AP$_{I}temSampler$

Release 1.0.0

AP$_{I}temSampler$

Nov 08, 2019

Developer Guide

CHAPTER 1

Application Overview

The Sample Items Website is a Microsoft ASP.NET Core MVC web application that displays sample test question items.

1.1 License

Mozilla Public License Version 2.0

1.2 Installation

- Clone the project
  - Don't forget to initialize the project submodules (instructions)
- Install the latest version of Microsoft .NET Core for your operating system (link)
- Install the latest TypeScript compiler (link)
- Download the latest sample item [ftp://ftps.smarterbalanced.org/~sbacpublic/Public/PracticeAndTrainingTests/{]}(content package).
  - Place this in the location specified in src/Web/appsettings.json (`ContentRootDirectory`)

1.3 Project Architecture

The Sample Items Website project is composed of three layers: Web, Core, and Dal.

Web is the startup project and contains controllers, views, style, and other web dependencies. See the **Dependencies** section for more information.

Core provides business logic to Web via repositories. It also houses the business logic for the diagnosticAPI and model translations.

Dal provides data to the Core layer. All data is supplied by a "content package" which is a set of XML files that represent test question items. This XML is parsed into the immutable `ItemDigest` and `ItemCard` models, which are used throughout the application.

Other XML data is used, such as `AccessibilityAccommodationConfigurations`, `ClaimConfigurations`, and `InteractionTypeConfigurations`. These configure information such as order and labels for accessibility and interaction types.

Test is the test project for Sample Items Website. It contains unit tests and integration tests for each of the three aforementioned layers.

CHAPTER 2

Project Setup

2.1 General Steps

- Clone the project
  - Don't forget to initialize the project submodules (instructions)
- Install the latest version of Microsoft .NET Core for your operating system (link)
- Install the latest TypeScript compiler (link)
- Download the latest sample item [ftp://ftps.smarterbalanced.org/~sbacpublic/Public/PracticeAndTrainingTests/{]}(content package)
  - Place this in the location specified in src/Web/appsettings.json (`ContentRootDirectory`)

2.2 Running

2.2.1 Using Visual Studio

- Open the project in Visual Studio 2015-17
- Click the run button

2.2.2 Using Command Line

- Set environment variable `ASPNETCORE_ENVIRONMENT` to `Development`
  - in Windows: `setx ASPNETCORE_ENVIRONMENT "Development"`, then close and reopen the command prompt
- `cd SampleItemsWebsite\SmarterBalanced.SampleItems`
- `dotnet restore`

- `cd src\SmarterBalanced.SampleItems.Web`
- `dotnet run`
- Navigate to `http://localhost:<port>` in your browser to view the running site

CHAPTER 3

Application Settings and Configurations

Within the `SmarterBalanced.SampleItems.Web` project, configurations are set in the file `appsettings.json`. This file contains all of the configurations and settings that are used throughout the application.

Within `appsettings.json`, the `SettingsConfig` object contains configurations necessary to start the application, as well as runtime configuration settings.

Note: Following is a description of important configurations within this object:

`ContentItemDirectory`: Location of the **Items** directory within the content package. Dependent on deployment environment setting (development, staging, production).

`ContentRootDirectory`: Location of the content package. Dependent on deployment environment setting (development, staging, production).

`AwsS3Bucket`: AWS S3 bucket that contains the content package. Required for the diagnostic status feature.

`AwsRegion`: AWS region. Required for the diagnostic status feature.

`ItemViewerServiceURL`: Base URL for itemviewerservice, which renders the test question items. URL is used to display an iframe of each item.

`AwsClusterName`: Name of AWS cluster. Required for the diagnostic status feature.

`StatusUrl`: Diagnostic status URL for local diagnostic status. Required for the diagnostic status feature.

`AccommodationsXMLPath`: Location of the accessibility configurations XML document.

`InteractionTypesXMLPath`: Location of the interaction types configuration XML document.

`ClaimsXMLPath`: Location of the claims configuration XML document.

Additionally, the `RubricPlaceHolderText` object contains strings that are filtered out of item

CHAPTER 4

Build

Sampleitems Website requires content before running.

There is a two-step process using docker. The base app without content needs to be created as a docker image called code. Then we combine the code image with the content package using another dockerfile.

