + token position information
    ///
    /// </summary>
    /// <seealso cref="F:Lucene.Net.Documents.Field.TermVector.YES">
    /// </seealso>
```

```
    WITH_POSITIONS,
    /// <summary> Store the term vector + Token offset information
    ///
    /// </summary>
    /// <seealso cref="F:Lucene.Net.Documents.Field.TermVector.YES">
    /// </seealso>
    WITH_OFFSETS,
    /// <summary> Store the term vector + Token position and offset information
    ///
    /// </summary>
    /// <seealso cref="F:Lucene.Net.Documents.Field.TermVector.YES">
    /// </seealso>
    /// <seealso cref="F:Lucene.Net.Documents.Field.TermVector.WITH_POSITIONS">
    /// </seealso>
    /// <seealso cref="F:Lucene.Net.Documents.Field.TermVector.WITH_OFFSETS">
    /// </seealso>
    WITH_POSITIONS_OFFSETS
}
```

Renforcer

Ajoutez une valeur boost à l'index Lucene pour cet élément

Lire Configuration de l'index en ligne: https://riptutorial.com/fr/sitecore/topic/7634/configuration-de-l-index

Chapitre 6: Diagnostic: Assert

Syntaxe

- Assert.ArgumentCondition (condition bool, argument string, message de chaîne)
- Assert.ArgumentNotNull (argument d'objet, argument argument de chaîne)
- Assert.ArgumentNotNull (argument d'objet, Func <string> getArgumentName)
- Assert.ArgumentNotNullOrEmpty (argument ID, argument argument de la chaîne)
- Assert.ArgumentNotNullOrEmpty (argument de chaîne, stringNameName)
- Assert.ArgumentNotNullOrEmpty (argument de chaîne, Func <string> getArgumentName)
- Assert.AreEqual (int value1, int value2, message de chaîne)
- Assert.AreEqual (int value1, int value2, format de chaîne, objet params [] args)
- Assert.AreEqual (valeur de chaîne1, valeur de chaîne2, message de chaîne)
- Assert.AreEqual (valeur de chaîne1, valeur de chaîne2, format de chaîne, objet params [] args)
- Assert.AreEqual (bool value1, bool value2, message de chaîne)
- Assert.AreEqual (valeur booléenne1, valeur booléenne2, format de chaîne, objet params [] args)
- Assert.CanRunApplication (application de chaîne)
- Assert.HasAccess (bool accessAllowed, message de chaîne)
- Assert.HasAccess (bool accessAllowed, format de chaîne, objet params [] args)
- Assert.IsEditing (élément d'élément)
- Assert.IsFalse (condition bool, message de chaîne)
- Assert.IsFalse (condition bool, Func <string> getMessage)
- Assert.IsTrue (condition bool, format de chaîne, objet params [] args)
- Assert.IsNotNull (valeur d'objet, message de chaîne)
- Assert.IsNotNull (valeur d'objet, format de chaîne, objet params [] args)
- Assert.IsNotNull (valeur d'objet, type de type)
- Assert.IsNotNull (valeur d'objet, type de type, format de chaîne, objet params [] args)
- Assert.IsNotNullOrEmpty (valeur de chaîne, message de chaîne)
- Assert.IsNotNullOrEmpty (valeur de chaîne, format de chaîne, paramètres params [] args)
- Assert.IsNull (valeur d'objet, message de chaîne)
- Assert.IsNull (valeur d'objet, format de chaîne, objet params [] args)
- Assert.IsTrue (condition booléenne, message de chaîne)
- Assert.IsTrue (condition bool, Func <string> getMessage)
- Assert.IsTrue (condition bool, format de chaîne, objet params [] args)
- Assert.Required (objet obj, message de chaîne)
- Assert.Required (objet obj, format de chaîne, objet params [] args)
- Assert.ResultNotNull <T> (résultat T, message de chaîne)
- Assert.ResultNotNull <T> (résultat T)

Examples

S'assurer que deux valeurs sont égales

Compare deux valeurs pour l'égalité. Il ne peut comparer que des chaînes, des entiers et des valeurs booléennes.

