EBook Gratis

# APRENDIZAJE sitecore

Free unaffiliated eBook created from
**Stack Overflow contributors.**

#sitecore

# Tabla de contenido

# Acerca de

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.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

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

# Capítulo 1: Empezando con sitecore

## Observaciones

Esta sección proporciona una descripción general de qué es sitecore y por qué un desarrollador puede querer usarlo.

También debe mencionar cualquier tema grande dentro del sitecore y vincular a los temas relacionados. Dado que la Documentación para sitecore es nueva, es posible que deba crear versiones iniciales de los temas relacionados.

## Versiones

| Versión | Fecha de lanzamiento |
|---|---|
| 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

## Instalación o configuración

### Archivo ejecutable

Descargue el archivo .exe desde http://dev.sitecore.net/ y haga doble clic para comenzar. Este .exe hará todo por usted: adjunte bases de datos, modifique el archivo del host y configure los permisos de la carpeta. El único inconveniente es que deja una entrada en el registro en [HKEY_LOCAL_MACHINE \ SOFTWARE \ Sitecore CMS] y hace que su instancia esté disponible en 'Agregar o quitar programas', lo cual es engañoso. Sitecore es solo una aplicación ASP.NET, no una aplicación de escritorio.

### Manual de instalación

Puede elegir instalar y configurar Sitecore manualmente descargando el archivo zip del sitio. Usted es responsable de adjuntar las bases de datos y asegurarse de que los permisos se configuren correctamente. Es posible que desee utilizar este método en entornos que requieren pasos de instalación personalizados (las bases de datos están en otro servidor, se necesitan permisos personalizados). Siga la guía de instalación (disponible en http://dev.sitecore.net/ ; asegúrese de estar leyendo la guía para su versión específica) y, si está instalando en producción, la Guía de refuerzo de la seguridad.

Una vez que se convierta en un desarrollador certificado de Sitecore, podrá descargar la última versión de Sitecore y todos los módulos asociados desde http://dev.sitecore.net/ . En dev.sitecore.net, siempre hay dos formatos diferentes disponibles para descargar: la raíz completa del sitio o un archivo ejecutable. La siguiente es una lista de formas en que puede instalar Sitecore.

### Paso a paso

1. Descargue la versión de Sitecore del sitio web de Sitecore ; está bien si prefiere el método de instalación automática, pero los siguientes pasos cubren el proceso manual
2. Descomprima los archivos en una carpeta en su computadora. Verás las siguientes subcarpetas:

- **Datos** : aquí es donde Sitecore guarda una serie de archivos relacionados con la instancia actualmente instalada, como registros, índices, paquetes y archivos de caché.
- **Bases de datos** - Archivos de base de datos de SQL Server y Oracle
- **Sitio web** - Todo lo que será accesible desde el navegador web.

3. Abra IIS y cree un nuevo sitio web que enlaza con la carpeta "Sitio web"
4. Configure el grupo de aplicaciones de su sitio web para usar .NET Framework a la versión 4.5
5. Ajusta la ruta de tu DataFolder (en el web.config) - para Dev puedes copiar la carpeta de datos a la carpeta del sitio web
6. Copie el archivo license.xml a la carpeta de datos
7. En las propiedades de la carpeta del sitio web, desmarque la casilla de verificación "solo

