­­­­­Branding Test Cases

# Intro: the functions in question

## These functions get data and have no side-effects

**metadata**(conns, brand\_id)

**get\_active**(conns)

**list**(conns, include\_default**=**True)

## These functions create or update branding data

**clone**(conns, brand\_id, new\_name)

**new**(conns, brand\_name, prod\_name\_long, prod\_name\_short, company\_name,

      primary\_color, secondary\_color, custom\_css, background\_img\_fd,

      logo\_img\_fd)

**rename**(conns, brand\_id, new\_name)

**update**(conns, brand\_id, new\_settings)

**activate**(conns, brand\_id)

**delete**(conns, brand\_id)

## These exceptions can be thrown

* NoSuchBrandError – the brand by the given name or ID does not exist.
* BrandNameTakenError – the given brand name is already taken & cannot be used again.
* BadColorError – A brand's color field is not a valid CSS color.
* BadWebImgError – thrown when a web-displayable image file is not recognized as an image or not an image format that can be displayed on the web.
* DeletionCriteriaViolation – thrown when someone tried to delete the "SparkWeave default" brand.

# Details: Test Cases

## metadata

**def** **metadata**(conns, brand\_id):

  """

@param brand\_id: a string for a user-created brand or None to use the

SparkWeave default

@return: the metadata for the given brand  
 EXAMPLE:

{

'id': "ttQctmyjN4kjyWyet2gwYE",

'name': "gna bold",

'active': True, # only one brand can be active at a time.

'prod\_name\_long': "Guns 'n Ammo Secure Collaboration Suite",

'prod\_name\_short': "GnA Secure Email",

'company\_name': "Guns 'n Ammo",

'primary\_color': "#663300", # brown

'secondary\_color': "#666633", # green

'custom\_css\_path': '/media/brand/css/custom.css' # or None

'header\_path': '/media/brand/imgs/leaderlogo.png' # or None

'background\_path': '/media/brand/imgs/background\_sides.png' # or None

'css\_hash': "fBM5r5W8bSBMZB2sAR7Jea"

'mod\_time': "2012-07-30 15:37:31"

'create\_time': "2012-03-05 13:44:59"

'media\_version': "fBM5r5W8bSBMZB2sAR7Jea" # hash of fields related to media

# a change indicates a need for

# fresh media ditribuition

}

"""

1. Test that NoSuchBrandError is thrown if the given brand ID cannot be found.
   1. In the frontend – the brand settings page should correctly handle being loaded w/ a URL whose brand\_id param is something made up)
2. Metadata is returned even if brand\_ID is None – returned data should be for the SparkWeave default brand
   1. This doesn't translate to any frontend behavior other than that the brand list table should show the correct information for the default brand too.
   2. The default brand should NOT be editable.

## Get\_active

**def** **get\_active**(conns):

  """

@return the metadata of the active brand OR the metadata of the default brand

if the one listed as active cannot be found.

"""

1. If no brand has been explicitly set as the active brand, the "SparkWeave default" brand is considered active (call returns None)
2. If a user-create brand was set as active and the brand was deleted, the "SparkWeave default" brand is once again considered active (call returns None)
3. If a user-create brand was set as active, that brand's ID is returned as active.
4. If a user-create brand was set as active and then that brand is cloned, the original brand ID is still considered active – not the clone.

## List

**def** **list**(conns, include\_default**=**True):

  """ List all saved brands

@param include\_default: If True (the default) includes the

"SparkWeave default" brand.

@return: a list of metadata dictionaries for all brands

"""

1. The "SparkWeave default" brand is returned as the single item in the list if nothing else has been added.
2. Correct ordering: the "SparkWeave default" brand shows up first & after that the list is otherwise alphabetized.

## Clone

**def** **clone**(conns, brand\_id, new\_name):

  """ Clones an existing brand into a new sandbox area. The new brand gets its

own name, ID, mod time, and creation time, but is otherwise identical to the

original brand.

@param brand\_id: the ID of the brand to clone

@param new\_name: the brand name to give to the newly created brand

@return: metadata for the newly created brand

"""

1. Test that NoSuchBrandError is thrown if the given brand ID cannot be found.
   1. In the frontend – the clone dialog should show a readable error message.
2. BrandNameTakenError is thrown if the name for the new brand is already taken.
   1. In the frontend – the clone dialog should show a readable error message.

## New

**def** **new**(conns, brand\_name, prod\_name\_long, prod\_name\_short, company\_name,

      primary\_color, secondary\_color, custom\_css, background\_img\_fd,

      logo\_img\_fd):

  """

Creates a new brand with the given name, text, colors, and media.

@param brand\_name: the name of the new brand. Must be unique. Must not be

"SparkWeave default" as this is reserved.

@param prod\_name\_long: replacement for "SparkWeave Secure Collaboration Suite"

@param prod\_name\_short: replacement for "SparkWeave"

@param company\_name: replacement for "SparkWeave, LLC"

@param primary\_color: replaces the SparkWeave teal

a CSS-compatible color string. rgb(102, 234, 98) or #FF6631

@param secondary\_color: replaces the SparkWeave orange. Same format as

primary\_color.

