```
using System;
using System.Globalization;
using System.Linq;
using System.Security.Claims;
using System.Threading.Tasks;
using System.Web;
using System.Web.Mvc;
using Microsoft.AspNet.Identity;
using Microsoft.AspNet.Identity.Owin;
using Microsoft.Owin.Security;
using AdventureLog.Models;
using System.Data.Entity;
using System.Collections.Generic;
using System.Security.Principal;
using System.Text.RegularExpressions;

namespace AdventureLog.Controllers
{
    /// <summary>
    /// Primary controller for managing adventures
    /// Index Route: ~/adventure
    /// </summary>
    public class AdventureHomeController : Controller
    {
        public AdventureHomeController()
        {
        }

        #region Index
        /// <summary>
        /// navigates to the ~/adventure index page.
        /// </summary>
        /// <returns>Index View</returns>
        [Authorize]
        [Route("adventure", Name ="adventurehome")]
        public ActionResult Index()
        {
            // Instantiate the model.
            var model = new AdventureHomeModel();
            // Create the view.
            var view = View(model);

            // Users must be logged in.
            if (Request.IsAuthenticated)
            {
                User.Identity.GetUserId();

                // Get the users adventures.
                using (ApplicationDbContext dbContext = new
                  ApplicationDbContext())
                {
                    var currentUserId = User.Identity.GetUserId();

                    var adventures = (from a in dbContext.Adventures
                                      let player = a.Players.FirstOrDefault(p
                    => p.UserId_PK == currentUserId)
```

```
55                                      where player.IsActive
56                                         && a.IsActive
57                                      select a).AsEnumerable();
58
59                      model.Adventures = adventures.ToList();
60                  }
61
62                  // Populate View with Administration messages.
63                  model.Messages.Add(new KeyValuePair<string, string>(
64                      "First Deployment on Azure",
65                      "Adventure Log is now running on Azure!  This is a big
                        step forward for Advnture Log."
66                      + "Our next push will be for an official product with the
                        features we wanted to create in Adventure Log."));
67              }
68
69              return view;
70          }
71
72          #endregion
73
74          #region Adventure Actions
75
76          #region Create
77
78          /// <summary>
79          /// HTTP Get: for the create adventure page.  Route: ~/adventure/
              Create
80          /// </summary>
81          [Authorize]
82          [Route("adventure/Create")]
83          public ActionResult Create()
84          {
85              // Create empty adventure with default values.
86              var model = new Adventure()
87              {
88                  CreatedDate = DateTime.Now,
89                  LastModifiedDate = DateTime.Now,
90                  LastModifiedUser = User.Identity.GetUserName()
91              };
92              var view = View("Create", model);
93
94              return View();
95          }
96
97          // Post: AdventureHome/Create
98          /// <summary>
99          /// HTTP Post for create adventure page.
100         /// </summary>
101         /// <param name="model"></param>
102         /// <returns></returns>
103         [HttpPost]
104         [ValidateAntiForgeryToken]
105         [Route("adventure/Create")]
106         public ActionResult Create(Adventure model)
107         {
```

```
108             // If required fields are filled out.
109             if (ModelState.IsValid)
110             {
111                 // Activate the adventure.
112                 model.IsActive = true;
113 
114                 // Add the user as the gamemaster of the adventure.
115                 var newPlayer = new Player()
116                 {
117                     UserId_PK = User.Identity.GetUserId(),
118                     Adventure = model,
119                     PlayerRole_PK = (int)PlayerRole.PlayerRoleKey.Gamemaster,
120                     IsActive = true
121                 };
122 
123                 // Update the database.
124                 using (var dbContext = new ApplicationDbContext())
125                 {
126                     dbContext.Adventures.Add(model);
127                     dbContext.Players.Add(newPlayer);
128                     dbContext.SaveChanges();
129                 }
130 
131                 // Go to the adventure home page.
132                 return RedirectToAction("Index");
133             }
134 
135             // If we got this far, something failed, redisplay form
136             return View(model);
137         }
138 
139         #endregion
140 
141         #region Details
142 
143         /// <summary>
144         /// Http Get for detail view of a specified Adventure
145         /// </summary>
146         /// <param name="id"></param>
147         /// <returns></returns>
148         [Authorize]
149         [Route("adventure/{id:long}")]
150         public ActionResult Details(long id)
151         {
152             Adventure adventure = null;
153             // Display the access invalid view if the adventure could not be ↵
                  found or the user does not have access to view it.
154             ActionResult view = View("AccessInvalid");
155 
156             using (var dbContext = new ApplicationDbContext())
157             {
158                 string userId = User.Identity.GetUserId();
159 
160                 // Get the adventure.
161                 adventure = (from a in dbContext.Adventures
162                              let player = a.Players.FirstOrDefault(p => ↵
```

```
                    p.UserId_PK == userId)
163                             where a.Adventure_PK == id
164                                && ( a.IsPublic || player.IsActive)
165                                && a.IsActive
166                             select a)
167                             // Include Items and areas as they are displayed
                     in cards.
