

Appendix A: Models and JSON Examples

Note: See Appendix C for additional documentation on each field.

C# Class	Example JSON Serialization
<pre>public class ItemModel { public int Id { get; set; } public string Name { get; set; } public int ItemOrder { get; set; } public bool IsAssociated { get; set; } }</pre>	<pre>{ "Id": 211, "Name": "Abilene, TX", "ItemOrder": 212, "IsAssociated": false }</pre>
<pre>public class PageModel { public int ItemsPerPage { get; set; } public int PageIndex { get; set; } }</pre>	<pre>{ "ItemsPerPage": 25, "PageIndex": 0 }</pre>
<pre>public class PageResultModel : PageModel { public int TotalResultCount { get; set; } public bool IsPaged { get; set; } }</pre>	<pre>{ "ItemsPerPage": 25, "PageIndex": 0, "TotalResultCount": 257, "IsPaged": true }</pre>
<pre>public class ItemModel { public IList<ItemModel> Items { get; set; } }</pre>	<pre>{ "Items": [{ "Id": 211, "Name": "Abilene, TX", "ItemOrder": 212, "IsAssociated": false }, { "Id": 96, "Name": "Akron, OH", "ItemOrder": 97, "IsAssociated": false }] }</pre>
<pre>public class ItemsViewModel : ItemsModel { public PageResultModel PageResult { get; set; } }</pre>	<pre>{ "Items": [{ "Id": 211, "Name": "Abilene, TX", "ItemOrder": 212, "IsAssociated": false }, { "Id": 96, "Name": "Akron, OH", "ItemOrder": 97, "IsAssociated": false }], "PageResult": { "ItemsPerPage": 2, "PageIndex": 0, "TotalResultCount": 257, "IsPaged": true } }</pre>

C# Class	Example JSON Serialization
<pre> public class AllItemsModel { public IList<ItemModel> AssociatedItems { get; set; } public IList<ItemModel> AvailableItems { get; set; } } </pre>	<pre> { "AssociatedItems": [], "AvailableItems": [{ "Id": 211, "Name": "Abilene, TX", "ItemOrder": 212, "IsAssociated": false }, { "Id": 96, "Name": "Akron, OH", "ItemOrder": 97, "IsAssociated": false }] } </pre>
<pre> public class AllItemsViewModel : AllItemsModel { public PageResultModel AssociatedItemsPageResult { get; set; } } </pre>	<pre> { "AssociatedItems": [], "AvailableItems": [{ "Id": 211, "Name": "Abilene, TX", "ItemOrder": 212, "IsAssociated": false }, { "Id": 96, "Name": "Akron, OH", "ItemOrder": 97, "IsAssociated": false }], "AssociatedItemsPageResult": { "ItemsPerPage": 2, "PageIndex": 0, "TotalResultCount": 2, "IsPaged": false } } </pre>

Table 1: Models

Appendix B: API Samples

Note: This is not an exhaustive set of available actions, and you may not need to all of them (or even the ones below) in your solution. These are provided for sample purposes only. See Appendix C for complete documentation on all available methods.

Note: The application/json Content-Type header must be sent on all requests.

Example 1: Get all items, with filtering (in California) and paging (3 items per page, page #2)

Request:

URL-stem: /items/getallitems

POST body:

```
{
  filter: ", CA",
  availableItemsPageModel:
  {
    ItemsPerPage: 3,
    PageIndex: 1
  }
}
```

Note: PageIndex is 0-based, so PageIndex: 1 requests the 2nd page.

Example Response:

```
{
  "success": true,
  "data": {
    "AvailableItemsPage": {
      "TotalResultCount": 61,
      "IsPaged": true,
      "ItemsPerPage": 3,
      "PageIndex": 1
    },
    "AssociatedItems": [],
    "AvailableItems": [
      {
        "Id": 262,
        "Name": "Berkeley, CA",
        "ItemOrder": 263,
        "IsAssociated": false
      },
      {
        "Id": 254,
        "Name": "Burbank, CA",
        "ItemOrder": 255,
        "IsAssociated": false
      },
      {
        "Id": 88,
        "Name": "Chula Vista, CA",
        "ItemOrder": 89,
        "IsAssociated": false
      }
    ]
  }
}
```

Example 2: Update an item

URL-stem: /items/updateitems

POST body:

```
{
  items: [
    {
      "Id": 54,
      "Name": "Anaheim, CA",
      "ItemOrder": 1,
      "IsAssociated": true
    }
  ]
}
```