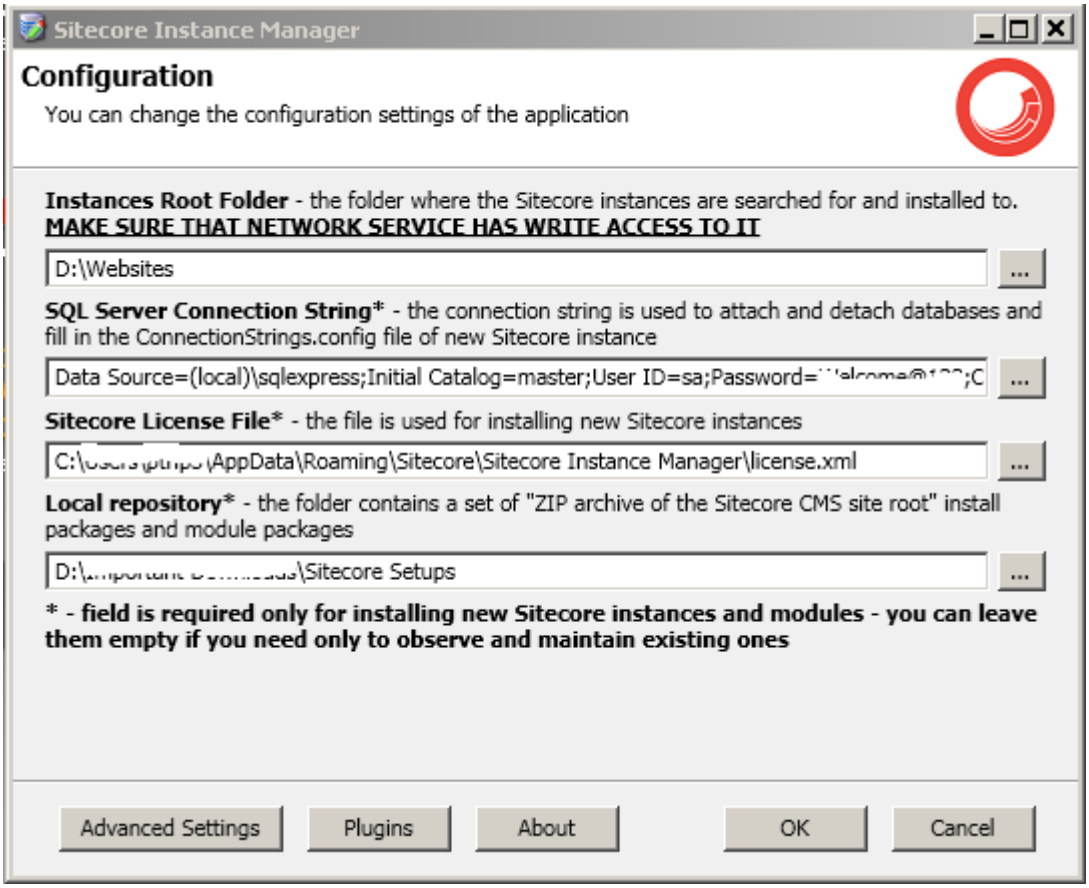
lectura". En Dev puede dar permisos completos al usuario del grupo de aplicaciones. En producción, debe seguir las instrucciones contenidas ( aquí ), bajo el ítem "4.2 Configuración de permisos de carpeta y registro".

8. Instale todas las bases de datos en la carpeta "Base de datos" y edite las configuraciones en el archivo ConnectionStrings.config, ubicado en la carpeta Website / App_Config
9. Aún en el ConnectionStrings.config, configure la información de conexión para todas las entradas de MongoDB (Aquellos con connectionString = "mongodb: ...")
10. Si su sitio está en localhost, dirija su navegador a http: // localhost / sitecore . Si ve una pantalla de inicio de sesión, ha instalado Sitecore correctamente.

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

## Usando el administrador SIM para la instalación

Sitecore Instance Manager es una herramienta de código abierto que se utiliza para administrar el parque local de las instancias de Sitecore. Puede instalar, localizar, mantener, reinstalar o eliminar productos de Sitecore. También le ayuda a instalar su instancia de sitecore con cualquier paquete de sitecore, módulos y lo único que debe hacer es configurar la carpeta para el repositorio local donde se guardan todos los archivos de instalación. Ver la captura de pantalla de abajo para la configuración.




También tiene API y motor de plugin para que pueda extenderlo a cualquiera de sus necesidades.

**Para la instalación haga clic aquí**

**Para más detalles sobre SIM, por favor haga clic aquí**

Lea Empezando con sitecore en línea: https://riptutorial.com/es/sitecore/topic/1007/empezando-con-sitecore

# Capítulo 2: Artículos

## Sintaxis

- Database.GetItem (ID itemId)
- Database.GetItem (ID itemId, idioma del idioma)
- Database.GetItem (ID itemId, idioma, versión, versión)
- Database.GetItem (cadena de ruta)
- Database.GetItem (ruta de cadena, idioma de idioma)
- Database.GetItem (ruta de la cadena, idioma, versión de la versión)

## Observaciones

La mayoría de los ejemplos anteriores utilizan `Sitecore.Context.Database` para buscar elementos. Tenga en cuenta que la mayoría de las implementaciones de Sitecore tienen múltiples bases de datos de contenido, por lo que es importante usar la base de datos correcta para recuperar su elemento.

## Examples

### Obtener artículo por ID

Para obtener la última versión de un elemento en el idioma actual:

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

### Obtener versión específica del artículo

Si necesita obtener un idioma o una versión específica de un elemento, puede usar estas sobrecargas de `GetItem()`

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

### Obtener artículo por ruta

Para obtener la última versión de un elemento en el idioma actual:

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

### Obtener valor bruto de campo en un elemento de Sitecore

Para obtener el valor en bruto de un campo en el elemento de contexto:

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

Para obtener el valor en bruto de un campo en un artículo determinado, `item` :

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

## Publicar elemento de Sitecore programáticamente

Al publicar el elemento de Sitecore mediante programación, el desarrollador debe tener en cuenta que Sitecore podría configurarse para varios destinos de publicación, así como que se podrían definir varios idiomas para el elemento.

```
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);
```

Lea Artículos en línea: https://riptutorial.com/es/sitecore/topic/2035/articulos

# Capítulo 3: Automatización de compromiso

## Observaciones

La automatización de Sitecore le brinda al comercializador la posibilidad de crear flujos de trabajo de marketing que pondrán al usuario en diferentes estados del sitio web.