168                             .Include(a => a.AdventureNotes.Select(n =>
                    n.ApplicationUser))
169                             .Include(a => a.AdventureNotes.Select(n =>
                    n.ChildNotes))
170                             .Include(a => a.Items.Select(w => w.ChildItems))
171                             .FirstOrDefault();
172             }
173
174             // If the adventure was found.
175             if (adventure != null)
176             {
177                 // Display the adventure.
178                 view = View("Details", adventure);
179             }
180
181             return view;
182         }
183
184         #endregion
185
186         #region Edit
187
188         /// <summary>
189         /// HTTP Get for editing an adventure
190         /// </summary>
191         /// <param name="id"></param>
192         /// <returns></returns>
193         [Authorize]
194         [Route("adventure/{id:long}/Edit")]
195         public ActionResult Edit(long id)
196         {
197             Adventure adventure = null;
198             // Display the access invalid view if the adventure could not be
                  found or the user does not have access to edit it.
199             ActionResult view = View("AccessInvalid");
200
201             using (var dbContext = new ApplicationDbContext())
202             {
203                 string userId = User.Identity.GetUserId();
204
205                 // Get the adventure.
206                 adventure = (from a in dbContext.Adventures
207                             let player = a.Players.FirstOrDefault(p =>
                    p.UserId_PK == userId)
208                             where a.Adventure_PK == id
209                                && player.IsActive
210                                // Assure the user is a gamemaster
211                                && player.PlayerRole.PlayerRole_PK == (long)
                    PlayerRole.PlayerRoleKey.Gamemaster
```

```
212                                 && a.IsActive
213                               select a)
214                               // Persist Player Role and Application User for displaying.
215                               .Include(a => a.Players.Select(p => p.PlayerRole))
216                               .Include(a => a.Players.Select(p => p.ApplicationUser))
217                               .FirstOrDefault();
218             }
219
220             // If the adventure was found.
221             if (adventure != null)
222             {
223                 // Display the edit view.
224                 view = View("Edit", adventure);
225             }
226
227             return view;
228         }
229
230         /// <summary>
231         /// HTTP Post for saving an edited adventure.
232         /// </summary>
233         /// <param name="model"></param>
234         /// <returns></returns>
235         [HttpPost]
236         [ValidateAntiForgeryToken]
237         [Authorize]
238         [Route("adventure/{id:long}/Edit")]
239         public ActionResult Edit(Adventure model, long[] activePlayers = null)
240         {
241             ActionResult result = View(model);
242
243             // If the model is valid, update database, and return to detail view.
244             if (ModelState.IsValid)
245             {
246                 using (var dbContext = new ApplicationDbContext())
247                 {
248                     // Update model values
249                     model.LastModifiedDate = DateTime.Now;
250                     model.LastModifiedUser = User.Identity.GetUserName();
251
252                     // If everthing goes well, update database with listed properties.
253                     dbContext.Entry(model).State = EntityState.Modified;
254
255                     // Update active player list.
256                     var players = (from p in dbContext.Players
257                                    where p.Adventure_PK == model.Adventure_PK
258                                    select p).AsEnumerable();
259
260                     foreach (var player in players)
261                     {
```

```
262                         // If there has been a change to the players active status...
263                         if (player.IsActive != activePlayers.Any(ap => ap.Equals(player.Player_PK)))
264                         {
265                             dbContext.Entry(player).Entity.IsActive = !dbContext.Entry(player).Entity.IsActive;
266                             dbContext.Entry(player).State = EntityState.Modified;
267                         }
268                     }
269 
270 
271                     // Save Changes
272                     dbContext.SaveChanges();
273                 }
274 
275                 // Return to the detail view
276                 result = RedirectToAction("Details", model.Adventure_PK);
277             }
278 
279             return result;
280         }
281 
282         #endregion
283 
284         #region Invite
285 
286         // Get: adventure/{id}/Invite/{Invite Password}
287         [Authorize]
288         [Route("adventure/{id:long}/Invite/{invitePassword?}")]
289         public ActionResult Invite(long id, string invitePassword = "")
290         {
291             ActionResult result = View("InviteFailed");
292 
293             using (var dbContext = new ApplicationDbContext())
294             {
295                 string userId = User.Identity.GetUserId();
296 
297                 // User must be logged in to use Invite Links
298                 if (userId != null)
299                 {
300                     // Find the adventure
301                     var adventure = (from a in dbContext.Adventures
302                                      where a.Adventure_PK == id
303                                         && a.IsActive
304                                      select a).FirstOrDefault();
305 
306                     // If the adventure was found and the password is valid.
307                     if (adventure != null
308                             && (string.IsNullOrWhiteSpace(adventure.InvitePassword)
309                                 || string.Equals(adventure.InvitePassword, invitePassword)))
310                     {
311                         var player = adventure.Players.Where(p => p.UserId_PK
```

```
                        == userId).FirstOrDefault();
312
313                         // Add the player if they do not exist.
