Creating Checkboxes using Bootstrap and MVC

The normal HTML check box just does not look good. Yes, bootstrap has a CSS class that will attempt to at least let it render consistently across browsers, but it still is just the default HTML look. In addition, trying to hit a check box on a mobile phone can sometimes be a little challenging.

In an MVC project we are doing for a client, they wanted the check boxes to look more like buttons. Since we were already using bootstrap for this project we were able to create our own check boxes using the "**btn-group**" class from bootstrap. Let's walk through how we created these.

Normal Check Boxes

As most of you know, creating normal check boxes in MVC is easy to do with the CheckBox HTML helper class. The code below is what you write for creating standard HTML check boxes in MVC:

```
@using (Html.BeginForm())
{
  <div>
    @Html.CheckBox("IsJazz") Jazz
    @Html.CheckBox("IsCountry") Country
    @Html.CheckBox("IsRock") Rock
  </div>
}
```

Bootstrap Check Boxes

With bootstrap you add a <div> with a class of either “checkbox” or “checkbox-inline” around your check boxes. You also add a <label> element around the check boxes in order to use bootstrap’s styling of labels. Normally this gives you a bolder font for the actual text of the check box.

```
@using (Html.BeginForm())
{
  <div class="form-group">
    <div class="checkbox-inline">
      <label>
        @Html.CheckBox("IsJazz") Jazz
      </label>
    </div>
    <div class="checkbox-inline">
      <label>
        @Html.CheckBox("IsCountry") Country
      </label>
    </div>
    <div class="checkbox-inline">
      <label>
        @Html.CheckBox("IsRock") Rock
      </label>
    </div>
  </div>
}
```

Both of the above pieces of code will produce some output that looks like the check boxes shown in Figure 1.

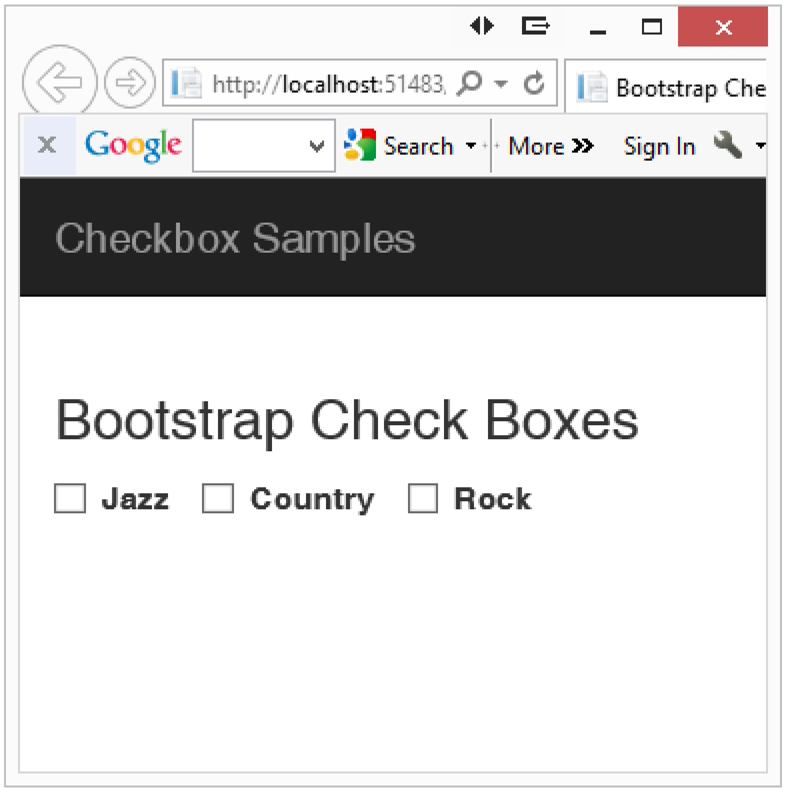

Figure 1: Normal HTML Check boxes

Use Button Groups

One of the classes that bootstrap supplies is “btn-group”. This class combined with the data-toggle=”buttons” gives you the ability to create check boxes that look like standard buttons but will toggle when pressed. What is nice about check boxes expressed this way is you can really tell that the label is a part of the check box. When you click on one of the buttons you will see it change color. In addition, these buttons are much easier to press on a mobile device.

```
@using (Html.BeginForm())
{
  <div class="form-group">
    <div class="btn-group" data-toggle="buttons">
      <label class="btn btn-primary">
        @Html.CheckBox("IsJazz") Jazz
      </label>
      <label class="btn btn-primary active">
        @Html.CheckBox("IsCountry") Country
      </label>
      <label class="btn btn-primary">
        @Html.CheckBox("IsRock") Rock
      </label>
    </div>
  </div>
}
```

Note the use of the “active” class on the Country check box. This is what makes this one selected when you display the screen (see Figure 2).

