**

**ExploreVA Maintenance Guide**

**June 2015**

**Introduction**

ExploreVA is a website created for the U.S. Department of Veterans Affairs (VA). It is hosted on VA servers, and is available to the public at http://explore.va.gov. This guide provides technical information on the site’s design and construction that may be useful for those who maintain or extend the site in the future.

***Audience***

This document is written for web developers. It assumes the reader is already familiar with databases, servers, programming and web design. In order to maintain or modify the site, a web developer should have knowledge and experience in these technologies:

* **Microsoft SQL Server**
* **Microsoft ASP.NET MVC 4 with C#**
* **Entity Framework**
* **JavaScript**
* **HTML5**
* **CSS3**

***Scope***

This guide focuses on the design and implementation details that are most likely to be useful for a developer who is taking over responsibility for the site. In many areas, the site’s design follows standard conventions for the industry and technologies involved. The emphasis in such cases is on the unique aspects of the ExploreVA site, and not on standard approaches.

**Hardware Architecture**

The production environment consists of multiple web servers and a single database server (Figure 1). These servers are entirely managed by VA’s Enterprise Web Infrastructure Support (EWIS) team. Consult EWIS for details on how to promote content into production.



*Figure 1: Production environment*

There is also a test environment hosted at VA. The test configuration consists of a web server and a database server (Figure 2). Both of these servers are shared with other VA test applications. Both servers are accessible only within the VA network. Access to the web server is via file transfer protocol (FTP). The database server can be updated using Microsoft’s SQL Server Management Studio.



*Figure 2: Test environment*

**Database**

ExploreVA uses a single SQL Server 2008 R2 database. In production, the web server has read-only access to the database, and no user generated content is stored there. Changes to the database are typically accomplished with SQL scripts. Each script is applied first to the test server to verify that it works correctly. Following successful testing, scripts are applied to the production database.

***Content Status***

Many tables in the database have a content status column, which defines whether specific content items are stored, published or deleted. Any table that has a ContentStatusId column is referencing the ContentStatus table, with IDs mapped as follows:

|  |  |
| --- | --- |
| Id | Title |
| 1 | Stored |
| 2 | Published |
| 3 | Deleted |

The web application will only presemt content with a ContentStatusId of 2 (Published). Other categories of content are retained for historical reasons, or in order to stage impending changes.

***Benefits and Benefit Categories***

The information architecture of ExploreVA is organized around 9 categories of benefits, which are defined in the BenefitCategories table:

|  |  |
| --- | --- |
| Id | Title |
| 1 | Spouses, Dependents, and Survivors |
| 2 | Disability Compensation |
| 3 | Education and Training |
| 4 | Employment Services |
| 5 | Health Care |
| 6 | Home Loans and Housing-Related Assistance |
| 7 | Life Insurance |
| 8 | Memorial Benefits |
| 9 | Pension |

It is important to know those 9 benefit IDs, since they are often used as foreign keys in other tables to identify which benefit area a resource applies to. The BenefitCategories table also stores a brief description of each benefit category, which displays on the main benefits grid in the home page, and a reference to the video that provides an overview of that benefit area. This video will appear in the “Learn More” link on the main benefit page in ExploreVA.

There is also a Benefits table, which captures individual benefits for which Veterans may be eligible. Each row in the Benefits table contains a reference code, title and description. The navigator logic is linked to specific benefits via the Conditions table. This process is described in the Benefits Navigator section.

***Videos***

Currently, there are three categories of videos on the ExploreVA site. The set of categories is defined by the VideoCategories table. The three current categories of video are:

|  |  |
| --- | --- |
| Id | Name |
| 1 | Veterans' Testimonial |
| 2 | How To |
| 3 | Benefits Overview |

So far, there are only testimonial and overview videos; no “how-to” videos have been released on the site (although there are plans to release some in the near future). Additional categories could be supported by adding more rows to VideoCategories.

The videos that appear for the “Learn More” link in a benefit category header are defined in the BenefitCategories table, as discussed above. Testimonial videos appear further down on each main benefit area page. The Testimonials table captures all of the information needed to link videos to individual benefit categories. It also contains a column that provides a link to the still image to present on the page, as well as a pull-quote and meta-data (name, branch and era of service) for the speaker(s).