314                         if (player == null)
315                         {
316                             // Add the user as a new player.
317                             var newPlayer = new Player()
318                             {
319                                 UserId_PK = userId,
320                                 Adventure = adventure,
321                                 PlayerRole_PK = (int)
                        PlayerRole.PlayerRoleKey.Player
322                             };
323
324                             // If the adventure is secured, do not whitelist
                        the player.
325                             newPlayer.IsActive = !adventure.IsSecured;
326
327                             dbContext.Players.Add(newPlayer);
328                             dbContext.SaveChanges();
329                         }
330
331                         if (!adventure.IsSecured || player.IsActive)
332                         {
333                             // Redirect to the details page of the newly
                        joined adventure.
334                             result = RedirectToAction("Details", id);
335                         }
336                         else
337                         {
338                             // Redirect to the success page.
339                             result = View("InviteSuccess");
340                         }
341                     }
342                 }
343             }
344
345             return result;
346         }
347
348         #endregion
349
350         #region Comments
351         /// <summary>
352         /// HTTP Post for creating a comment.
353         /// </summary>
354         [Authorize, HttpPost, ValidateInput(false)]
355         public ActionResult CreateComment(long Adventure_PK, string
              newComment, long? parentComment = null)
356         {
357             // Return to the detail view of the adventure.
358             ActionResult result = RedirectToAction("Details", new { id =
                  Adventure_PK });
359
360             // The comment must contain something.
361             if (!string.IsNullOrWhiteSpace(newComment))
```

```
362             {
363                 var userId = User.Identity.GetUserId();
364 
365                 // Create the comment.
366                 var note = new AdventureNote()
367                 {
368                     Adventure_PK = Adventure_PK,
369                     UserId_PK = userId,
370                     ParentAdventureNote_PK = parentComment,
371                     Text = newComment,
372                     IsActive = true,
373                     CreatedDate = DateTime.Now,
374                     LastModifiedDate = DateTime.Now,
375                     LastModifiedUser = User.Identity.GetUserName()
376                 };
377 
378                 using (var dbContext = new ApplicationDbContext())
379                 {
380                     // Check if the adventure exists that meets the qualifications.
381                     var isPlayer = (from a in dbContext.Adventures
382                                     let player = a.Players.FirstOrDefault(p => p.UserId_PK == userId)
383                                     where a.Adventure_PK == Adventure_PK
384                                         && a.IsActive
385                                         && player.IsActive
386                                     select a).Any();
387 
388                     // Add the comment.
389                     if (isPlayer)
390                     {
391                         dbContext.AdventureNotes.Add(note);
392                         dbContext.SaveChanges();
393                     }
394                 }
395             }
396 
397             return result;
398         }
399 
400         /// <summary>
401         /// HTTP Post for deleting a comment.
402         /// </summary>
403         [Authorize, HttpPost]
404         public ActionResult DeleteComment(long AdventureNote_PK)
405         {
406             ActionResult result = RedirectToAction("Index");
407 
408             using (var dbContext = new ApplicationDbContext())
409             {
410                 // Get the comment.
411                 var note = (from n in dbContext.AdventureNotes
412                             where n.AdventureNote_PK == AdventureNote_PK
413                                 && n.IsActive
414                             select n).FirstOrDefault();
415 
```

```
416                 if (note != null)
417                 {
418                     // If the user is the gamemaster.
419                     var isGamemaster = IsInRole(User.Identity,
                        note.Adventure_PK, PlayerRole.PlayerRoleKey.Gamemaster);
420
421                     if (isGamemaster)
422                     {
423                         // Delete all replies including the deleted comment.
424                         DeleteOrphanAdventureNotes(note.AdventureNote_PK,
                        dbContext);
425                         dbContext.SaveChanges();
426                         result = RedirectToAction("Details", new { id =
                        note.Adventure_PK });
427                     }
428                 }
429             }
430
431             return result;
432         }
433
434         /// <summary>
435         /// Http Post for editing the comment.
436         /// </summary>
437         [Authorize, HttpPost, ValidateInput(false)]
438         public ActionResult EditComment(long AdventureNote_PK, string
              commentText, long? parentComment = null)
439         {
440             ActionResult result = null;
441             var userId = User.Identity.GetUserId();
442
443             using (var dbContext = new ApplicationDbContext())
444             {
445                 // Get the comment to edit.
446                 var note = (from n in dbContext.AdventureNotes
447                             where n.AdventureNote_PK == AdventureNote_PK
448                             select n)
449                             .FirstOrDefault();
450
451                 if (note != null)
452                 {
453                     // Check if the adventure exists that meets the
                        qualifications.
454                     var isPlayer = (from a in dbContext.Adventures
455                                     let player = a.Players.FirstOrDefault(p
                        => p.UserId_PK == userId)
456                                     where a.Adventure_PK == note.Adventure_PK
457                                         && a.IsActive
458                                         && player.IsActive
459                                     select a).Any();
460
461                     // If the user is the player that posted the comment,
                        edit it.