El ejemplo del uso de la automatización podría ser el flujo de trabajo de registro (registrado, confirmado, conectado) o el flujo de trabajo de compra (nuevo pedido, productos agregados, detalles de pago, compra completa).

## Examples

### Recibe el contacto por nombre de usuario

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

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

### Cambiar el estado de automatización del contacto.

Este método no requiere la inicialización del rastreador, lo cual es útil si el estado debe cambiarse fuera del contexto del sitio (por ejemplo, en el shell).

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

### Inscribir contacto programáticamente en el plan de compromiso.

```
 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);
```

Lea Automatización de compromiso en línea:
https://riptutorial.com/es/sitecore/topic/6879/automatizacion-de-compromiso

# Capítulo 4: Buscar

## Observaciones

La búsqueda de Sitecore se construye sobre Lucene, lo que brinda la capacidad de crear capacidades de búsqueda muy rápidas para su sitio. En lugar de consultar una base de datos centralizada (como la base de datos SQL de Sitecore), consulta los archivos de índice de Lucene que se almacenan en el sistema de archivos físico del servidor web. Sitecore proporciona una capa de abstracción sobre la API de Lucene.NET que incluye un proveedor LINQ que hace que las consultas de Lucene sean un proceso simple y familiar para los desarrolladores de .NET. Sitecore se envía con algunos índices estándar configurados que puede ampliar o definir los suyos. También puede optar por utilizar SOLR; una plataforma centralizada y escalable construida sobre Lucene.

## Examples

### Configuración

Sitecore se envía con un conjunto de índices estándar preconfigurados que puede ampliar, o puede definir el suyo propio. De los preconfigurados, `sitecore_master_index` y `sitecore_web_index` son los más interesantes para la búsqueda de su sitio. Estos son los índices predefinidos para todos sus elementos de Sitecore en el árbol de sus bases de datos maestras y web, respectivamente, y están configurados para almacenar todos los campos estándar de un elemento de Sitecore que serán comunes entre todas las plantillas.

Puede ver esta configuración del índice web estándar en esta ubicación: `<Your Site>\App_Config\Include\Sitecore.ContentSearch.Lucene.Index.Web.config`

Las principales áreas de importancia de la configuración de un índice son:

- *Configuración de campo* : qué campos deben almacenarse en el índice y cómo deben almacenarse.
- *La estrategia* : cómo y cuándo debe actualizarse el índice.
- *El rastreador* : la ubicación donde el índice puede obtener sus datos de Sitecore

#### Configuración de campo

Mirando en la configuración de `sitecore_web_index` , puede ver la siguiente referencia: `<configuration ref="contentSearch/indexConfigurations/defaultLuceneIndexConfiguration" />` . Esto se refiere a un archivo de configuración de índice compartido que se encuentra aquí: `<Your Site>\App_Config\Include\Sitecore.ContentSearch.Lucene.DefaultIndexConfiguration.config` . Aquí puede ver todos los campos que se incluyen en la configuración estándar.

Básicamente, hay dos formas de definir un campo: el campo se obtiene directamente de un campo de elemento de Sitecore o es un campo computado. Un campo computado le permite escribir algo de código para hacer algunos cálculos y almacenar el resultado en el campo. Este

código se ejecutará cuando el índice se construya / actualice, *no* cuando se consulte el índice. Esto es particularmente útil si el campo necesita almacenar datos agregados, como conteos, etc.

Dentro del elemento `<fieldMap>` verá los elementos `<fieldNames hint="raw:AddFieldByFieldName">` y `<fields hint="raw:AddComputedIndexField">` , que contienen los campos de fuente directa y los campos calculados respectivamente.

**Estrategia**

La estrategia de su índice determina cuándo se actualiza su índice. Hay las siguientes opciones para elegir:

- **OnPublishEndAsynchronousStrategy (onPublishEndAsync)** : cuando se publica un elemento, el índice se actualizará de forma asíncrona.
- **SynchronousStrategy (syncMaster)** : cuando se guarda un elemento, el índice se actualizará de forma instantánea y sincrónica.
- **IntervalAsynchronousStrategy (intervalAsyncCore / intervalAsyncMaster)** : comprueba periódicamente las actualizaciones de elementos y actualiza el índice de forma asíncrona
- **ManualStrategy** - No hay actualizaciones automáticas de índice Los índices solo se actualizarán manualmente (a través del panel de control, o mediante programación)
- **RebuildAfterFullPublishStrategy (rebuildAfterFullPublish)** - Después de una publicación, el índice se reconstruirá por completo
- **RemoteRebuildStrategy (remoteRebuild)** : esta estrategia es para varias instancias de Sitecore. Por ejemplo, si se requiere una reconstrucción desde el servidor de administración de contenido, entonces los servidores de entrega de contenido remotos se suscribirán a este evento y reconstruirán sus propios índices.

