You can control which users are able to view which pages on the live site by configuring page-level permissions and adjusting the controller action that retrieves page content.

Controlling access to pages

The following example uses the *DancingGoat.Article* page type from the Dancing Goat MVC demo application. It demonstrates how to limit access to the *On Roasts* article so that only the user *Andy* can see the page content on the live site. You can use the same approach to control page-level permissions of other users.

To control access to pages on MVC sites you first need to set page-level permissions (ACLs) for a page in the content tree:

1. Open the **Pages** application.
2. Select the *On Roasts* page in the content tree.
3. Switch to the **Properties -> Security** tab.
4. Click **Add users**.
5. Select the user *Andrew Jones (Andy)*.
6. In the *Access rights* table, click **Allow** for the **Read** permission.
7. Click **Save**.

Next, adjust the controller definition:

1. In your MVC application, open the controller that retrieves content of pages based on the *DancingGoat.Article* page type.
2. In the controller action, extend the document query expression by adding the *CheckPermissions* method. We recommend using generated providers, such as the generated *ArticleProvider* class.

```
// Gets the specified articles using the generated provider and checks their page
permissions
IEnumerable<Article> articles = ArticleProvider.GetArticles()
    .OnSite("MySite")
    .Culture("en-US")
    .Path("/Articles/", PathTypeEnum.Children)
    .CheckPermissions();
```

3. Build the MVC project.

Only the user *Andy* can now view the content of the *On Roasts* page on the live site. Other users who do not have the required permission see a page not found error.

Note: By default, the permission matrix is blank for all pages. This means that no users have permission to any action (for example, read page content) when permissions are checked.