462                     if (isPlayer)
463                     {
464                         note.Text = commentText;
```

```
465
466                         dbContext.Entry(note).Property("Text").IsModified =
                        true;
467                         dbContext.SaveChanges();
468
469                         result = RedirectToAction("Details", new { id =
                        note.Adventure_PK});
470                      }
471                 }
472             }
473
474
475             return result;
476         }
477
478         /// <summary>
479         /// Recursively deletes all items associated with an comment.
480         /// </summary>
481         /// <param name="AdventureNote_PK">Comment to delete.</param>
482         /// <param name="dbContext">database access to use.</param>
483         private void DeleteOrphanAdventureNotes(long AdventureNote_PK,
              ApplicationDbContext dbContext)
484         {
485             var note = (from n in dbContext.AdventureNotes
486                         where n.AdventureNote_PK == AdventureNote_PK
487                         select n)
488                         .Include(n => n.ChildNotes)
489                         .FirstOrDefault();
490
491             foreach (var child in note.ChildNotes)
492             {
493                 DeleteOrphanAdventureNotes(child.AdventureNote_PK,
                      dbContext);
494             }
495
496             note.IsActive = false;
497             dbContext.Entry(note).Property("IsActive").IsModified = true;
498         }
499
500         #endregion
501
502         #region Delete Adventure
503
504         /// <summary>
505         /// Http Post to delete an adventure.
506         /// </summary>
507         /// <param name="adventure_PK"></param>
508         [Authorize, HttpPost]
509         public ActionResult DeleteAdventure(long adventure_PK)
510         {
511             ActionResult result = RedirectToAction("Index");
512
513             using (var dbContext = new ApplicationDbContext())
514             {
515                 var userId = User.Identity.GetUserId();
516
```

```
517                 // Find the adventure.
518                 var adventure = (from a in dbContext.Adventures
519                                  let player = a.Players.FirstOrDefault(p =>
                        p.UserId_PK == userId)
520                                  where a.Adventure_PK == adventure_PK
521                                     && a.IsActive
522                                     && player.IsActive
523                                     // Assure the user is the gamemaster.
524                                     && player.PlayerRole.PlayerRole_PK ==
                        (long)PlayerRole.PlayerRoleKey.Gamemaster
525                                  select a)
526                                  .Include(a => a.Items)
527                                  .Include(a => a.AdventureNotes)
528                                  .FirstOrDefault();
529
530                 if (adventure != null)
531                 {
532                     // disable the adventure.
533                     adventure.IsActive = false;
534
535                     // update the database.
536                     dbContext.Entry(adventure).Property
                        ("IsActive").IsModified = true;
537
538                     // Delete all sub-items.
539                     foreach (var item in adventure.Items)
540                     {
541                         DeleteOrphanItems(item.Item_PK, dbContext);
542                     }
543
544                     // Delete all notes associated with the adventure.
545                     foreach (var note in adventure.AdventureNotes)
546                     {
547                         DeleteOrphanAdventureNotes(note.AdventureNote_PK,
                        dbContext);
548                     }
549
550                     dbContext.SaveChanges();
551                 }
552             }
553
554             return result;
555         }
556
557         #endregion
558
559         #region Search
560         /// <summary>
561         /// Http Post for searching for an item in an adventure.
562         /// </summary>
563         /// <param name="adventure_PK"></param>
564         /// <param name="searchText"></param>
565         /// <returns></returns>
566         [Authorize, HttpPost]
567         public ActionResult AdventureSearch(long adventure_PK, string
            searchText)
```

```
568         {
569             ActionResult result = RedirectToAction("SearchResults", new { id
                  = adventure_PK });
570
571             using (var dbContext = new ApplicationDbContext())
572             {
573                 var userId = User.Identity.GetUserId();
574
575                 // Check to see if there is an exact match on the name.
576                 var results = (from i in dbContext.Items
577                                let player =
                        i.Adventure.Players.FirstOrDefault(p => p.UserId_PK ==
                        userId)
578                                where i.Adventure_PK == adventure_PK
579                                 && i.Name == searchText
580                                 && i.IsActive
581                                 && (i.Adventure.IsPublic || player.IsActive)
582                                select i)
583                                .ToList();
584
585                 if (results.Count() == 1)
586                 {
587                     // If there is only one exact match, go to the detail
                        page.
588                     result = RedirectToAction("ItemDetails", new { id =
                        results.First().Item_PK });
589                 }
590                 else
591                 {
592                     // If there are multiple matches, start the fuzzy search.
593                     // Get all items to search through.
594                     results = (from i in dbContext.Items
595                                where i.Adventure_PK == adventure_PK
596                                 && i.IsActive
597                                select i)
598                                .ToList();
599
600                     // Find a match on any names that contain the non-case-
                        sensitive text of the search.
601                     Regex regex = new Regex("^.*" + searchText.ToLower() +
                        ".*$");
602
603                     // Remove all items that do not match.