De forma predeterminada, el índice maestro está configurado como `syncMaster` . Esto se debe a que si está en el editor de experiencia al guardar elementos y al mostrar en la página los resultados de un índice, querrá ver los cambios que haya realizado en los elementos inmediatamente en los resultados. El índice web se configura como 'onPublishEndAsync', esto se debe a que los índices de su base de datos web solo necesitan actualizarse cuando los elementos se publican desde la base de datos maestra a la web.

También puedes combinar múltiples estrategias. Por ejemplo, si tiene instancias de Sitecore separadas para la administración de contenido (CM) y la entrega de contenido (CD), tendría sentido combinar `onPublishEndAsync` con `remoteRebuild` , de modo que los índices de CD se actualicen cuando se publiquen los elementos y se reconstruyan cuando un usuario activa una reconstrucción desde el panel de control del servidor CM.

Puedes elegir tu estrategia usando la siguiente configuración:

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

**El rastreador**

Esto le permite especificar la ubicación de los datos de Sitecore que desea indexar. El índice web tiene la siguiente configuración por defecto:

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

Los dos bits importantes son los elementos `<Database>` y `<Root>` . El elemento `<Root>` permite especificar la posición inicial en su árbol de Sitecore que el índice debe indexar. En realidad, es probable que tengas un nodo 'Inicio' debajo del nodo de contenido al que apuntarías para que solo indexe el contenido real / las páginas en lugar de tus plantillas, etc.

## Crear un filtro de entrada para la búsqueda.

El filtro de entrada se aplica cuando el elemento se agrega al índice de búsqueda y permite especificar si el elemento se incluye en el índice o no.

Ejemplos de uso del filtro de entrada: no incluyen los valores estándar y las versiones anteriores del elemento en el índice.

Los filtros de entrada se establecen en la configuración:

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

Implementación de código:

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

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

Se pueden encontrar más ejemplos e información en http://www.sitecore.net/learn/blogs/technical-blogs/sitecore-7-development-team/posts/2013/04/sitecore-7-inbound-and-outbound- filter-pipelines.aspx

## Crear un filtro de salida para la búsqueda.

El filtro de salida se puede utilizar para filtrar los resultados de búsqueda.

Uno de los ejemplos de uso del filtro de salida es eliminar elementos a los que el usuario no tiene acceso desde los resultados de búsqueda.

Los filtros de salida se establecen en la configuración:

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

Ejemplo de implementación de filtro de salida:

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

Se pueden encontrar más ejemplos e información en http://www.sitecore.net/learn/blogs/technical-blogs/sitecore-7-development-team/posts/2013/04/sitecore-7-inbound-and-outbound- filter-pipelines.aspx

## Elimine todas las versiones anteriores del elemento en el índice al agregar una nueva versión

Por defecto, Sitecore agrega todas las versiones del elemento al sitecore_master_index. El inconveniente es que si los usuarios utilizan flujos de trabajo y agregan muchas versiones, todos ellos se agregarán a los resultados de búsqueda en el editor de contenido.

Configuración:

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

Implementacion de manejador

```
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();
            }
        }
    }
```

Lea Buscar en línea: https://riptutorial.com/es/sitecore/topic/2440/buscar

# Capítulo 5: Configuración del índice

## Examples

### Configuración de campo

Al agregar campos personalizados a un índice de Lucene, puede agregar nuevos campos al índice usando la siguiente configuración:

```
 <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 campo tiene un par de propiedades posibles:

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

Estos campos se relacionan directamente con el contenido de la clase `Sitecore.ContentSearch.LuceneProvider.LuceneSearchFieldConfiguration` . Si reflejamos los valores de esta clase, podemos ver sus posibles valores, etc.

## Tipo de almacenamiento

```
/// <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
}
```

## Tipo de índice