All published videos appear in the Video Gallery and in the Outreach & Social Sharing Portal. The Videos table contains full information on each video. Here are the columns in the Videos table:

|  |  |
| --- | --- |
| Column Name | Description |
| VideoCategoryId | Type of video (testimonial, how-to or overview). References one of the rows in the VideoCategory table (1-3) |
| BenefitCategoryId | Which benefit area this video applies to. Value can be NULL if the video is not associated with a specific benefit category. |
| Title | This title appears below the video on the sharing portal and at the top of the video-specific sharing page. |
| Tweet | Text to use for sharing with Twitter. This text is also used with Pinterest, although in that case any initial “RT” is replaced with “Repin.” |
| Description | Appears below the video in the lightwindow and on the video-specific share page. |
| PublishDate | When the video was published to the site. Since videos are show in reverse chronological order on the video gallery and sharing portal pages, adjustments to PublishDate will also adjust the ordering of videos on the page. |
| FriendlyUrl | The unique portion of the URL for the video-specific sharing page. For example, if FriendlyUrl = ‘my-video’, the video-specific sharing page would be <http://explore.va.gov/video/my-video>. |
| VideoUrl | Provides the link to YouTube. Content should always be in this format:  //www.youtube.com/embed/ABCDEFGHIJK?rel=0  Where ABCDEFGHIJK is the unique 11-digit code that YouTube assigns. |
| Image | Still image for the video, also obtained from YouTube. The standard format for this field is:  //img.youtube.com/vi/ABCDEFGHIJK/0.jpg |
| ImageWidth | Width in pixels. Should be 480. |
| ImageHeight | Height in pixels. Should be 360. |
| ShowOnHomePage | Set to “1” to indicate the video should display for the “Watch Video” link on the home page. Only one published video should have this value set to 1; the others should all be zero (0). |
| ContentStatusId | Publication status of the video. |
| FacebookShare | Text to use for sharing on Facebook. |

The Videos table also includes three columns which are currently not used: MediaCampaignId (which references the MediaCampaign table, also unused), MetaInfo, and AudienceType.

To publish a new video, create a SQL insert statement along the following lines:

INSERT INTO [Videos] (

VideoCategoryId,

BenefitCategoryId,

Title,

Tweet,

Description,

PublishDate,

FriendlyUrl,

VideoUrl,

Image,

ImageWidth,ImageHeight,

ShowOnHomePage,

ContentStatusId,

FacebookShare

) VALUES (

1,

5,

'Three Veterans, three stories of women’s health.',

'RT to share the stories of Karla, Kim, and Natasha, who benefited from VA health care. #ExploreVA',

'Natasha, Karla, and Kim on service, identity, and choosing VA health care. Share their stories and help female Veterans learn more about the VA benefits they may be eligible for.',

'2015-05-22 13:00:00',

'womens-health',

'//www.youtube.com/embed/uvOxco20tCo?rel=0',

'//img.youtube.com/vi/uvOxco20tCo/0.jpg',

480,360,

0,

2,

'Three Veterans’ stories of health. Karla, Kim, and Natasha discuss service, identity, and VA health care. Share their stories and help female #Veterans learn more about the VA benefits they may be eligible for. http://explore.va.gov'

)

***Benefits Navigator***

The benefits navigator presents a series of questions that are designed to figure out which benefits one may be eligible for. Questions are assigned to specific pages, and the next page a participant views may depend on their answers to prior pages. The database tables that support the benefits navigator logic are shown below.



*Figure 3: Schema for Benefits Navigator*

The original design included support for storing a user’s answers in the database. Prior to public launch, however, the site was modified so it can work with a read-only database. As a result, the database no longer stores any user data. That is why the AnswerChoice and SystemUser tables are grayed out in Figure 3; they are not currently used for anything. The set of answers a user has given is now stored in a browser cookie instead.

The remaining tables shown in Figure 3 all support the question, answer and associated benefits logic of the Benefits Navigator. Starting in the center of the diagram, it is possible to see that a Page has one or more Questions, each of which has multiple Answers. The Navigation table captures paths between two pages, sometimes depending on whether a specific answer has already been given. Navigation rows that have NULL for the AnswerId represent the default path out of a given page. There should only be one default path in the table for each unique value of FromPageId. The default path is taken if none of the other paths applies (i.e. no AnswerIds already given in this session match against any outgoing paths).

Beginning at the upper right and moving counter-clockwise will show how benefits are tied to answer choices. Each of the 9 benefit categories may have multiple associated benefits. In turn, each benefit has one or more conditions that determine eligibility. Each Condition is a combination of specific answers; the condition is satisfied for a given user if they have provided all of the answers associated with the condition. The Benefit applies if at least one of its conditions is met. The benefits logic, therefore, is a sum of products[[1]](#footnote-1). Each condition represents a product (logical **and**) of its related answers, while the benefit logic sums the set of conditions (applying a logical **or**).