604                     results = results.Where(i => regex.IsMatch(i.Name.ToLower
                        ())).ToList();
605
606                     // Save the list for the view in temporary data
607                     TempData["searchResults"] = results;
608                     result = RedirectToAction("SearchResults", new { id =
                        adventure_PK });
609                 }
610             }
611
612             return result;
613         }
614
```

```
615         /// <summary>
616         /// Http Get for the search results page.
617         /// </summary>
618         /// <param name="id"></param>
619         /// <returns></returns>
620         [HttpGet]
621         [Route("adventure/{id:long}/Search/Results")]
622         public ActionResult SearchResults(long id)
623         {
624             // Get the results from temporary data.
625             var searchResults = TempData["searchResults"] as List<Item>;
626             // Create the view model.
627             var viewModel = new SearchResultsViewModel(null, searchResults);
628 
629             using (var dbContext = new ApplicationDbContext())
630             {
631                 var adventure = (from a in dbContext.Adventures
632                                  where a.Adventure_PK == id
633                                     && a.IsActive
634                                  select a)
635                                  // Get the list of items for the search
                        menu.
636                                  .Include(a => a.Items)
637                                  .FirstOrDefault();
638 
639                 if (adventure != null)
640                 {
641                     viewModel.Adventure = adventure;
642                 }
643             }
644 
645             return View(viewModel);
646         }
647 
648         #endregion
649 
650         #endregion
651 
652         #region Item Actions
653 
654         #region Create
655 
656         /// <summary>
657         /// HTTP Get. for creating an item. Route: ~/adventure/{adventureId}/
              Item/Create
658         /// </summary>
659         /// <param name="adventureId"></param>
660         [Authorize]
661         [Route("adventure/{adventureId:long}/CreateItem/{parentId:long?}",
              Name ="ItemCreate")]
662         public ActionResult ItemCreate(long adventureId, long? parentId)
663         {
664             Adventure adventure = null;
665             // Display the access invalid view if the adventure could not be
                found or the user does not have access to view it.
666             ActionResult result = View("AccessInvalid");
```

```
667
668             using (var dbContext = new ApplicationDbContext())
669             {
670                 string userId = User.Identity.GetUserId();
671
672                 adventure = (from a in dbContext.Adventures
673                              let player = a.Players.FirstOrDefault(p =>
                        p.UserId_PK == userId)
674                              where a.Adventure_PK == adventureId
675                                 && a.IsActive
676                                 && player.IsActive
677                                 // Ensure the user is a gamemaster.
678                                 && player.PlayerRole.PlayerRole_PK == (long)
                        PlayerRole.PlayerRoleKey.Gamemaster
679                              select a).FirstOrDefault();
680             }
681
682             if (adventure != null)
683             {
684                 // Create the default item with preset values.
685                 var model = new Item()
686                 {
687                     Adventure = adventure,
688                     Adventure_PK = adventure.Adventure_PK,
689                     ParentItem_PK = parentId,
690                     CreatedDate = DateTime.Now,
691                     LastModifiedDate = DateTime.Now,
692                     LastModifiedUser = User.Identity.GetUserName()
693                 };
694
695                 result = View("ItemCreate", model);
696             }
697
698             return result;
699         }
700
701         /// <summary>
702         /// HTTP Post for Creating a new Item.
703         /// </summary>
704         /// <param name="model"></param>
705         /// <returns></returns>
706         [HttpPost]
707         [Authorize]
708         [Route("adventure/{adventureId:long}/CreateItem")]
709         public ActionResult ItemCreate(Item model)
710         {
711             ActionResult result = View(model);
712
713             if (ModelState.IsValid)
714             {
715                 model.IsActive = true;
716
717                 using (var dbContext = new ApplicationDbContext())
718                 {
719                     dbContext.Items.Add(model);
720                     dbContext.SaveChanges();
```

```
721                 }
722
723                 if (model.ParentItem_PK != null)
724                 {
725                     // Return to the parent.
726                     result = RedirectToAction("ItemDetails", new { id =
                        model.ParentItem_PK });
727                 }
728                 else
729                 {
730                     // Return to the advenure view.
731                     result = RedirectToAction("Details", new { id =
                        model.Adventure_PK });
732                 }
733             }
734
735             // If we got this far, redisplay the view for corrections.
736             return result;
737         }
738
739         #endregion
740
741         #region Details
742         /// <summary>
743         /// HTTP Get for item details.  Route: ~/Item/id
744         /// </summary>
745         /// <param name="id"></param>
746         /// <returns></returns>
747         [Authorize]
748         [Route("Item/{id:long}", Name = "ItemDetails")]
749         public ActionResult ItemDetails(long id)
750         {
751             Item Item = null;
752             ActionResult result = View("AccessInvalid");
753
754             using (var dbContext = new ApplicationDbContext())
755             {
756                 string userId = User.Identity.GetUserId();
757
758                 Item = (from i in dbContext.Items
759                         let player = i.Adventure.Players.FirstOrDefault(p =>
                        p.UserId_PK == userId)
760                         where i.Item_PK == id
761                             && i.IsActive
762                             // Assure that the adventure is one the player
                        can view AND is not deleted.