```
/// <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
}
```

# Aumentar

Agrega un valor de impulso al índice de Lucene para este artículo

Lea Configuración del índice en línea: https://riptutorial.com/es/sitecore/topic/7634/configuracion-del-indice

# Capítulo 6: Diagnóstico: afirmaciones

## Sintaxis

- Assert.ArgumentCondition (condición bool, string nombre de cadena, mensaje de cadena)
- Assert.ArgumentNotNull (argumento de objeto, cadena nombre de argumento)
- Assert.ArgumentNotNull (argumento de objeto, Func <cadena> getArgumentName)
- Assert.ArgumentNotNullOrEmpty (argumento de ID, string nombre de argumento)
- Assert.ArgumentNotNullOrEmpty (argumento de cadena, cadena nombre de argumento)
- Assert.ArgumentNotNullOrEmpty (argumento de cadena, Func <cadena> getArgumentName)
- Assert.AreEqual (int value1, int value2, string message)
- Assert.AreEqual (int value1, int value2, formato de cadena, objeto params [] argumentos)
- Assert.AreEqual (string value1, string value2, string message)
- Assert.AreEqual (string value1, string value2, string format, params object [] args)
- Assert.AreEqual (bool value1, bool value2, string message)
- Assert.AreEqual (bool value1, bool value2, formato de cadena, objeto params [] args)
- Assert.CanRunApplication (aplicación de cadena)
- Assert.HasAccess (bool accessAllowed, mensaje de cadena)
- Assert.HasAccess (bool accessAllowed, formato de cadena, objeto params [] args)
- Assert.IsEditing (item item)
- Assert.IsFalse (condición bool, mensaje de cadena)
- Assert.IsFalse (bool condition, Func <string> getMessage)
- Assert.IsTrue (condición bool, formato de cadena, objeto params [] args)
- Assert.IsNotNull (valor de objeto, mensaje de cadena)
- Assert.IsNotNull (valor de objeto, formato de cadena, objeto params [] args)
- Assert.IsNotNull (valor de objeto, tipo de tipo)
- Assert.IsNotNull (valor de objeto, tipo de tipo, formato de cadena, objeto params [] argumentos)
- Assert.IsNotNullOrEmpty (valor de cadena, mensaje de cadena)
- Assert.IsNotNullOrEmpty (valor de cadena, formato de cadena, objeto params [] argumentos)
- Assert.IsNull (valor de objeto, mensaje de cadena)
- Assert.IsNull (valor de objeto, formato de cadena, objeto params [] args)
- Assert.IsTrue (condición bool, mensaje de cadena)
- Assert.IsTrue (bool condition, Func <string> getMessage)
- Assert.IsTrue (condición bool, formato de cadena, objeto params [] args)
- Assert.Required (objeto obj, mensaje de cadena)
- Assert.Required (objeto obj, formato de cadena, objeto params [] args)
- Assert.ResultNotNull <T> (resultado de T, mensaje de cadena)
- Assert.ResultNotNull <T> (resultado de T)

## Examples

### Asegúrese de que dos valores son iguales

Compara dos valores para la igualdad. Puede comparar cadenas, enteros y valores booleanos solamente.

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

### Asegúrate de que un valor sea verdadero o falso

Para afirmar que un valor es verdadero o falso:

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

También puede pasar parámetros de formato para el mensaje de excepción

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

### ResultNotNull

`ResultNotNull()` comprueba si el objeto pasado no es nulo. Si el objeto y el mensaje no son nulos, simplemente devolverá el objeto que se pasó, de lo contrario lanzará la `InvalidOperationException` .

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

### Objeto requerido

Esto verifica si el objeto dado es nulo y, a continuación, lanza la `RequiredObjectIsNullException` si lo es.

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

### Cheques nulos / vacíos

## No es nulo

Este es un método muy simple y popular que se usa para verificar si un elemento no es nulo. Simplemente verifica el objeto que se pasa para ver si es nulo.

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

## IsNotNullOrEmpty

Es lo mismo que IsNotNull anterior, pero funciona con valores de cadena en lugar de objetos.

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

## Es nulo

Esto es simplemente una inversa del método `IsNotNull()` . Este método afirma que el objeto es nulo.

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

### Verificación de argumentos

## ArgumentCondition

Este método verifica si el argumento especificado es verdadero. También toma el nombre del argumento que se registra si la condición falla.

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

## ArgumentNotNull

Este método asegura que el argumento pasado no es nulo. Hay dos firmas para este método, la primera toma un objeto y un nombre de parámetro y realiza una simple comprobación nula.

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

## ArgumentNotNullOrEmpty

Esto es similar al método ArgumentNotNull, pero también verificará si el objeto está vacío. Hay tres variantes de este método. La primera variante toma un ID de Sitecore y un nombre de argumento, y comprueba si el ID es nulo.