Example Success Response:

```
{
  "success": true,
  "data": null
}
```

Example Error Response:

```
{
  "success": false,
  "errorMessage": "I'm sorry Dave. I'm afraid I can't do that...",
  "errorDetails": "System.InvalidOperationException: I'm sorry Dave. I'm afraid I can't do that...\r\n at Association.ModelAdapters.ItemsModelAdapter.UpdateItems(ItemsModel model) in c:\\dev\\website-project\\Association\\ModelAdapters\\ItemsModelAdapter.cs:line 132\r\n at Association.Controllers.ItemsController.<>c__DisplayClass16.<UpdateItems>b__15() in c:\\dev\\website-project\\Association\\Controllers\\ItemsController.cs:line 139\r\n at Association.Controllers.ItemsController.UpdateModel(Action setter) in c:\\dev\\website-project\\Association\\Controllers\\ItemsController.cs:line 160"
}
```

Appendix C: API Reference

Methods

The following table documents all available methods. To invoke a method, HTTP POST a JSON serialization of the parameters described for the method. See Appendix B for examples. Each method returns a `ResponseModel` response with the data field set to an instance of the return type indicted (except for `void` in which case data is not populated).

GetAllItems
<pre>/// <summary> /// Gets all associated items and available items, with optional name filtering applied /// and available items optionally paged. /// </summary> /// <param name="filter">The filter to apply to item names. If not specified, no filter /// is applied</param> /// <param name="availableItemsPageModel"> /// The page model to apply to available items. If not specified, no paging is applied /// all available items are returned. /// </param> AllItemsViewModel GetAllItems(string filter = null, PageModel availableItemsPageModel = null)</pre>
GetAvailableItems
<pre>/// <summary> /// Gets all available items, with optional name filtering and paging applied /// </summary> /// <param name="filter">The filter to apply to item names. If not specified, no filter /// is applied</param> /// <param name="pageModel"> /// The page model to apply to available items. If not specified, no paging is applied /// all available items are returned. /// </param> ItemsViewModel GetAvailableItems(string filter = null, PageModel pageModel = null)</pre>
GetAssociatedItems
<pre>/// <summary> /// Gets all available items, with optional name filtering applied /// </summary> /// <param name="filter">The filter to apply to item names. If not specified, no filter /// is applied</param> void GetAssociatedItems(string filter = null)</pre>
UpdateAllItems
<pre>/// <summary> /// Updates all available and associated items in the specified model. /// </summary> /// <remarks> /// IsAssociated state on each item is ignored, and instead determined from the item's /// membership in either the /// available or associated lists in the model. /// </remarks> void UpdateAllItems(AllItemsModel model)</pre>

UpdateAllAssociatedItems
<pre> /// <summary> /// Updates all items in the specified model, forcing their status to associated /// </summary> /// <remarks> /// IsAssociated state on each item is ignored, and instead is forced to true /// </remarks> void UpdateAllAssociatedItems(ItemsModel model) </pre>
UpdateAllAvailableItems
<pre> /// <summary> /// Updates all items in the specified model, forcing their status to available /// </summary> /// <remarks> /// IsAssociated state on each item is ignored, and instead is forced to true /// </remarks> void UpdateAllAvailableItems(ItemsModel model) </pre>
UpdateItems
<pre> /// <summary> /// Updates all items in the specified model /// </summary> void UpdateItems(ItemsModel model) </pre>

Models

The following table describes the structure of all available models.

ItemModel
<pre> /// <summary> /// Represents an item in the system /// </summary> /// <remarks> /// This model is appropriate for view rendering and updates /// </remarks> ItemModel { /// <summary> /// The unique identifier of the item /// </summary> int Id { get; set; } /// <summary> /// The name of the item /// </summary> string Name { get; set; } /// <summary> /// The order of the item if it is associated /// </summary> int ItemOrder { get; set; } /// <summary> /// A boolean value indicating if the item is associated or not /// </summary> bool IsAssociated { get; set; } } </pre>

PageModel

```
/// <summary>
/// Represents metadata for a requested page of results
/// </summary>
PageModel
{
    /// <summary>
    /// The number of items requested per page
    /// </summary>
    int ItemsPerPage { get; set; }

    /// <summary>
    /// The zero-based index of the requested page
    /// </summary>
    int PageIndex { get; set; }
}
```

PageResultModel

```
/// <summary>
/// Represents metadata for a page of results
/// </summary>
PageResultModel : PageModel
{
    /// <summary>
    /// The total number of results, regardless of page
    /// </summary>
    int TotalResultCount { get; set; }

    /// <summary>
    /// A boolean indicating whether the request was paged
    /// </summary>
    bool IsPaged { get; set; }

    PageResultModel() { }

    PageResultModel(PageModel pageModel, int totalResultCount)
    {
        if (pageModel == null) { throw new ArgumentNullException("pageModel"); }

        this.ItemsPerPage = pageModel.ItemsPerPage;
        this.PageIndex = pageModel.PageIndex;
        this.TotalResultCount = totalResultCount;
        this.IsPaged = true;
    }
}
```