763                             && i.Adventure.IsActive
764                             && (i.Adventure.IsPublic || player.IsActive)
765                         select i)
766                          .Include(w => w.ItemNotes.Select(n =>
                        n.ApplicationUser))
767                          .Include(w => w.ItemNotes.Select(n => n.ChildNotes))
768                          // Load 2 levels of children
769                          .Include(w => w.ChildItems.Select(c =>
                        c.ChildItems))
770                          .FirstOrDefault();
```

```
771             }
772
773             if (Item != null)
774             {
775                 result = View("ItemDetails", Item);
776             }
777
778             return result;
779         }
780         #endregion
781
782         #region Edit
783         /// <summary>
784         /// HTTP Get for edititng an item. Rotue: ~/item/id/Edit
785         /// </summary>
786         /// <param name="id"></param>
787         /// <returns></returns>
788         [Authorize]
789         [Route("Item/{id:long}/Edit", Name="ItemEdit")]
790         public ActionResult ItemEdit(long id)
791         {
792             Item Item = null;
793             // If there is an issue, go to the detail view of the item.
794             ActionResult view = RedirectToAction("ItemDetails", id);
795
796             using (var dbContext = new ApplicationDbContext())
797             {
798                 string userId = User.Identity.GetUserId();
799
800                 Item = (from i in dbContext.Items
801                         let player = i.Adventure.Players.FirstOrDefault(p =>
                        p.UserId_PK == userId)
802                         where i.Item_PK == id
803                             // Assure that the adventure is one the player
                        can edit AND is not deleted.
804                         && i.IsActive
805                         && i.Adventure.IsActive
806                         && player.IsActive
807                         // Ensure the user is a gamemaster.
808                         && player.PlayerRole.PlayerRole_PK == (long)
                        PlayerRole.PlayerRoleKey.Gamemaster
809                          select i).FirstOrDefault();
810             }
811
812             if (Item != null)
813             {
814                 view = View("ItemEdit", Item);
815             }
816
817             return view;
818         }
819
820
821         /// <summary>
822         /// Http Post for editing an item.
823         /// </summary>
```

```
824         /// <param name="model"></param>
825         /// <returns></returns>
826         [HttpPost, ValidateAntiForgeryToken, Authorize]
827         [Route("Item/{id:long}/Edit")]
828         public ActionResult ItemEdit(Item model)
829         {
830             ActionResult result = View(model);
831 
832             // If the model is valid, update database, and return to detail
                view.
833             if (ModelState.IsValid)
834             {
835                 using (var dbContext = new ApplicationDbContext())
836                 {
837                     // Update model values
838                     model.LastModifiedDate = DateTime.Now;
839                     model.LastModifiedUser = User.Identity.GetUserName();
840 
841                     // If everthing goes well, update database with listed
                    properties.
842                     dbContext.Entry(model).State = EntityState.Modified;
843                     dbContext.SaveChanges();
844                 }
845 
846                 result = RedirectToAction("ItemDetails", model.Item_PK);
847             }
848 
849             return result;
850         }
851         #endregion
852 
853         #region Delete Item
854         /// <summary>
855         /// Http Post for deleting an item.
856         /// </summary>
857         /// <param name="item_PK"></param>
858         /// <param name="adventure_pk"></param>
859         /// <returns></returns>
860         [Authorize, HttpPost]
861         public ActionResult DeleteItem(long item_PK, long adventure_pk)
862         {
863             // If it is successfully deleted, go to the adventure detail
                page.
864             ActionResult result = RedirectToAction("Details", new { id =
                adventure_pk });
865 
866             using (var dbContext = new ApplicationDbContext())
867             {
868                 var userId = User.Identity.GetUserId();
869 
870                 // Find the item.
871                 var item = (from i in dbContext.Items
872                             let player = i.Adventure.Players.FirstOrDefault(p
                    => p.UserId_PK == userId)
873                             where i.Item_PK == item_PK
874                                 && player.IsActive
```

```
875                                 // ensure the user is the gamemaster.
876                                 && player.PlayerRole.PlayerRole_PK == (long)
                        PlayerRole.PlayerRoleKey.Gamemaster
877                             select i).FirstOrDefault();
878
879                 if (item != null)
880                 {
881                     // Delete all child items.
882                     DeleteOrphanItems(item.Item_PK, dbContext);
883                     dbContext.SaveChanges();
884                 }
885             }
886
887             return result;
888         }
889
890         /// <summary>
891         /// Recursively delete all child items of an item.
892         /// </summary>
893         /// <param name="Item_PK">item to delete.</param>
894         /// <param name="dbContext">database context to use.</param>
895         private void DeleteOrphanItems(long Item_PK, ApplicationDbContext
              dbContext)
896         {
897             var item = (from i in dbContext.Items
898                         where i.Item_PK == Item_PK
899                         select i)
900                         .Include(i => i.ChildItems)
901                         .Include(i => i.ItemNotes)
902                         .FirstOrDefault();
903
904             // Delete childs of the children.