```
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));
```

El segundo método agrega una verificación para ver si la cadena dada es nula o está vacía.

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

### Asegúrese de que el elemento esté en modo de edición

Comprueba si el `Item` pasado está en modo de edición. Si no, lanza una `EditingNotAllowedException` .

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

### Afirmaciones de seguridad

## CanRunApplication

Para comprobar si el usuario tiene permiso para ejecutar la aplicación dada. Si no, se lanza `AccessDeniedException` .

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

## Tiene acceso

`HasAccess` verificará si el parámetro dado es verdadero, de lo contrario se `AccessDeniedException` una `AccessDeniedException` .

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

Lea Diagnóstico: afirmaciones en línea: https://riptutorial.com/es/sitecore/topic/1297/diagnostico--afirmaciones

# Capítulo 7: Escalada

## Observaciones

Sitecore fuera de la caja soporta el equilibrio de carga para múltiples servidores. La configuración típica es la administración de contenido (CM) y el servidor de entrega de contenido (CD), sin embargo, también se admiten múltiples servidores de CM y CD.

## Examples

### Implementar eventos remotos en entornos de carga equilibrada

Si Sitecore está configurado en un entorno CM-CD, podría ser necesario disparar eventos en el servidor de CD cuando se activen los eventos CM.

El ejemplo podría ser la publicación de publicación: final: remoto en CD cuando los editores de contenido publican en CM.

Para asegurarse de que los eventos se activan, se deben realizar los siguientes pasos:

1. Compruebe que las colas de eventos están habilitadas en 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. Agregue ScalabilitySettings.config a cada instancia. Establezca InstanceName para cada servidor y PublishingInstance al nombre de instancia del servidor CM.

El ejemplo de ScalabilitySettings.config se puede encontrar en la carpeta 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.

    -->
```

Lea Escalada en línea: https://riptutorial.com/es/sitecore/topic/5043/escalada

# Capítulo 8: Flujo de trabajo

## Examples

### Establecer estado del elemento y ejecutar acciones en ese estado

```
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);
    }
}
```

Gracias a este hilo

### Ejecutando el comando de flujo de trabajo para cambiar el estado del flujo de trabajo

Gracias a este gran post

Si queremos imitar el comportamiento de la interfaz de usuario de Sitecore y ejecutar el comando que cambiará el estado del flujo de trabajo, debemos usar WorkflowProvider para obtener una instancia del flujo de trabajo asignado al elemento dado y llamar al método Execute con un ID de comando elegido. Esto activará todas las acciones que se definen bajo el nodo del elemento de comando, cambiará el estado del elemento y activará todas las acciones automáticas definidas debajo del nuevo nodo de elemento de estado:

```
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]);
}
```

Lea Flujo de trabajo en línea: https://riptutorial.com/es/sitecore/topic/4235/flujo-de-trabajo

# Capítulo 9: Flujo de trabajo

## Observaciones

Los flujos de trabajo proporcionan una forma flexible y controlable de creación, mantenimiento y revisión de contenido. El flujo de trabajo contiene una lista de estados y comandos.

## Examples

### Asignar flujo de trabajo a un elemento

La información sobre el flujo de trabajo del elemento se almacena en el campo "Flujo de trabajo".

```
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);
```

Lea Flujo de trabajo en línea: https://riptutorial.com/es/sitecore/topic/6315/flujo-de-trabajo

# Capítulo 10: Formularios web para vendedores (WFFM)

## Introducción

Web Forms For Marketeers es el popular módulo de Sitecore que permite crear formularios en Sitecore y personalizar y ampliar su comportamiento mediante acciones de guardado.

Puede encontrar más información sobre el módulo en https://dev.sitecore.net/Downloads/Web_Forms_For_Marketers.aspx

## Examples

### Presentar el formulario WFFM programáticamente

Este ejemplo muestra cómo enviar el formulario WFFM en código.

```
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);
```

Lea Formularios web para vendedores (WFFM) en línea: https://riptutorial.com/es/sitecore/topic/9758/formularios-web-para-vendedores--wffm-

# Capítulo 11: LinkManager

## Sintaxis

- cadena estática pública GetItemUrl (elemento del artículo)
- cadena estática pública GetItemUrl (elemento de elemento, opciones de UrlOptions);

## Examples

### Obtención de una url para un artículo

Dado un simple elemento sitecore:

```
Item item;
```

El artículo en sí no contiene su URL. Para obtener una url para un elemento, necesita hacer una llamada a la clase `static Sitecore.Links.LinkManager`

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

una sobrecarga de esto acepta una clase `UrlOptions` :

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

Lea LinkManager en línea: https://riptutorial.com/es/sitecore/topic/7664/linkmanager

# Capítulo 12: Mapeador de vidrio

## Introducción

Glass.Mapper es el increíble marco de mapeo de código abierto que te permite enfocarte en resolver tus problemas de negocios. Hace el arduo trabajo de convertir los datos de su CMS a algo con lo que su código puede trabajar.

Usando Glass.Mapper y su CMS favorito, puede asignar datos a modelos fuertemente tipados en su código C #. Sus modelos no requieren ningún marcado especial para trabajar con Glass.Mapper y casi no requieren configuración para comenzar.

## Observaciones

Visite este url para tutoriales http://glass.lu/Mapper/Sc/Tutorials

## Examples

### La forma más fácil de asignar datos de Sitecore a código.

Glass.Mapper.Sc le permite mover sus datos desde Sitecore a su código sin problemas usando objetos fuertemente tipados.