```
Assert.AreEqual(documentElement.LocalName, "xamlControls", "Xaml files must have a root node named 'xamlControls'.");
```

S'assurer qu'une valeur est vraie ou fausse

Pour affirmer qu'une valeur est vraie ou fausse:

```
Assert.IsFalse(Settings.DoBadThings, "Bad things should not happen, disable DoBadThings.");
Assert.IsTrue(magicNumber =< 42, "The magic number is greater than 42!");
```

Vous pouvez également passer des paramètres de formatage pour le message d'exception

```
Assert.IsFalse(myValue > 5, "The value should not be greater than 5, it's currently {0}", myValue);
```

RésultatNotNull

`ResultNotNull()` vérifie si l'objet transmis n'est pas nul. Si l'objet et le message ne sont pas nuls, il retournera simplement l'objet qui a été transmis, sinon il lancera `InvalidOperationException` .

```
return Assert.ResultNotNull(this.Database.GetItem(this.ItemRootId), string.Concat("Root item not found. ID: ", this.ItemRootId));
```

Objet requis

Cela vérifie si l'objet donné est null, puis lève `RequiredObjectIsNullException` si c'est le cas.

```
Assert.Required(parameter, "parameter is required.");
```

Contrôles Nul / Vide

Est non nulle

C'est une méthode très simple et populaire à utiliser pour vérifier si un élément n'est pas nul. Il vérifie simplement l'objet qui est passé pour voir s'il est nul.

```
Assert.IsNotNull(database, type, "Name: {0}", item);
```

IsNotNullOrEmpty

Ceci est identique à IsNotNull ci-dessus, mais fonctionne sur les valeurs de chaîne au lieu des objets.

```
Assert.IsNotNullOrEmpty(propertyName, "user");
```

IsNull

Ceci est simplement un inverse de la méthode `IsNotNull()` . Cette méthode affirme que l'objet est nul.

```
Assert.IsNull(this.StandardValues, "A Standard Values item has already been created for this
template ");
```

Vérification des arguments

ArgumentCondition

Cette méthode vérifie si l'argument spécifié est vrai. Il prend également en compte le nom de l'argument consigné si la condition échoue.

```
Assert.ArgumentCondition(pageIndex >= 0, "pageIndex", "Value must be greater than or equal to
zero.");
```

ArgumentNotNull

Cette méthode garantit que l'argument passé n'est pas nul. Il y a deux signatures pour cette méthode, la première prend un objet et un nom de paramètre et effectue une simple vérification de null.

```
Assert.ArgumentNotNull(item, "item");
```

ArgumentNotNullOrEmpty

Ceci est similaire à la méthode ArgumentNotNull, mais vérifie également si l'objet est vide. Il existe trois variantes de cette méthode. La première variante prend en compte un identifiant Sitecore et un nom d'argument, et vérifie si l'ID est nul.

```
var nullId = new new ID("{00000000-0000-0000-0000-000000000000}");

// Both of these calls will result in an exception
Assert.ArgumentNotNullOrEmpty((ID)null, "null");
Assert.ArgumentNotNullOrEmpty(nullId, nameof(nullId));
```

La seconde méthode ajoute une vérification pour voir si la chaîne donnée est nulle ou vide.

```
// Both of these calls will result in an exception
Assert.ArgumentNotNullOrEmpty((string)null, "null");
Assert.ArgumentNotNullOrEmpty("", nameof(emptyString));
```

Assurez-vous que l'élément est en mode édition

Vérifie si l' `Item` transmis est en mode édition. Sinon, il `EditingNotAllowedException` une `EditingNotAllowedException` .

```
Assert.IsEditing(Sitecore.Context.Item);
```

Assertions de sécurité

CanRunApplication

Pour vérifier si l'utilisateur est autorisé à exécuter l'application donnée. Sinon, `AccessDeniedException` est `AccessDeniedException` .