905             foreach (var child in item.ChildItems)
906             {
907                 DeleteOrphanItems(child.Item_PK, dbContext);
908             }
909
910             // Delete all notes on the item.
911             foreach (var note in item.ItemNotes)
912             {
913                 DeleteOrphanComments(note.ItemNote_PK, dbContext);
914             }
915
916             // Delete the item.
917             item.IsActive = false;
918             dbContext.Entry(item).Property("IsActive").IsModified = true;
919         }
920
921         #endregion
922
923         #region Comments
924         /// <summary>
925         /// Http Post for creating a comment.
926         /// </summary>
927         /// <param name="Item_PK"></param>
928         /// <param name="newComment"></param>
```

```
929         /// <param name="parentComment"></param>
930         /// <returns></returns>
931         [Authorize, HttpPost, ValidateInput(false)]
932         public ActionResult CreateItemComment(long Item_PK, string
              newComment, long? parentComment = null)
933         {
934             ActionResult result = RedirectToAction("ItemDetails", new { id =
                Item_PK });
935 
936             if (!string.IsNullOrWhiteSpace(newComment))
937             {
938                 var userId = User.Identity.GetUserId();
939 
940                 var note = new ItemNote()
941                 {
942                     Item_PK = Item_PK,
943                     UserId_PK = userId,
944                     ParentItemNote_PK = parentComment,
945                     Text = newComment,
946                     IsActive = true,
947                     CreatedDate = DateTime.Now,
948                     LastModifiedDate = DateTime.Now,
949                     LastModifiedUser = User.Identity.GetUserName()
950                 };
951 
952                 using (var dbContext = new ApplicationDbContext())
953                 {
954                     // Check if an item exists that meets the qualifications.
955                     var isPlayer = (from i in dbContext.Items
956                                     let player =
                                    i.Adventure.Players.FirstOrDefault(p => p.UserId_PK ==
                                    userId)
957                                     where i.Item_PK == Item_PK
958                                         && i.IsActive
959                                         && i.Adventure.IsActive
960                                         && player.IsActive
961                                     select i).Any();
962 
963                     if (isPlayer)
964                     {
965                         dbContext.ItemNotes.Add(note);
966                         dbContext.SaveChanges();
967                     }
968                 }
969             }
970 
971             return result;
972         }
973 
974         /// <summary>
975         /// Http Post for deleting an item.
976         /// </summary>
977         /// <param name="ItemNote_PK"></param>
978         /// <returns></returns>
979         [Authorize, HttpPost]
980         public ActionResult DeleteItemComment(long ItemNote_PK)
```

```
981         {
982             ActionResult result = RedirectToAction("Index");
983 
984             using (var dbContext = new ApplicationDbContext())
985             {
986                 var note = (from n in dbContext.ItemNotes
987                             where n.ItemNote_PK == ItemNote_PK
988                                 && n.IsActive
989                             select n)
990                             .Include(n => n.Item)
991                             .FirstOrDefault();
992 
993                 if (note != null)
994                 {
995                     // Ensure the user is the gamemaster.
996                     var isGamemaster = IsInRole(User.Identity,
                        note.Item.Adventure_PK,
                        PlayerRole.PlayerRoleKey.Gamemaster);
997 
998                     if (isGamemaster)
999                     {
1000                        // Delete the comment and all child comments.
1001                        DeleteOrphanComments(note.ItemNote_PK, dbContext);
1002                        dbContext.SaveChanges();
1003                        result = RedirectToAction("ItemDetails", new { id =
                           note.Item_PK });
1004                    }
1005                }
1006            }
1007 
1008            return result;
1009        }
1010 
1011        /// <summary>
1012        /// Http Post to edit the comment.
1013        /// </summary>
1014        /// <param name="ItemNote_PK"></param>
1015        /// <param name="commentText"></param>
1016        /// <param name="parentComment"></param>
1017        /// <returns></returns>
1018        [Authorize, HttpPost, ValidateInput(false)]
1019        public ActionResult EditItemComment(long ItemNote_PK, string
            commentText, long? parentComment = null)
1020        {
1021            ActionResult result = null;
1022            var userId = User.Identity.GetUserId();
1023 
1024            using (var dbContext = new ApplicationDbContext())
1025            {
1026                var note = (from n in dbContext.ItemNotes
1027                            where n.ItemNote_PK == ItemNote_PK
1028                            select n)
1029                            .Include(n => n.Item)
1030                            .FirstOrDefault();
1031 
1032                if (note != null)
```

```
1033                {
1034                    // Check if the adventure exists that meets the qualifications.