El marco le permite asignar datos a las clases e interfaces de c # sin ningún tipo de marcado adicional. Cuando los datos se asignan a los objetos de destino, se convierten al tipo de destino. Echa un vistazo a este sencillo ejemplo:

```
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;
    }
```

Lea Mapeador de vidrio en línea: https://riptutorial.com/es/sitecore/topic/8845/mapeador-de-vidrio

# Capítulo 13: Plantillas

## Examples

### Obtener el elemento de la plantilla de Sitecore por ID

Para obtener el elemento de plantilla por ID de plantilla

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

### Obtener Sitecore Template Item por nombre

Ingrese el nombre de la plantilla para obtener la plantilla usando el método GetTemplate.

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

Lea Plantillas en línea: https://riptutorial.com/es/sitecore/topic/2448/plantillas

# Capítulo 14: Seguridad

## Observaciones

Sitecore ofrece dos formas de acceder a elementos a los que el usuario de contexto no tiene permisos de acceso. La forma preferida es usar la clase UserSwitcher para cambiar temporalmente el usuario que se usará para acceder al elemento. La razón por la que se prefiere esto es porque aún puede tener permisos vigentes para la cuenta de usuario que se está utilizando.

La alternativa es usar la clase SecurityDisabler. Esto realiza la acción sin ninguna restricción de seguridad.

Se recomienda usar solo estas clases para las operaciones que requieren permisos elevados. La mejor manera de asegurar esto es utilizando la palabra clave 'usar' en C #; esto asegurará que el UserSwitcher / SecurityDisabler esté correctamente eliminado.

## Examples

### Deshabilitar la verificación de permisos al acceder a un elemento

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

### Hacerse pasar por un usuario diferente al acceder a un elemento

```
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");
}
```

Lea Seguridad en línea: https://riptutorial.com/es/sitecore/topic/2586/seguridad

# Capítulo 15: Sintaxis de consultas de Sitecore

## Observaciones

**Nombres de plantillas frente a identificadores de plantillas frente a nombres de elementos en consultas:**

Recomiendo encarecidamente que *utilice los ID de plantilla y **no los** nombres de plantillas o nombres de elementos* en sus consultas. Esto asegurará que sus consultas seguirán funcionando, incluso cuando se renombren las plantillas y / o los elementos.

La única excepción a esto es cuando se trabaja con plantillas OOTB, mientras se consulta una estructura OOTB, por ejemplo, `/sitecore/content` o `/sitecore/system/Marketing Control Panel` . En estas situaciones, la pérdida de legibilidad es a menudo mayor que el riesgo de ruptura de consultas, ya que es mucho menos probable que se cambie el nombre de estas plantillas.

Tenga en cuenta que los nombres de las plantillas se usaron en mis ejemplos, arriba, por motivos de lectura. Esas consultas no deben usarse en producción, a menos que los nombres de las plantillas se reemplacen con los ID de plantilla.

**Hoja de referencia:**

Noté que la hoja de referencia de Sitecore Query ya no está disponible para descargar en la web (todos los enlaces alojados en Sitecore ahora se redirigen a 404 páginas). Afortunadamente, tenía una copia en mi máquina, y he añadido una captura de pantalla, a continuación:

| 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<br>Sitecore.Data.Items.Item.Axes | SelectItems()<br>SelectSingleItem() | | |
| Sitecore.Data.Query.Query | Execute() | | |
| Legend | | | |
| Red text | Escape with hash characters (#) when value appears in item name or field name | | |

# Examples

## Seleccionar por ruta de elemento

### Consulta:

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

### Resultado

```
bar
```

## Consulta relativa

### Elemento actual:

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

### Consulta:

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

### Resultado:

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

## Consulta de atributos de artículo

### Consulta:

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

### Resultado:

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

## Consulta específica del sitio

### Estructura de árbol:

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

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

- La plantilla de cada elemento del sitio ( `foo-site` , `bar-site` , `baz-site` ) se denomina `Site Node` .
- La plantilla de cada elemento de inicio ( `home` , `home` , `home` ) se llama `Homepage`
- La plantilla de cada elemento de cuenta de usuario ( `my-account` , `my-account` , `my-account` ) se denomina `User Account Page`

**Elemento actual:**

El elemento actual podría ser el elemento de `home` o cualquier página debajo del elemento de `home` para cualquiera de los sitios dados, y esta consulta seguirá funcionando (siempre que no haya elementos con la plantilla de la `Homepage` debajo de los elementos de `home` que son un antepasado de la actual ít).