4.1 Before starting build

Locate the dockerfile.stage and dockerfile.prod files from github repo, navigate to deployScripts

4.2 Using a previous docker code image

To build a docker image for sample items website code base (without content) see Building from scratch

4.2.1 Docker

1. Navigate to directory containing dockerfiles
2. Get docker code repo for stage/prod, run `docker pull reponame:{tag}`
    1. Example stage: run `docker pull xxx.dkr.ecr.us-west-2.amazonaws.com/sampleitemscode:stage`
    2. Example producation: run `docker pull xxx.dkr.ecr.us-west-2.amazonaws.com/sampleitemscode:prod`
3. Docker tag code, run `docker tag reponame:{tag} sampleitemscode:{tag}`
    1. Example stage: run `docker tag xxx.dkr.ecr.us-west-2.amazonaws.com/sampleitemscode:stage sampleitemscode:stage`
    2. Example production: run `docker tag xxx.dkr.ecr.us-west-2.amazonaws.com/sampleitemscode:prod sampleitemscode:prod`

4. place content within deployScripts directory
    1. Content needs to be unzipped
    2. content directory root level needs Items directory
5. Docker build, run `docker build -t sampleitemsapp:{tag} -f Dockerfile.{tag} .`
    1. Example stage: run `docker build -t sampleitemsapp:stage -f Dockerfile.stage .`
    2. Example production: run `docker build -t sampleitemsapp:prod -f Dockerfile.prod .`
6. Docker Run app , run `docker run -it -p 8012:80 sampleitemsapp:{tag}`
    1. Example stage: run `docker run -it -p 8012:80 sampleitemsapp:stage`
    2. Example production: run `docker run -it -p 8012:80 sampleitemsapp:prod`
7. Go to localhost:8012
8. point to docker

4.3 Building from scratch

4.3.1 Dependencies

1. See *Project Dependencies*
2. Install Nodejs and npm

4.3.2 Dotnet build and Publish

1. `cd SampleItemsWebsite\SmarterBalanced.SampleItems`
2. `dotnet restore`
3. `cd src\SmarterBalanced.SampleItems.Web`
4. `npm install`
5. `grunt all`
6. `dotnet publish -o ../../publish`
7. `cd ../../publish`

4.3.3 Docker build

1. Go to the publish directory containing the dockerfile
2. Build app, run `docker build -t sampleitemscode:{tag}`
    1. Example stage: run `docker build -t sampleitemscode:stage`
    2. Example producation: run `docker build -t sampleitemscode:prod`

4.3.4 Deploy Sample Items app

1. see *Publish Docker*

CHAPTER 5

Dependencies

5.1 NPM

NPM is required to install Grunt, Grunt packages, Less, and TypeScript. See `package.json` in the **Web** project for configurations.

5.2 Bower

Bower is required to add Bootstrap, JQuery, and React. See `bower.json` in the **Web** project for configurations.

5.3 Grunt

Grunt is used to perform tasks with Visual Studio events as well as with build events. Grunt:

- Compiles TypeScript files into JavaScript on save and build
- Compiles Less CSS into CSS on save and build
- Minifies CSS on build
- Removes temp files

For Grunt configurations, see `Gruntfile.js` in the **Web** project.

5.3.1 External Dependencies

5.4 Docker

Docker is used to simplify and unify the build and runtime environment for the application.

See `Dockerfile` in the **Web** project for Docker configurations. The **Web** project also contains a `.dockerignore`.

5.5 TravisCI

TravisCI is used to verify the state of the application every time a developer pushes to a branch in the GitHub repository. It first installs project dependencies, pulls and builds the project, and then runs tests.

After the previous steps succeed, Travis builds a Docker image with the application and pushes it to AWS.

On the **dev**, **stage**, and **master** branches, Travis triggers a build process in AWS that combines the content package, hosted in an S3 bucket, with the application Docker image and performs a rolling update to the running versions of these branches.

See the `travis.yml` file in the root directory project for configurations.

5.6 Content package

The content for the site is provided by a content package supplied by the Smarter Balanced Assessment Consortium. It consists of XML files that describe the test question items (the "content package").

Currently, content packages can be accessed at ftp://ftps.smarterbalanced.org/~sbacpublic/Public/PracticeAndTrainingTests/.