```
Assert.CanRunApplication("WebEdit");
```

HasAccess

`HasAccess` vérifiera si le paramètre donné est vrai, sinon une exception `AccessDeniedException` sera lancée.

```
Assert.HasAccess(Context.User.IsAdministrator, "Only administrators can create new domains");
```

Lire Diagnostic: Assert en ligne: https://riptutorial.com/fr/sitecore/topic/1297/diagnostic--assert

Chapitre 7: Formulaires Web pour les spécialistes du marketing (WFFM)

Introduction

Web Forms For Marketeers est le module Sitecore populaire qui permet de créer des formulaires dans Sitecore et de personnaliser et d'étendre leur comportement à l'aide des actions de sauvegarde.

Plus d'informations sur le module peuvent être trouvées dans https://dev.sitecore.net/Downloads/Web_Forms_For_Marketers.aspx

Examples

Soumettre le formulaire WFFM par programmation

Cet exemple montre comment soumettre le formulaire WFFM dans le code.

```
var controlResults = new List<ControlResult>();
controlResults.Add(new ControlResult(Pdf_Request_Form.Name.ItemID.ToString(), "Name", name,
string.Empty));
controlResults.Add(new ControlResult(Pdf_Request_Form.Email.ItemID.ToString(), "Email", email,
string.Empty));
FormDataHandler.ProcessData(Pdf_Request_Form.ItemID, controlResults.ToArray(), new
IActionDefinition[] {}, DependenciesManager.ActionExecutor);
```

Lire Formulaires Web pour les spécialistes du marketing (WFFM) en ligne: https://riptutorial.com/fr/sitecore/topic/9758/formulaires-web-pour-les-specialistes-du-marketing--wffm-

Chapitre 8: Glass Mapper

Introduction

Glass.Mapper est le formidable framework de cartographie Open Source qui vous permet de vous concentrer sur la résolution de vos problèmes métier. Il travaille dur pour convertir les données de votre CMS en quelque chose avec lequel votre code peut fonctionner.

En utilisant Glass.Mapper et votre CMS préféré, vous pouvez mapper des données sur des modèles fortement typés dans votre code C #. Vos modèles ne requièrent aucun marquage spécial pour fonctionner avec Glass.Mapper et presque aucune configuration pour démarrer.

Remarques

Visitez cette URL pour des tutoriels http://glass.lu/Mapper/Sc/Tutorials

Examples

La manière la plus simple de mapper les données Sitecore au code.

Glass.Mapper.Sc vous permet de déplacer vos données de Sitecore et dans votre code de manière transparente en utilisant des objets fortement typés.

Le framework vous permet de mapper des données sur des classes et interfaces c # sans aucun balisage supplémentaire. Les données étant mappées sur vos objets cibles, elles sont converties dans le type cible. Jetez un oeil à cet exemple simple:

```
public class Demo
    {
        public virtual Guid Id { get; set; }

        public virtual string Title { get; set; }

        public virtual DateTime Date { get; set; }

        public virtual string Url { get; set; }
    }


    public void DoWork(
            ISitecoreContext sitecoreContext)
    {
        var model =
            sitecoreContext.GetCurrentItem<Demo>();