ItemsModel

```
/// <summary>
/// A collection of multiple items
/// </summary>
/// <remarks>
/// This model is appropriate for view rendering and updates
/// </remarks>
ItemsModel
{
    /// <summary>
    /// The collection of items
    /// </summary>
    List<ItemModel> Items { get; set; }
}
```

ItemsViewModel

```
/// <summary>
/// A collection of multiple items
/// </summary>
/// <remarks>
/// This model is appropriate for view rendering only
/// </remarks>
ItemsViewModel : ItemsModel
{
    PageResultModel PageResult { get; set; }
}
```

AllItemsModel

```
/// <summary>
/// A collection of all available and associated items
/// </summary>
/// <remarks>
/// This model is appropriate for view rendering and updates
/// </remarks>
AllItemsModel
{
    /// <summary>
    /// The associated items
    /// </summary>
    /// <remarks>
    /// All items will have .IsAssociated == true
    /// </remarks>
    IList<ItemModel> AssociatedItems { get; set; }

    /// <summary>
    /// The available items
    /// </summary>
    /// <remarks>
    /// All items will have .IsAssociated == false
    /// </remarks>
    IList<ItemModel> AvailableItems { get; set; }
}
```

AllItemsViewModel

```
/// <summary>
/// A collection of all available and associated items
/// </summary>
/// <remarks>
/// This model is appropriate for view rendering only
/// </remarks>
AllItemsViewModel : AllItemsModel
{
    PageResultModel AvailableItemsPage { get; set; }
}
```

ResponseModel

```
/// <summary>
/// Contains response data and metadata. Use for all responses
/// </summary>
ResponseModel
{
    /// <summary>
    /// A bool indicating whether the request was processed successfully
    /// </summary>
    bool success { get; set; }

    /// <summary>
    /// The JSON serialized response data, if applicable; otherwise null
    /// </summary>
    object data { get; set; }

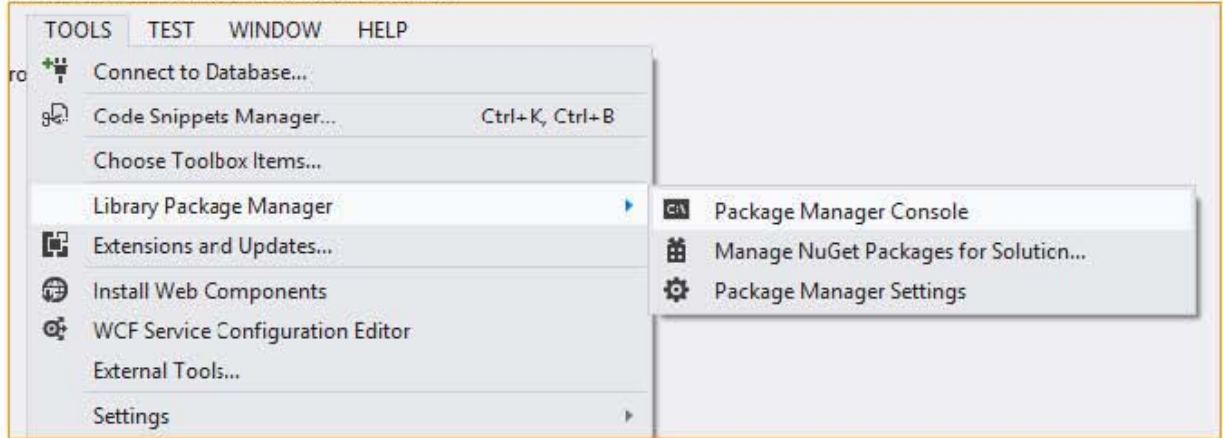
    /// <summary>
    /// A short, descriptive error message if success == false; otherwise
    /// null or undefined.
    /// </summary>
    string errorMessage { get; set; }

    /// <summary>
    /// An optional detailed message regarding the error (e.g. a stack
    /// trace) when success == false; otherwise null or undefined
    /// </summary>
    bool errorDetails { get; set; }
}
```

Appendix C: Troubleshooting Guide

A: The build fails with a "Could not locate the assembly "EntityFramework"" warning or error: This can occur if you do not Entity Framework installed, or have the wrong version. Follow these steps:

1. Launch the Package Manager Console



2. Uninstall any existing version of EF with this command: `Uninstall-Package EntityFramework`
3. Install EFv5 with this command: `Install-Package EntityFramework -Version 5.0.0`

B: When I run the website I get a "Configuration Error" with the message: "Unrecognized element 'providers'."

Sometimes, Visual Studio can modify the web.config file with settings that don't work with EFv5. If you see this error, replace the web.config file with the one sent in the .zip file originally (after completing steps described in A: above).