CHAPTER 6

Docker

6.1 Publish Docker to AWS

1. Go to Amazon ECS
2. Select Repositories
3. Select a repository or create
4. Select push Commands
5. Follow the push Commands or follow:
    1. Go to the root directory containing Dockerfile
    2. Run `aws ecr get-login --region us-west-2`
    3. Run the docker login command
    4. Run `docker build -t {repo-name}:{latest/dev/stage/prod} .`
    5. Run `docker tag {repo-name}:{latest/dev/stage/prod} {amazon-repo}:{latest/dev/stage/prod}`
    6. Run `docker push {amazon-repo}:{latest/dev/stage/prod}`

6.2 Publish Docker to DockerHub

1. Go to the root directory containing Dockerfile
2. Run `docker login`
3. Run `docker build -t {repo-name}:{latest/dev/stage/prod} .`
4. Run `docker push {repo-name}:{latest/dev/stage/prod}`

6.3 Docker Commands

To update a docker image, please follow publish to Aws or DockerHub

CHAPTER 7

Accessibility Resources

- See ISAAP docs

7.1 Terms

- Accessibility Family: Default accessibility Resources available to the item based on the item's grade and subject
- Accessibility Resource: The group of accessibility options.
  - Example: Dictionary, Calculator
- Accessibility Option: Accessibility feature code. An ISAAP code to enable/disable accessibility.
  - Example: Calculator has Off and On options
- ISAAP: A unique code for accessibility. Each code has a `TDS` prefix.
  - See docs

7.2 AP_ItemViewerService

7.2.1 Usage

Please see github repo AP_ItemViewerService

7.2.2 Dictionary and Thesaurus

This is an external api for merriam-webster and depends on a running instance of TDS_Dictionary. AP_ItemSampler does not have any overrides or connection settings for this instance.

7.3 AP_ItemSampler usage

7.3.1 Client (Web)

Item Page and About Test Items use the AP_ItemViewerService as an iframe. Item Sampler passes a list of items, need to be related, with isaap codes. These codes are generated within the accessibility modal using the local item's accessibility options. Items and Item's family accessibility can be loaded from the Item Sampler's api. The api accepts requests in the following format:

- "https://siw-ivs.smarterbalanced.org/item/{bankKey}-{itemKey}"

Access the api https://siw-ivs.smarterbalanced.org/item/

7.3.2 DAL

Accessibility is loaded from xml sources from a git submodule Github. Item Sampler loads them as is then applies business logic to create a complete list of accessibility resources for each family.

XML

Provides a global list of all accessibility resources, options, groups, and families. Families are a list of rules to apply to the global list for the subject and grade range.

Accessibility Resource Groups

This is the label for the group. Labels are loaded from the AppSettings.json and matched on the xml group key.

Merged Accessibility Family

This contains a list of accessibility resources for a range of grades and a subject. This contains the default options supported for a subject (ELA) and a grade range (3-5). Accessibility resource families from the xml contain rules to enable and disable options but not the options themselves. To simplify this process, merged accessibility families were created to execute the rules on the global list and contain the final result.

7.3.3 Item Accessibility

Local item accessibility is calculated from the accessibility family resource, based on subject and grade, and the item metadata attributes.

Item's that can have different accessibility than the family:

- Metadata attribute AllowCalculator
- Item types
- Performance Task
- Thesaurus and Dictionary
- Braille (based on the ftp server resources)
- Metadata attachements (ASL)
- Claim

Calculator

Metadata can enable and disable calculator based on `AllowCalculator`. Calculator only shows Off and On for all math items. On refers to the item's specific calculator option based on grade.

Thesaurus

Can be disabled if calculator is not allowed/available and if the item type is not WER.

Braille

Can be enabled if the ftp server has a listing for the item

American Sign Language

Can be enabled if the item's contents has a ASL video attachment

Closed Captioning

Can be enabled if item is subject ELA and claim 3

Global Notes

Can be enabled if item is a performance task

7.4 Options

7.4.1 Universal Tools

Digital Notepad

Displays a notepad in the item's hamburger menu. This will open a dialog for a user to enter text and save.

English Glossary

Provides glossary definition for an item's word list. This will show a dashed top and bottom border for words that have a glossary definition.

Highlighter

Allows highlighting the item and/or passage's text. To highlight, select text and right click or use the hamburger menu then select highlight text. To remove, select highlighted text and from the menu, select remove highlight.

English Dictionary

Provides a dialog to search the merriam-webster api for definitions. When enabled, a toolbar icon will display with text Dictionary.