**Consulta:**

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

**Resultado:**

Si consulta desde el elemento de `home` o uno de sus descendientes en el `foo-site` site:

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

Si realiza consultas desde el elemento de `home` o uno de sus descendientes en el `bar-site` sitio de `bar-site` :

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

Si realiza consultas desde el elemento de `home` o uno de sus descendientes en el `baz-site` site:

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

Lea Sintaxis de consultas de Sitecore en línea:
https://riptutorial.com/es/sitecore/topic/7212/sintaxis-de-consultas-de-sitecore

# Capítulo 16: Unicornio

## Introducción

Unicorn es una utilidad para Sitecore que resuelve el problema de mover plantillas, representaciones y otros elementos de la base de datos entre instancias de Sitecore. Esto se vuelve problemático cuando los desarrolladores tienen sus propias instancias locales: los paquetes son propensos a errores y tienden a ser olvidados en el camino hacia la producción. Unicorn resuelve este problema escribiendo copias en serie de los elementos de Sitecore en el disco junto con el código; de esta manera, una copia de los elementos de la base de datos necesarios para un código base dado lo acompaña en el control de origen.

## Observaciones

Información útil se puede encontrar aquí:

- Unicornio en Github
- Fundador de Unicorn - Blog de Kam Figy
- Sitecore: ¡Empezando con Unicorn!

## Examples

### Configuración inicial

- Necesitará Sitecore 6.6 o posterior (incluido Sitecore 8.x ). Tenga en cuenta que para la compatibilidad con Sitecore 6.6 debe tener instalado .NET 4.5.
- Instala unicornio. Esto es tan simple como agregar el paquete Unicorn NuGet a su proyecto.

PM> Install-Package Unicorn

- Cuando instale el paquete NuGet, aparecerá un archivo README en Visual Studio con ayuda para que pueda comenzar.

### Instalación manual / Instalar desde la fuente

- Clonar el repositorio
- Coloque una copia de su ensamblaje Sitecore.Kernel.dll en / lib / sitecore / v7 (para v7 / v8)
- Cree el proyecto para su versión de Sitecore usando Visual Studio 2012 o posterior
- Copie Unicorn.dll, Rainbow.dll, Rainbow.Storage.Sc.dll, Rainbow.Storage.Yaml.dll y Kamsar.WebConsole.dll a su proyecto principal de la manera que desee (referencia del proyecto, como referencias binarias, etc.)
- Copie los archivos de configuración estándar * .config a la carpeta App_Config \ Include \ Unicorn
- Configure a su gusto; El archivo README de configuración es un buen punto de partida.
- Pulse $ yoursite / unicorn.aspx para realizar la serialización inicial de su predicado

configurado

## Arquitectura del proveedor de datos

Hay dos componentes para el proveedor de datos Unicorn: la implementación específica de la base de datos y la implementación Unicorn.

La implementación de Unicorn es una configuración individual de las dependencias de Unicorn que obtienen una serialización automática. Por ejemplo, si estuviera serializando dos ajustes preestablecidos, necesitaría dos instancias de `UnicornDataProvider` , una para cada implementación de `IPredicate` .

La implementación específica de la base de datos es una subclase del proveedor de datos de Sitecore original que proporciona un contenedor para una o más instancias de `UnicornDataProvider` . Fuera de la caja, se proporciona un `UnicornSqlServerDataProvider` . Usted podría rodar su propio si estás en Oracle. Este proveedor es efectivamente un controlador de eventos que no se puede bloquear y que permite que Unicorn atrape los cambios de elementos, incluso si se usa la clase malvada `EventDisabler` .

Si desea conectar varios proveedores de datos Unicorn a su base de datos, cree una clase que se derive de `UnicornSqlServerDataProvider` . En esta clase puede seleccionar:

- Cree un constructor que inyecte su (s) proveedor (es) usando el constructor base:

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

- Cree un constructor que inyecte su (s) proveedor (es) usando un código (esto es mejor si tiene que construir dependencias, etc. que no encajan bien en una llamada de base):

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

Lea Unicornio en línea: https://riptutorial.com/es/sitecore/topic/8852/unicornio

# Creditos

| S. No | Capítulos | Contributors |
|---|---|---|
| 1 | Empezando con sitecore | Adrian Iorgu, Community, Jack Spektor, Liam, Paritosh Tripathi, Rodrigo Peplau |
| 2 | Artículos | Jack Spektor, JohnD, Zachary Kniebel |
| 3 | Automatización de compromiso | Jack Spektor |
| 4 | Buscar | David Masters, Jack Spektor, Laurel |
| 5 | Configuración del índice | Liam |
| 6 | Diagnóstico: afirmaciones | James de la Bastide, JohnD |
| 7 | Escalada | Jack Spektor, Liam |
| 8 | Flujo de trabajo | Rodrigo Peplau |
| 9 | Formularios web para vendedores (WFFM) | Jack Spektor |
| 10 | LinkManager | Liam |
| 11 | Mapeador de vidrio | Shriroop |
| 12 | Plantillas | Dheeraj Palagiri, Wesley Lomax |
| 13 | Seguridad | Jack Spektor, Laurel, SamMullins |
| 14 | Sintaxis de consultas de Sitecore | Zachary Kniebel |
| 15 | Unicornio | Shriroop |