Venimus Dev Diary

# Wednesday 11th Dec 2019

Hooked up the amend registration details button. Decided to simplify things by combining the POST and PUT methods for CREATE/AMENDED event registration into a single PUT method.

Next step it to make the registration questions a bit more generic. Currently we ask the following three questions:

* Number of Guests
* Dietary Requirements
* Message To Organiser

It would be nice to make this more generic so that in the future the group owner can define their own questions.

# Tuesday 10th Dec 2019

Hooked up the button to allow a user to unregister from an event. Although from the websites point of view we are issuing a delete command to the endpoint (api/user/groups/{groupSlug}/Events/{eventSlug})

Internally we just update their registration record to say that they are no longer attending. This means we can tell if a user has never signed up for the event, or if they were signed up but can no longer attend.

The view members button on the MyGroups page is now hidden if the user doesn’t have access to call the API. (They must be either a system administrator or be an approved member of the group)

I’m now wondering if I need to change the design of the APIs slightly. Instead of having multiple APIs to return a list of groups, I think we need just one, but it also contains a list of available actions you can carry out on the group.

For example:

* Join Group
* View Members
* Leave Group

The API might need a new parameter to be able to specify if you only want to see groups you are/aren’t a member of.

# Monday 9th Dec 2019

Aim for today was to add support for the rendering of the markdown used in the descriptions of the groups and events.

First decision is where the conversion into HTML should take place. We can do it either in the APIs or the Frontend. I think it makes sense to do it in the frontend as it’s possible that a client might need a format other than HTML. For example, in a windows app.

I’ve included a nuget library called Markdig. This seems to be straight forward to use

var html = Markdown.ToHtml(markdown);

As we need to do this in a few places in the frontend site I’ve create a custom TagHelper called **MarkdownTagHelper**. This means that razor view we just need to write

<**markdown**>@item.Description</**markdown**>

The is library is then rendering links and text formatting (bold etc) fine. Had trouble with carriage returns at first. It turned out that the **TagHelper** was encoding them **as &#xD;&#xA;** which **MarkDig** didn’t understand. Easily fixed by **Replace("&#xD;", "\r").**

Next, I think I’ll add the ability to unregister from an event, and also change your registration details.