Expandable Passages

Allows the passage to expand over the item. To enable, an icon will appear to the left of the hamburger menu for the passage, select the expand icon.

Global Notes

Provides a dialog to enter global notes for the user's session. When enabled, a toolbar icon will display.

Strikethrough

Allows strike-through text for the item and/or passage's text. To enable, select text and right click or use the hamburger menu and then select strikethrough. To remove, select the striked-through text and from the menu select remove strikethrough. (TODO: Verify name in menu.)

Thesaurus

Provides a dialog to search the merriam-webster api thesaurus. When enabled, a toolbar icon will display with text Dictionary. Dictionary enabled accessibility option is required for thesaurus.

Zoom

Increases/decreases text size for both passage and item. This should always be enabled. Two toolbar icons will appear.

7.4.2 Designated Supports

Color Choices

Changes the background color and text color for the item and passage.

Masking

Creates a overlay on top of the item and/or passage. A toolbar icon will display with text Masking. To enable, select the masking icon. To dismiss, select the x icon in the upper right of the masked overlay.

Translations (Glossaries)

Provides glossary definition for an item's word list. This will show a dashed top and bottom border for words that have a glossary definition for the selected language.

Translations (Stacked)

Provides translated item and passage text.

7.4.3 Accommodations

American Sign Language

Displays a dialog with an ASL video to describe item and passage. To enable, select the option from the hamburger menu.

Braille Type

Provides braille files for the item and/or passage to be printed. Above the item viewer, a new link will be displayed.

Closed Captioning

Displays a dialog on the bottom of the item viewer. TODO: add more info

Streamlined Interface

Changes the layout of the item and passage. When enabled, the passage will be above the item.

CHAPTER 8

Rubric and Rationale

8.1 Content Package

Applies to item-{bankKey}-itemKey.xml files within the content directory that are items. There are two types of "scoring" information. Rubric and Rationale list. Rationale list seems to be multiple choice explanations of the correct/incorrect answer. Rubric list has two parts, samples and rubrics. Rationale list also has rationale and option lists. There can be many sets based on the language.

8.1.1 Other Scoring Attributes

- `MachineRubric`: Specifies the machine rubric qrx file for automated scoring.
- `itm_att_Answer Key`

8.1.2 Rubric List

Rubric contains rubric entries and sample entries. Rubric entries are listed for each possible point value. There is usually many sample lists within a rubric list even though most sample lists contains one entry. Sample are example responses to get a certain point value. Rubrics are the requirements to achieve a specific point value.

- `content` > `rubriclist` > `rubric`
  - Score Point - entries point value
  - Name - specifies rubric name with point
  - Val - html of response
- `content` > `rubriclist` > `samplelist` > `sample`
  - Purpose
  - Name - specifies sample name with point value
  - Annotation * not mapped

  - Sample Content - html of explanation

8.1.3 Rationale List

Multiple choice and rationale for each possible answer. Note: Not mapped Explanation of Correct Answer and rationaloptlist

- Not Mapped - `content` > `rationaleoptlist` > `rationale`
  - Name - each option value (a,b, . . .)
  - Val - html of choice option (usually blank)
- `content` > `optionlist` > `option`
  - Name - each option value (a,b, . . .)
  - Val - html of choice option
  - Feedback - html explanation of chosen answer

8.2 Item Sampler Translation

`ItemDigest` is made up of two files, `ItemContents` and `ItemMetadata`. Item Digest is the merged result of the two files in the content package. `ItemContents` has `ItemXmlFieldRepresentation` of the Item. This has the list of `Content` which holds all Rubric and Rationale List. `SampleItem` is the final version of the translated content package.

8.2.1 `ItemDigest` - `ItemContents` and `ItemMetadata`

- `rubriclist` from content is mapped to `RubricList`
  - `rubric` from content is mapped to `RubricEntry`
  - `samplelist` is mapped to `RubricSample`
- `optionlist` > `option` is mapped to `SmarterAppOption`

8.2.2 `SampleItem` from `ItemDigest`

`ItemDigest` is translated to `SampleItemScore`. Please see code documentation for more information. `SampleItemScore` combines rubric and rationale for easier consumption in the API. It also filters out placeholder text and identifies correct/incorrect options.

CHAPTER 9

Indices and tables

- genindex
- modindex
- search