        var url = model.Url;
    }
```

Lire Glass Mapper en ligne: https://riptutorial.com/fr/sitecore/topic/8845/glass-mapper

Chapitre 9: Licorne

Introduction

Unicorn est un utilitaire pour Sitecore qui résout le problème du déplacement de modèles, de rendus et d'autres éléments de base de données entre des instances Sitecore. Cela devient problématique lorsque les développeurs ont leurs propres instances locales - les packages sont sujets aux erreurs et ont tendance à être oubliés sur le chemin de la production. Unicorn résout ce problème en écrivant des copies sérialisées des éléments Sitecore sur le disque avec le code - de cette manière, une copie des éléments de base de données nécessaires pour une base de code donnée l'accompagne dans le contrôle de code source.

Remarques

Des informations utiles peuvent être trouvées ici:

- Licorne sur Github
- Fondateur de Licorne - Le blog de Kam Figy
- Sitecore: Démarrer avec Unicorn!

Examples

La configuration initiale

- Vous aurez besoin de Sitecore 6.6 ou version ultérieure (y compris Sitecore 8.x). Notez que pour la compatibilité avec Sitecore 6.6, vous devez avoir installé .NET 4.5.
- Installez Unicorn. C'est aussi simple que d'ajouter le package Unicorn NuGet à votre projet.

PM> Install-Package Unicorn

- Lorsque vous installez le package NuGet, un fichier README s'affiche dans Visual Studio pour vous aider à démarrer.

Installation / installation manuelle à partir de la source

- Cloner le référentiel
- Placez une copie de votre assembly Sitecore.Kernel.dll dans / lib / sitecore / v7 (pour v7 / v8)
- Générez le projet pour votre version de Sitecore à l'aide de Visual Studio 2012 ou version ultérieure
- Copiez les fichiers Unicorn.dll, Rainbow.dll, Rainbow.Storage.Sc.dll, Rainbow.Storage.Yaml.dll et Kamsar.WebConsole.dll dans votre projet principal de la manière que vous souhaitez (référence de projet, références binaires, etc.)
- Copier les fichiers de configuration standard * .config dans le dossier App_Config \ Include \ Unicorn
- Configurez à votre goût le fichier README d'installation est un bon point de départ.

- Hit $ yoursite / unicorn.aspx pour effectuer la sérialisation initiale de votre prédicat configuré

Architecture du fournisseur de données

Le fournisseur de données Unicorn comporte deux composants: l'implémentation spécifique à la base de données et l'implémentation Unicorn.

L'implémentation Unicorn est une configuration individuelle des dépendances Unicorn qui reçoivent une sérialisation automatique. Par exemple, si vous sérialisiez deux préréglages, vous auriez besoin de deux instances de `UnicornDataProvider` - une pour chaque implémentation `IPredicate` .

L'implémentation spécifique à la base de données est une sous-classe du fournisseur de données Sitecore d'origine qui fournit un conteneur pour une ou plusieurs instances `UnicornDataProvider` . `UnicornSqlServerDataProvider` est fourni `UnicornSqlServerDataProvider` . Vous pouvez lancer votre propre si vous êtes sur Oracle. Ce fournisseur est en fait un gestionnaire d'événement pouvant être débloqué qui permet à Unicorn d'intercepter les modifications d'éléments même si la classe `EventDisabler` maléfique est utilisée.

Si vous souhaitez raccorder plusieurs fournisseurs de données Unicorn à votre base de données, vous créez une classe `UnicornSqlServerDataProvider` de `UnicornSqlServerDataProvider` . Dans cette classe, vous pouvez choisir de:

- Créez un constructeur qui injecte votre ou vos fournisseurs en utilisant le constructeur de base:

```
public MyDataProvider(string connectionString) :
base(connectionString, new UnicornDataProvider(), new
UnicornDataProvider(), ...)
```

- Créez un constructeur qui injecte votre (vos) fournisseur (s) à l'aide du code (c'est mieux si vous devez construire des dépendances, etc. qui ne correspondent pas bien à un appel de base):

```
public MyDataProvider(string connectionString) : base(connectionString, null)
 {
    AddUnicornDataProvider(new UnicornDataProvider());
    // ...
 }
```

Lire Licorne en ligne: https://riptutorial.com/fr/sitecore/topic/8852/licorne

Chapitre 10: LinkManager

Syntaxe

- chaîne statique publique GetItemUrl (élément d'élément)
- chaîne statique publique GetItemUrl (élément Item, options UrlOptions);

Examples

Obtenir une URL pour un article

Étant donné un simple élément de sitecore:

```
Item item;
```

L'élément lui-même ne contient pas son URL. Pour obtenir une URL pour un élément, vous devez appeler la classe `static Sitecore.Links.LinkManager`

```
string url = LinkManager.GetItemUrl(item);
```

une surcharge de ceci accepte une classe `UrlOptions` :

```
UrlOptions options = new UrlOptions
{
     AddAspxExtension = false
     ....
};
string url = LinkManager.GetItemUrl(item, options);
```

Lire LinkManager en ligne: https://riptutorial.com/fr/sitecore/topic/7664/linkmanager

Chapitre 11: Mise à l'échelle

Remarques

Sitecore hors de la boîte prend en charge l'équilibrage de charge pour plusieurs serveurs. La configuration type est le serveur de gestion de contenu (CM) et de contenu (CD), mais plusieurs serveurs CM et CD sont également pris en charge.

Examples

Implémenter des événements distants dans un environnement à charge équilibrée

Si Sitecore est configuré dans un environnement CM-CD, il peut être nécessaire de déclencher des événements sur le serveur CD lorsque des événements CM sont déclenchés.

L'exemple pourrait être tiré de la publication: end: remote sur CD lorsque les éditeurs de contenu ont publié sur CM.

Pour vous assurer que les événements se déclenchent, les étapes suivantes doivent être effectuées:

1. Vérifiez que les files d'attente d'événements sont activées dans web.config

```
    <!--  ENABLE EVENT QUEUES
            If enabled, Sitecore sends local events to the event queue available to remote
instances,
            and handles events in the queue from remote instances.
            Default value: true
      -->
      <setting name="EnableEventQueues" value="true" />
```

2. Ajoutez ScalabilitySettings.config à chaque instance. Définissez InstanceName pour chaque serveur et PublishingInstance sur le nom de l'instance du serveur CM.

L'exemple de ScalabilitySettings.config se trouve dans le dossier App_Config / Include.

```
<!--  INSTANCE NAME
            Unique name for Sitecore instance.
            Default value: (machine name and IIS site name)
      -->
      <setting name="InstanceName">
        <patch:attribute name="value">BAYERUATCD</patch:attribute>
      </setting>
      <!--  PUBLISHING INSTANCE
            Assigns the instance name of dedicated Sitecore installation for publishing
operations.
            When empty, all publishing operations are performed on the local installation of
Sitecore.
```

```
      Default vaue: (empty)
-->
<setting name="Publishing.PublishingInstance">
  <patch:attribute name="value">BAYERUATCM</patch:attribute>
</setting>
<!--  COUNTERS INSTANCE NAME
      Instance name for performance counters.
      Default value: (value of InstanceName setting)
-->
<setting name="Counters.InstanceName">
  <patch:attribute name="value">BAYERUATCD</patch:attribute>
</setting>
<!--  SECURITY CACHE EXPIRATION
      Sets the absolute expiration on the cached security data.
      A value of 00:00:00 disables automatic expiration of security caches.

-->
```

Lire Mise à l'échelle en ligne: https://riptutorial.com/fr/sitecore/topic/5043/mise-a-l-echelle

Chapitre 12: Modèles

Examples

Obtenir un modèle Sitecore par ID

Pour obtenir l'élément de gabarit par l'ID de gabarit.

```
TemplateItem templateItem = Sitecore.Context.Database.GetTemplate(new ID("{11111111-1111-1111-1111-111111111111}"));
```

Obtenir un modèle Sitecore par nom

Entrez le nom du modèle pour obtenir le modèle à l'aide de la méthode GetTemplate.

