Bind Check Boxes in MVC

After the last post on how to create check boxes that use the bootstrap “btn-group” to modify the look and feel of check boxes, I thought it would be good to show how to bind these check boxes using MVC. After all, you will most likely need to display check boxes based on data from a table.

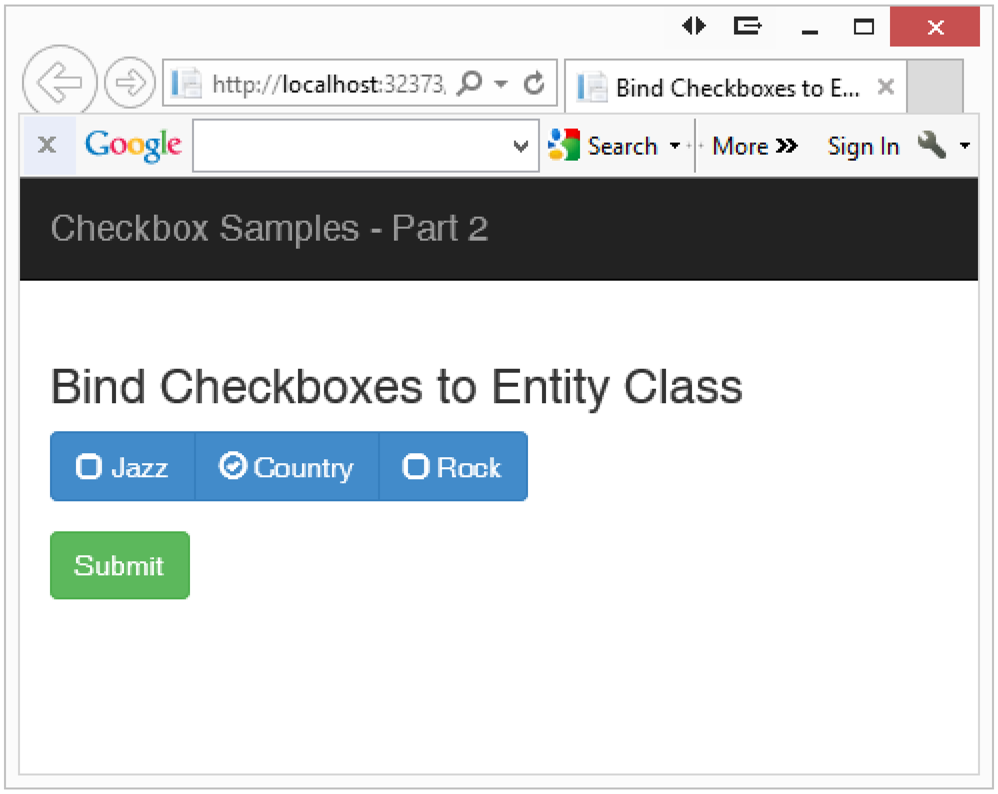

Figure 1: Check boxes should be bound to an entity class

Musical Tastes Entity Class

The first step is to have an entity (or model) class that contains the appropriate properties to bind to these check boxes. Below is a class I called MusicalTastes that simply has three Boolean properties that correspond to the three check boxes on the screen shown in Figure 1.

```
public class MusicalTastes
{
  public bool IsJazz { get; set; }
  public bool IsCountry { get; set; }
  public bool IsRock { get; set; }
}
```

View for Musical Tastes

Create a .cshtml view and add a **@model** statement at the top of the page to bind to an instance of this MusicalTastes class. Use the @Html.CheckBoxFor() helper to bind to each property instead of the @Html.CheckBox() helper as you did in the last blog entry.

```
@model BootstrapCheckBoxes2.MusicalTastes

@using (Html.BeginForm())
{
  <div class="form-group">
    <div class="btn-group" data-toggle="buttons">
      <label class="btn btn-primary">
        <span class="glyphicon glyphicon-unchecked"></span>
        @Html.CheckBoxFor(m => m.IsJazz) Jazz
      </label>
      <label class="btn btn-primary">
        <span class="glyphicon glyphicon-unchecked"></span>
        @Html.CheckBoxFor(m => m.IsCountry) Country
      </label>
      <label class="btn btn-primary">
        <span class="glyphicon glyphicon-unchecked"></span>
        @Html.CheckBoxFor(m => m.IsRock) Rock
      </label>
    </div>
  </div>
  <div class="form-group">
    <button type="submit"
            class="btn btn-success">Submit
    </button>
  </div>
}
```

Notice that the expressions you pass to the first parameter of this CheckBoxFor helper have the names of each of the properties in the MusicalTastes class. This is what binds this check box to each of the properties.

Binding to Musical Tastes

In the controller for this .cshtml page create an instance of the MusicalTastes class and set one or more of the properties to true in order to see the check box checked when the page displays.

```
public ActionResult BindingTest()
{
  MusicalTastes entity = new MusicalTastes();

  entity.IsCountry = true;

  return View(entity);
}
```

jQuery for Musical Tastes

In order to get the correct display for any property set to true you need to write some JavaScript/jQuery to toggle the glyphs. Below is the code you would add to the end of the $(document).ready(). Keep the same code you had in the previous blog post to toggle the check boxes when you click on each one, but add code that will run when the page loads as shown in the bold code below:

```
@section scripts
{
  <script>
    $(document).ready(function () {
      // Connect to 'change' event in order to toggle glyphs
      $("[type='checkbox']").change(function () {
        if ($(this).prop('checked')) {
          $(this).prev().addClass('glyphicon-ok-circle');
          $(this).prev().removeClass('glyphicon-unchecked');
        }
        else {
          $(this).prev().removeClass('glyphicon-ok-circle');
          $(this).prev().addClass('glyphicon-unchecked');
        }
      });

      // Detect checkboxes that are checked and toggle glyphs
      var checked = $("input:checked");
      checked.prev().removeClass('glyphicon-unchecked');
      checked.prev().addClass('glyphicon-ok-circle');
    });
  </script>
}
```

This code selects all check boxes checked via the automatic data binding. It then removes the unchecked glyph and adds the ok-circle glyph to all those check boxes.

Posting Back Musical Tastes Selected

There is nothing to do to get the selected check boxes to post back to your entity class. Simply create a method in your controller with the [HttpPost] attribute. Pass in the entity class to this method and MVC will take care of matching the names of the check boxes to the appropriate properties in your entity class.

```
[HttpPost]
public ActionResult BindingTest(MusicalTastes entity)
{
  System.Diagnostics.Debugger.Break();

  return View(entity);
}
```

I added the Debugger.Break() statement so I can hover over the ‘entity’ variable and verify that the check boxes checked have been updated in the instance of the MusicalTastes class passed in.

Summary

Binding an entity class with boolean properties to a set of check boxes on a .cshtml is very easy to do. Simply create your class and use the @Html.CheckBoxFor() helper class to bind your check boxes to the appropriate properties. Add a little bit of client-side JavaScript/jQuery to toggle the glyphs and you have a very nice looking interface for your check box controls.