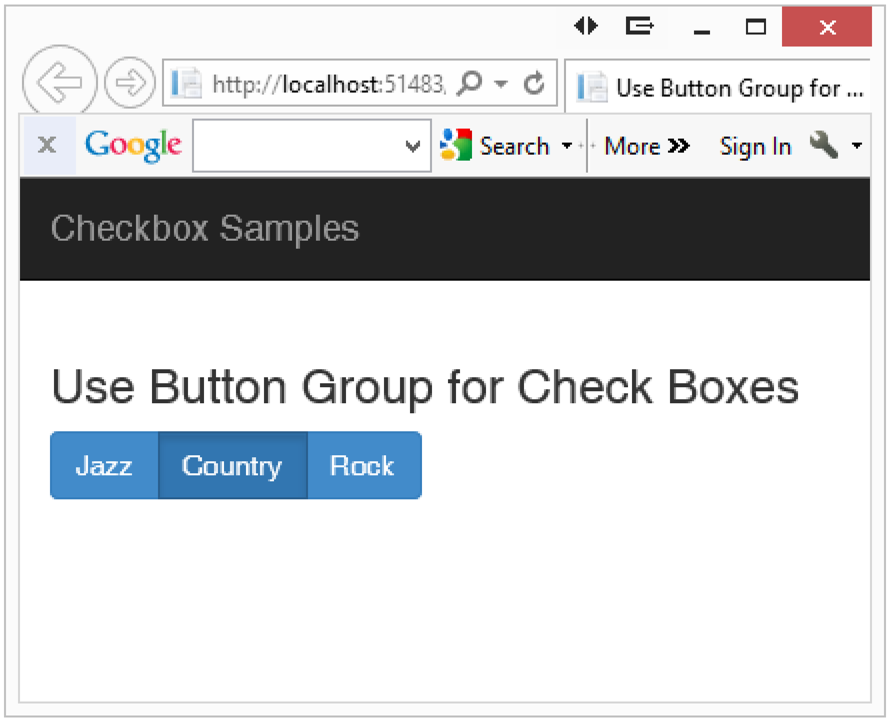

Figure 2: Check boxes displayed as button groups.

Add Glyphs

Users are used to seeing a check mark or an empty box next to the label for check box controls. So, you should add those into your design. Use the built-in glyphs in bootstrap to display a check or an empty box. Below is the code to do this:

```
@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.CheckBox("IsJazz", true) Jazz
      </label>
      <label class="btn btn-primary active">
        <span class="glyphicon glyphicon-ok-circle"></span>
        @Html.CheckBox("IsCountry") Country
      </label>
      <label class="btn btn-primary">
        <span class="glyphicon glyphicon-unchecked"></span>
        @Html.CheckBox("IsRock", true) Rock
      </label>
    </div>
  </div>
}
```

Take a look at the Country check box. There are two modifications to this check box. First, you add the “active” class on the <label> element. This makes it a different color from the other two check boxes. You also change the glyph in the <span> element from “glyphicon-unchecked” to “glyphicon-ok-circle” to give it a check mark within a circle.

Feel free to use whatever icons you want. Font Awesome (www.fontawesome.io) has a lot of different CSS glyphs that you can use instead of the limited ones in bootstrap.

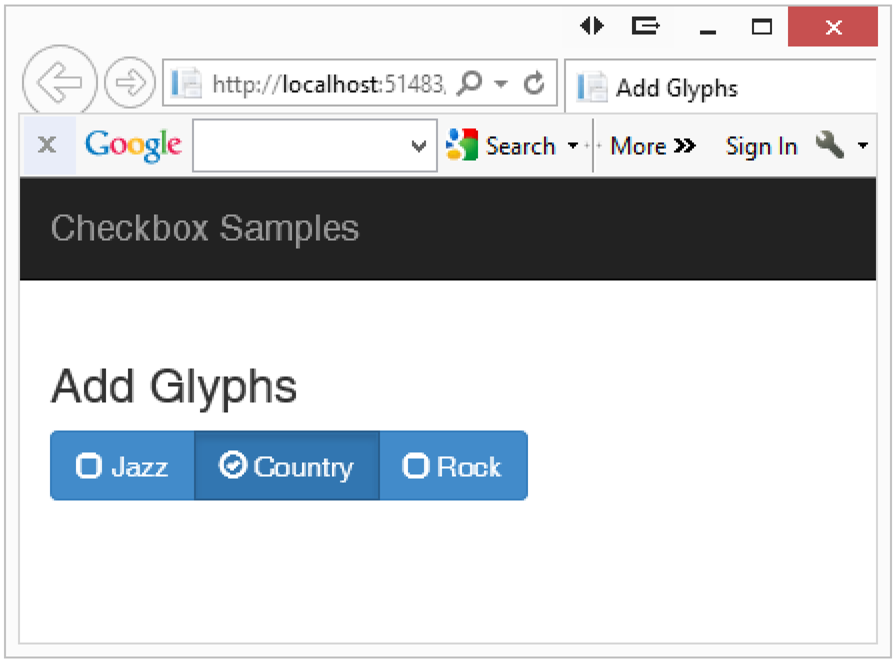

Figure 3: Add glyphs to make the buttons look more like check boxes.

Change Glyphs Dynamically using jQuery

When the user clicks on a check box you want to toggle the glyph. Luckily when you click on the check box, bootstrap takes care of toggling the "active" class on the <label> elements for you. All you need to do is to write some jQuery to toggle the glyphs. Below is the jQuery code you add at the bottom of the .cshtml MVC page to toggle the glyphs.

```
@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');
        }
      });
    });
  </script>
}
```

This jQuery is fairly simple to understand. When the user changes the check box you determine if the check box is checked or unchecked. Based on this setting you either add or remove the appropriate classes from the <span> element that is just prior to the check box control.

Summary

In this blog post you learned how to present check boxes just a little differently. While it is not necessary to do this, I like the way these check boxes look and feel. With just a little use of built-in bootstrap CSS and jQuery you can get something that looks a little different from the traditional web look.