```
TemplateItem templateItem = Sitecore.Context.Database.GetTemplate("NameOfTheTemplate");
```

Lire Modèles en ligne: https://riptutorial.com/fr/sitecore/topic/2448/modeles

Chapitre 13: Sécurité

Remarques

Sitecore propose deux méthodes pour accéder aux éléments auxquels l'utilisateur contextuel n'a pas accès. La méthode préférée consiste à utiliser la classe UserSwitcher pour modifier temporairement l'utilisateur qui sera utilisé pour accéder à l'élément. La raison pour laquelle cela est préférable est que vous pouvez toujours avoir des autorisations pour le compte d'utilisateur utilisé.

L'alternative consiste à utiliser la classe SecurityDisabler. Ceci exécute l'action sans aucune contrainte de sécurité.

Il est recommandé d'utiliser uniquement ces classes pour les opérations nécessitant des autorisations élevées. La meilleure façon de garantir cela est d'utiliser le mot clé 'using' en C #; Cela garantira que le UserSwitcher / SecurityDisabler est correctement éliminé.

Examples

Désactiver la vérification des autorisations lors de l'accès à un élément

```
using (new Sitecore.SecurityModel.SecurityDisabler())
{
    var item = Sitecore.Context.Database.GetItem("/sitecore/content/home");
}
```

Emprunter l'identité d'un autre utilisateur lors de l'accès à un élément

```
var user = Sitecore.Security.Accounts.User.FromName("sitecore/testname", false);

using (new Sitecore.Security.Accounts.UserSwitcher(user))
{
    var item = Sitecore.Context.Database.GetItem("/sitecore/content/home");
}
```

Lire Sécurité en ligne: https://riptutorial.com/fr/sitecore/topic/2586/securite

Chapitre 14: Syntaxe de requête Sitecore

Remarques

Noms de modèles vs ID de modèles vs noms d'articles dans les requêtes:

Je vous recommande fortement d' *utiliser les ID de modèle et* ***non les*** *noms de modèles ou les noms d'éléments* dans vos requêtes. Cela garantira que vos requêtes fonctionnent toujours, même lorsque les modèles et / ou les éléments sont renommés.

La seule exception à cela est l'utilisation de modèles OOTB, lors de l'interrogation d'une structure OOTB, par exemple `/sitecore/content` ou `/sitecore/system/Marketing Control Panel` . Dans ces situations, la perte de lisibilité est souvent plus grande que le risque de rupture des requêtes, car ces modèles sont beaucoup moins susceptibles d'être renommés.

Notez que les noms de modèles ont été utilisés dans mes exemples, ci-dessus, pour des raisons de lecture. Ces requêtes ne doivent pas être utilisées en production, à moins que les noms de modèles ne soient remplacés par des ID de modèle.

Fiche de référence:

J'ai remarqué que le Aide-mémoire Sitecore Query n'est plus disponible au téléchargement sur le Web (tous les liens hébergés sur Sitecore sont désormais redirigés vers 404 pages). Heureusement, j'avais une copie sur ma machine et j'ai ajouté une capture d'écran ci-dessous:

Item Attributes		Common Standard Template Fields	
@@name	Case sensitive	@__Display name	Display name
@@key	ToLower(@@name)	@__Updated	Update date/time
@@templateid	Template ID	@__Updated by	Case sensitive username
@@templatename	Case sensitive	@__Created	Creation date/time
@@templatekey	ToLower(@@templatename)	@__Created by	Case sensitive username
@@id	Item ID	@__Lock	Lock owner
@@masterid	Branch template used	@__Workflow state	Workflow state ID
Supported Functions		@__Publish	Item publish date
contains('does this string', 'this string')		@__Unpublish	Item unpub. Date
startswith('does this string', 'this string')		@__Valid to	Version publish date
endswith('does this string', 'this string')		@__Valid from	Version unpublish date
not(condition)		@__Never publish	Prevent item publication
position()		@__Renderings	Layout details
Axes			
ancestor	Ancestors of item		
ancestor-or-self	Item and its ancestors		
child (/*)	Children of item		
descendant (//*)	Descendants of item		
descendant-or-self	Item and its descendants		
following	Siblings sorted after item		
parent	Parent of item		
preceding	Siblings sorted before item		
self (.)	Item		
[index]	Position index: *[5] gets the fifth child		
Operators			
and	And	/*/content//*[@@templateid='<ID>' and key='home']	
or	Or	../*[@@templateid='<ID>' or key='home']	
xor	Exclusive or	Equivalent to and not()	
|	Catenation	/*/content//*|/sitecore /media library//*	
+	Addition		
-	Subtraction		
*	Multiplication		
div	Division		
mod	Modulus (remainder)	/*/content//*[@IntField mod 2 = 1]	
=	Equal comparison		
!=	Not equal comparison		
<	Less than		
<=	Less than or equal to		
>	Greater than		
>=	Greater than or equal to		
false	Evaluates to false		
true	Evaluates to true	Equivalent to not(false)	
Prototypes			
Item selected in field	/*/content//*[contains(@ChecklistMultilistTreelistOrTreelistExField, '<ItemID>']		
Escape Item/Field Names	/*/content//#item-name#[@#field-name#='value']		
Date Comparison	/*/content/*[@__Updated > 'yyyyMMddTHHmmss']		
APIs			
Sitecore.Data.Database Sitecore.Data.Items.Item.Axes	SelectItems() SelectSingleItem()		
Sitecore.Data.Query.Query	Execute()		
Legend			
Red text	Escape with hash characters (#) when value appears in item name or field name		

Examples

Sélectionner par chemin de l'article

Question:

```
query:/sitecore/content/home/foo/bar
```

Résultat

```
bar
```

Requête relative

Article courant:

```
bar (path: /sitecore/content/home/foo/bar)
```

Question:

```
query:./child/grandchild
```

Résultat:

```
grandchild (path: /sitecore/content/home/foo/bar/child/grandchild)
```

Requête d'attributs d'article

Question:

```
query:/sitecore/content/[@@templatename='Homepage']
```

Résultat:

```
home (name: home, path: /sitecore/content/home, template name: Homepage)
```

Requête spécifique au site

Structure de l'arbre:

```
/sitecore
    /content
        /foo-site
            /home
                /my-account
        /bar-site
```

```
            /home
                /my-account
        /baz-site
            /home
                /my-account
```

- Le modèle de chaque élément du site (`foo-site` , `bar-site` , `baz-site`) est nommé `Site Node` .
- Le modèle de chaque élément de la maison (la `home` , la `home` , la `home`) est nommé `Homepage` d' `Homepage`
- Le modèle de chaque élément de compte d'utilisateur (`my-account` , `my-account` , `my-account`) est nommé `User Account Page`

Article courant:

L'élément actuel pourrait être l'élément d' `home` ou n'importe quelle page sous l'élément d' `home` pour l'un des sites donnés, et cette requête fonctionnera toujours (à condition qu'il n'y ait aucun élément avec le modèle de `Homepage` sous les éléments d' `home` article).

Question:

```
query:./ancestor-or-self::*[@@templatename='Homepage']/*[@@templatename='my-account']
```

Résultat:

En cas d'interrogation à partir de l'élément d' `home` ou de l'un de ses descendants sur le `foo-site` site:

```
/sitecore/content/foo-site/home/my-account
```

Si vous effectuez une requête à partir de l'élément d' `home` ou de l'un de ses descendants sur le `bar-site` site:

```
/sitecore/content/bar-site/home/my-account
```

En cas d'interrogation depuis l'élément d' `home` ou l'un de ses descendants sur le `baz-site` site:

```
/sitecore/content/baz-site/home/my-account
```

Lire Syntaxe de requête Sitecore en ligne: https://riptutorial.com/fr/sitecore/topic/7212/syntaxe-de-requete-sitecore

Chapitre 15: Workflow

Examples

Définit les actions et l'état des éléments dans cet état

```
public void MoveToStateAndExecuteActions(Item item, ID workflowStateId)
{
    Sitecore.Workflows.IWorkflowProvider workflowProvider = Item.Database.WorkflowProvider;
    Sitecore.Workflows.IWorkflow workflow = workflowProvider.GetWorkflow(item);