1035                    var isPlayer = (from a in dbContext.Adventures
1036                                    let player = a.Players.FirstOrDefault(p => p.UserId_PK == userId)
1037                                    where a.Adventure_PK == note.Item.Adventure_PK
1038                                        && a.IsActive
1039                                        && player.IsActive
1040                                    select a).Any();
1041
1042                    if (isPlayer)
1043                    {
1044                        note.Text = commentText;
1045
1046                        dbContext.Entry(note).Property("Text").IsModified = true;
1047                        dbContext.SaveChanges();
1048
1049                        result = RedirectToAction("ItemDetails", new { id = note.Item_PK });
1050                    }
1051                }
1052            }
1053
1054
1055            return result;
1056        }
1057
1058        /// <summary>
1059        /// Delete all child comments of a comment.
1060        /// </summary>
1061        /// <param name="ItemNote_PK"></param>
1062        /// <param name="dbContext"></param>
1063        private void DeleteOrphanComments(long ItemNote_PK, ApplicationDbContext dbContext)
1064        {
1065            var note = (from n in dbContext.ItemNotes
1066                        where n.ItemNote_PK == ItemNote_PK
1067                        select n)
1068                        .Include(n => n.ChildNotes)
1069                        .FirstOrDefault();
1070
1071            foreach (var child in note.ChildNotes)
1072            {
1073                DeleteOrphanComments(child.ItemNote_PK, dbContext);
1074            }
1075
1076            note.IsActive = false;
1077            dbContext.Entry(note).Property("IsActive").IsModified = true;
1078        }
1079
1080        #endregion
1081
1082        #endregion
```

```
1083
1084            #region Utilities
1085            /// <summary>
1086            /// Method to ascertain if a user is in a role for a specified adventure.
1087            /// </summary>
1088            /// <param name="user">User Id to check for.</param>
1089            /// <param name="adventure_Pk">Specified Adventure.</param>
1090            /// <param name="playerRoleKey">Player Role to check for.</param>
1091            /// <returns></returns>
1092            public static bool IsInRole(IIdentity user, long adventure_Pk, PlayerRole.PlayerRoleKey playerRoleKey)
1093            {
1094                bool isInRole = true;
1095                string userId = user.GetUserId();
1096
1097                using (var dbContext = new ApplicationDbContext())
1098                {
1099                    isInRole = (from p in dbContext.Players
1100                                where p.UserId_PK == userId
1101                                    && p.Adventure_PK == adventure_Pk
1102                                    && p.PlayerRole_PK == (int)playerRoleKey
1103                                    && p.IsActive
1104                                select p).Any();
1105                }
1106
1107                return isInRole;
1108            }
1109
1110            /// <summary>
1111            /// Check if a user is in any role for a specified adventure.
1112            /// </summary>
1113            /// <param name="user">User Id to check for.</param>
1114            /// <param name="adventure_Pk">Specified adventure.</param>
1115            /// <param name="playerRoleKeys">enumerable item list of player roles to check for.</param>
1116            /// <returns></returns>
1117            public static bool IsInAnyRole(IIdentity user, long adventure_Pk, IEnumerable<PlayerRole.PlayerRoleKey> playerRoleKeys)
1118            {
1119                bool isInAnyRole = false;
1120
1121                foreach (var key in playerRoleKeys)
1122                {
1123                    if (IsInRole(user, adventure_Pk, key))
1124                    {
1125                        isInAnyRole = true;
1126                        break;
1127                    }
1128                }
1129
1130                return isInAnyRole;
1131            }
1132
1133            /// <summary>
1134            /// Checks if a user is the original creator of a comment.
```

```
1135          /// </summary>
1136          /// <param name="user">User Id to check for.</param>
1137          /// <param name="AdventureNote_PK">key of adventure comment to check
                for.</param>
1138          /// <returns></returns>
1139          public static bool IsAdventureNoteOwner(IIdentity user, long
                AdventureNote_PK)
1140          {
1141              bool isAdventureNoteOwner = true;
1142              string userId = user.GetUserId();
1143
1144              using (var dbContext = new ApplicationDbContext())
1145              {
1146                  isAdventureNoteOwner = (from n in dbContext.AdventureNotes
1147                                          where n.AdventureNote_PK ==
                        AdventureNote_PK
1148                                              && n.UserId_PK == userId
1149                                          select n).Any();
1150              }
1151
1152              return isAdventureNoteOwner;
1153          }
1154
1155          /// <summary>
1156          /// Checks if a user is the original creator of a comment.
1157          /// </summary>
1158          /// <param name="user">User Id to check for.</param>
1159          /// <param name="ItemNote_PK">Key of adventure comment to check
                for.</param>
1160          /// <returns></returns>
1161          public static bool IsItemNoteOwner(IIdentity user, long ItemNote_PK)
1162          {
1163              bool isAdventureNoteOwner = true;
1164              string userId = user.GetUserId();
1165
1166              using (var dbContext = new ApplicationDbContext())
1167              {
1168                  isAdventureNoteOwner = (from n in dbContext.ItemNotes
1169                                          where n.ItemNote_PK == ItemNote_PK
1170                                              && n.UserId_PK == userId
1171                                          select n).Any();
1172              }
1173
1174              return isAdventureNoteOwner;
1175          }
1176          #endregion
1177      }
1178  }
```