**Application**

The application is built with ASP.NET MVC 4, and written in C#. It uses Entity Framework as the object-relational mapping layer. The solution file contains two projects: one for the data layer (ExploreVA.Data) and one for the web application (ExploreVA.Web).

***Dependencies***

The web application uses Ninject[[2]](#footnote-2) for dependency injection (also known as inversion of control). Most of the controllers require a data context and cache manager, for which Ninject creates a single instance per request scope. Ninject is also used to create a singleton instance of the browser detection module.

Browser detection (via the open source library WURFL[[3]](#footnote-3)) checks the user agent string to determine if it is most appropriate to serve a desktop or mobile version of the site. Now that the site’s CSS is fully responsive, browser detection is not used, but it remains available in case it is needed to enable or disable specific site features in the future.

A few other free libraries are included in the solution. DDay[[4]](#footnote-4) provides the ability to create iCal compatible files (for adding an event to the user’s calendar). The application uses Log4Net as its logging framework. Elmah[[5]](#footnote-5) is included to support logging of unhandled exceptions.

***Benefits Logic***

There were two key requirements driving the design of the Benefits Navigator: it must be stateless and it must not store any user data in the database. A stateless implementation is important not only because it is most consistent with the MVC design philosophy, but also because the production site runs in a web farm environment, where any given web request may be handled by a different web server from the last request. Not storing user data in the database is a derived requirement. The security guidelines that EWIS had in place at the time of launch required web applications to connect to the database via a data source name (DSN) with Open Database Connectivity (ODBC). Entity Framework is not compatible with DSN or ODBC, so the only alternative to meet the EWIS security requirements was to make the ExploreVA database read-only.

The web application uses two browser cookies to hold the user’s session data while they are interacting with the Benefits Navigator. One (“pagesVisited”) stores a comma-separated list of all pages they have answered questions on so far. Since there is more than one path out of each page, knowing the actual path taken is important for two reasons. First, it helps determine the correct path when backtracking to an earlier page. Second, it improves the accuracy of estimating the percent completion for the survey as a whole. In cases where the user backs up, the abandoned parts of their current visit tree are pruned off, so the list only covers the currently active path from the first page to their current location.

The other cookie (“answerIdList”) stores a comma-separated list of all answers provided so far. Like the pagesVisited cookie, it only stores answers for the currently active path. Once the user completes the survey, their full list of answers is used to determine the applicable benefits. This list is encoded into a unique alphanumeric string, which is a URL parameter for the final page in the benefits navigator. That allows the user to bookmark their benefits summary page and return to it later. The application will decode the URL parameter and derive the list of answers, which it uses to re-create the list of benefits.

See the Memento class[[6]](#footnote-6) for details on how a list of answers is converted to an alphanumeric string and back. Since there are only about 200 unique answer choices to all questions, the full set of answers can be represented by an ordered sequence of ones and zeros, where one represents an answer that was chosen, and zero represents one that was not. This creates a large integer, which is encoded 6 bits at a time. Each encoded digit corresponds to one of 64 unique character choices.[[7]](#footnote-7) The actual logic is a little more complicated, since it includes a version indicator and the ability to compress and expand strings as needed.

***Events***

The events page is used to advertise upcoming outreach events, and to allow users to provide their email address for event related notifications. This page was originally designed to be entirely data-driven, so that all event-specific details come from the database. In practice, however, each event has had some unique requirements related to how it is depicted on the site. As a result, the current implementation of the events page pulls some information from the database, but also relies on the view and view model to provide additional customization. The page is capable of supporting multiple events that the user can sign up for at the same time, but in practice only one event has been published at a time.

Each event has its own email list in GovDelivery that is used to collect emails, and notify users with reminders as the event date approaches. There is also a GovDelivery list for the series of outreach events, and of course the overall VA email list that currently has about a million subscribers. When users sign up for the event-specific email list, they can choose to subscribe to one or both of the other lists as well. All sign-ups are handled through the GovDelivery web API.