    // if item is in any workflow
    if (workflow != null)
    {
        using (new Sitecore.Data.Items.EditContext(item))
        {
            // update item's state to the new one
            item[Sitecore.FieldIDs.WorkflowState] = workflowStateId.ToString();
        }

        Item stateItem = ItemManager.GetItem(workflowStateId,
            Language.Current, Sitecore.Data.Version.Latest, item.Database,
SecurityCheck.Disable);

        // if there are any actions for the new state
        if (!stateItem.HasChildren)
            return;

        WorkflowPipelineArgs workflowPipelineArgs = new WorkflowPipelineArgs(item, null,
null);

        // start executing the actions
        Pipeline pipeline = Pipeline.Start(stateItem, workflowPipelineArgs);
        if (pipeline == null)
            return;
        WorkflowCounters.ActionsExecuted.IncrementBy(pipeline.Processors.Count);
    }
}
```

Merci à ce fil

Exécution de la commande de workflow pour modifier l'état du workflow

Merci à ce super post

Si nous voulons imiter le comportement de Sitecore UI et exécuter la commande qui changera l'état du workflow, nous devons utiliser WorkflowProvider pour obtenir une instance du workflow assignée à l'élément donné et appeler la méthode Execute avec un ID de commande choisi. Cela déclenche toutes les actions définies sous le nœud de l'élément de commande, modifie l'état de l'élément et déclenche toutes les actions automatiques définies sous le nouveau nœud d'élément d'état:

```
public static WorkflowResult ExecuteCommand(Item item, string commandName, string comment)
{
    IWorkflow workflow = item.Database.WorkflowProvider.GetWorkflow(item);

    if (workflow == null)
    {
        return new WorkflowResult(false, "No workflow assigned to item");
    }

    WorkflowCommand command = workflow.GetCommands(item[FieldIDs.WorkflowState])
        .FirstOrDefault(c => c.DisplayName == commandName);

    if (command == null)
    {
        return new WorkflowResult(false, "Workflow command not found");
    }

    return workflow.Execute(command.CommandID, item, comment, false, new object[0]);
}
```

Lire Workflow en ligne: https://riptutorial.com/fr/sitecore/topic/4235/workflow

Chapitre 16: Workflow

Remarques

Les flux de travail constituent un moyen flexible et contrôlable de création, de maintenance et de révision de contenu. Le workflow contient une liste d'états et de commandes.

Examples

Affecter un flux de travail à un élément

Les informations sur le workflow de l'élément sont stockées dans le champ "Workflow".

```
var workflow = Factory.GetDatabase("master").WorkflowProvider.GetWorkflow(workflowId);
workflow.Start(item);

var workflowId = item.Fields["__Default workflow"].Value;
var workflow = Factory.GetDatabase("master").WorkflowProvider.GetWorkflow(workflowId);
workflow.Start(item);
```

Lire Workflow en ligne: https://riptutorial.com/fr/sitecore/topic/6315/workflow

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec sitecore	Adrian Iorgu, Community, Jack Spektor, Liam, Paritosh Tripathi, Rodrigo Peplau
2	Articles	Jack Spektor, JohnD, Zachary Kniebel
3	Automatisation de l'engagement	Jack Spektor
4	Chercher	David Masters, Jack Spektor, Laurel
5	Configuration de l'index	Liam
6	Diagnostic: Assert	James de la Bastide, JohnD
7	Formulaires Web pour les spécialistes du marketing (WFFM)	Jack Spektor
8	Glass Mapper	Shriroop
9	Licorne	Shriroop
10	LinkManager	Liam
11	Mise à l'échelle	Jack Spektor, Liam
12	Modèles	Dheeraj Palagiri, Wesley Lomax
13	Sécurité	Jack Spektor, Laurel, SamMullins
14	Syntaxe de requête Sitecore	Zachary Kniebel
15	Workflow	Rodrigo Peplau