A new event can be added to the database via SQL insert statement like this:

INSERT INTO CalendarEvents(

[Title],

[Start],[End],[TimeZone],

[Location],[LocationUrl],

[Description],

[IsAllDay],[FileName],

[SignUpList],

[Hangout],

[InnerDescription]

) VALUES(

'#ExploreVA Twitter Chat: VA Memorial Benefits Honor Veterans',

'2015-05-20 15:00:00', '2015-05-05 16:00:00', 'EDT',

'https://twitter.com/search?q=%23HonorAVeteran&src=typd', '',

'Join VA, American Legion representatives, and others in the Veteran community for a Twitter chat about VA memorial benefits.',

0, 'ExploreVA-MemorialBenefits-May2015',

'USVA\_61',

'https://twitter.com/search?q=%23HonorAVeteran&src=typd',

'#ExploreVA Twitter Chat: Join VA, American Legion representatives, and others in the Veteran community for a Twitter chat about VA memorial benefits. ||https://twitter.com/search?q= %23HonorAVeteran&src=typd'

)

GO

For each event, it is necessary to supply both a start and end date, as well as the exact time zone (for example, “EDT” or “EST”). Those values are used to create iCal downloads, as well as to display the event time on the page. For physical events (taking place at a conference center, for example), supply both Location and LocationUrl if appropriate. The Location is the name of the location, and the URL points either to a web site or an online map for details. Most events have been online-only, and in that case only the Location field is used, while LocationUrl is left blank, as in the example above. The FileName column is used to name the iCal file that users can download. Typically each event should have a unique file name, although this rule is flexible.

The SignUpList column stores the unique ID for the GovDelivery list of the event itself (in this case, USVA\_61). The Hangout column stores a link to the online collaboration location, such as a Google Hangout, Twitter chat, or YouTube event.

The Description field is the main event description that appears on the page. HTML mark-up can be included if necessary (such as <ul> tags for a bulleted list). The InnerDescription column provides the text to include inside the downloadable iCal reminder. Use a pipe symbol (“|”) to represent a new line. Do not include any HTML markup in the InnerDescription.

It is also a good idea to look at the view (Events.cshtml) and make any necessary updates there. In particular, look at the code block at the top of the view, and adjust the string variable “typeOfEvent” as needed. That variable is used to show the correct icon on the sign up button, depending on where the event is hosted. Valid values are as follows:

* facebook-chat
* twitter-chat
* crowdhall
* youtube

Another part of the view that has often been customized for each event is the text that appears in the “Join” buttons. There are two such buttons – one that displays on and before the event date, and another that displays after the event has passed. Look for <a> tags with class “join-event” to find these buttons in the view.

Some of the text for the events page comes from the EventsViewModel class. The properties you can change inside this view model are:

|  |  |
| --- | --- |
| Property | Example value |
| SeriesName | From Service to Success |
| SeriesBlurb | An Online Event Series About Benefits for Veterans |
| SeriesDescription | Millions of Veterans and their family members are successfully using VA benefits to buy homes, earn degrees, start careers, stay healthy, and do so much more in life after the military. At these events, learn how Veterans have gone <b>from service to success</b> |
| Quote | I can’t think of a better place for [my dad] to be resting. |
| Attribution | Mary, on the VA cemetery where her father, a Navy Veteran, is buried |

The first three properties apply to the whole series, and tend not to change much. The final two (quote and attribution) will change for each event. It is OK to leave the quote and/or attribution blank, in which case they will not appear on the page. Do not include quote marks in the quote, since those are supplied by the CSS style applied on the page.

The header image on the Events page is determined by CSS. Here is the current CSS rule for that heading area (currently found in main.css, around line 4429):

body.outreach-portal\_\_events .row.row-event.event-1 {

background-image: url(../images/event-images/event-1.jpg); }

To select a different image, first add the image to the “event-images” folder under “Content/images”, then update the CSS to set the background-image attribute to reference the new image. Make sure the image dimensions are consistent with the current background image for the page.

To remove the Events page from the site navigation altogether, it is only necessary to update two partial view files. Both files are located under Views\Shared. The first one defines the layout of the Explore main menu, \_ExploreVAMenu.cshtml. The second file has the markup for the tabs in the sharing portal area, and it is \_PortalTabs.cshtml. Either remove or comment out the relevant list item (<li>…</li>) in both partial views.

1. It is a fundamental theorem of Boolean algebra that any logical expression can be represented as a sum of products, which means that this combination of tables can capture any set of eligibility rules, regardless of their complexity. [↑](#footnote-ref-1)
2. http://www.ninject.org [↑](#footnote-ref-2)
3. http://wurfl.sourceforge.net [↑](#footnote-ref-3)
4. http://sourceforge.net/projects/dday-ical/ [↑](#footnote-ref-4)
5. http://code.google.com/p/elmah/ [↑](#footnote-ref-5)
6. Found in the Utilities directory. [↑](#footnote-ref-6)
7. Since 26 = 64. [↑](#footnote-ref-7)