@param custom\_css: a string containing custom CSS to load on every page.

@param background\_img\_fd: a file descriptor for a repeating background image.

@param logo\_img\_fd: a file descriptor for a header logo image.

@return: metadata for the newly created brand.

"""

1. Test that NoSuchBrandError is thrown if the given brand ID cannot be found.
   1. In the frontend – "name" field on the brand settings page should have a readable error shown by it after the form is submitted.
2. If primary\_color or secondary\_color is not a valid CSS color string, then thrown BadColorError.
   1. In the frontend – "primary color" or "secondary color" field on the brand settings page should have a readable error shown by it after the form is submitted.
3. If the uploaded background image or logo image is not a valid image file, BadWebImgError – should be thrown.
4. The following image types should be valid for upload: PNG, GIF, JPG/JPEG, TIF
5. If a non-image file (such as Word) is renamed to have a .png or .jpg file extension, the user should still see a error msg by the field telling them that their upload file is not recognized as an image.
6. Some other image types will not be accepted (see the following [page on image file types](http://en.wikipedia.org/wiki/Image_file_formats) & spot check a few)
7. If the brand name contains illegal characters / exceed # of allowable characters or if the brand name is already taken, it should show an error message after the form is submitted.

Some i/p values for brand name-

1. “ “ -fail
2. “ leading spaces brand” –pass with leading spaces trimmed
3. “trailing spaces brand name “- pass with trailing spaces trimmed
4. “123” –pass
5. “abcd”- pass
6. Test combinations of the following characters “~`!@#$%^&\*()\_+-=[]\{}|;’:”,./<>?”
7. Test for long brand names- test lowest, highest and one intermediate character limit test case
8. Product name and company name should show an error message after the form is submitted, if illegal characters/ exceed # of allowable characters used.

Some i/p values for product and company name-

1. “ “ -fail
2. “ leading spaces brand” –pass with leading spaces trimmed
3. “trailing spaces brand name “- pass with trailing spaces trimmed
4. “123” –pass
5. “abcd”- pass
6. Test combinations of the following characters “~`!@#$%^&\*()\_+-=[]\{}|;’:”,./<>?”
7. Test for long brand names- test lowest, highest and one intermediate character limit test case
8. Primary & secondary color: Test for illegal characters & values. Appropriate error messages should pop up if illegal characters entered. Test that the following i/p values are invalid-
9. Abcd
10. 1234
11. ()\_+
12. #$%^&
13. ~!`
14. …….
15. And the list goes on .. Maybe restrict the users to enter ONLY letters, numbers & # ???
16. Test the following valid image types- PNG, GIF, JPG/JPEG, TIF. Test if files with small case and capital case extensions work right. E.g foo.jpg, foo.JPG & foo. Jpg
17. If a non-image file (such as Word) is renamed to have a .png , .PNG or .jpg , .JPG file extension, the user should still see an error msg by the field telling them that their upload file is not recognized as an image.
18. Some fields missing and “preview” selected. Error message should be seen.
19. Cancel should NOT create the new brand and discard any changes that were made.
20. “Save changes”/”Create” should create the new brand.
21. Header logo and/or custom background pattern is valid but dimensions not enough. Still show it correctly/in stretched format or reject it?

## Rename

**def** **rename**(conns, brand\_id, new\_name):

  """ Give an existing brand a new name.

@return: the metadata for the given brand after the rename.

1. """
2. BrandNameTakenError is thrown if the name for the new name of the brand is already taken.
3. NoSuchBrandError is thrown if the given brand ID cannot be found.
4. name is new\_name afterwards if all successful. mod\_time has also changed, but no other fields.
   1. media\_hash does not change after the rename (media does not need to be redistributed)
5. Test for the following brand names/company name i/p values-
6. “ “ -fail
7. “ leading spaces brand” –pass with leading spaces trimmed
8. “trailing spaces brand name “- pass with trailing spaces trimmed
9. “123” –pass
10. “abcd”- pass
11. Test combinations of the following characters “~`!@#$%^&\*()\_+-=[]\{}|;’:”,./<>?”
12. Test for long brand names- test lowest, highest and one intermediate character limit test case

## Update

**def** **update**(conns, brand\_id, new\_settings):

  """ Update all of the settings for the given brand.

@param new\_settings: a dictionary of at least the settings that have been

changed since the last save that should be updated.

Possible setting keys are: brand\_name, prod\_name\_long, prod\_name\_short,

company\_name, primary\_color, secondary\_color, custom\_css, background\_img\_fd,

logo\_img\_fd

Any value set to None will revert to using the "SparkWeave default" value.

Any value not included will simply not be changed.

The reason for this discrepancy is so that users don't have to re-upload

images every time they set settings in order for the images to not get

delete.

@return: the metadata for the given brand after the update.

"""

1. Test cases for new\_settings keys
   1. The following keys are valid to change the value of:
      1. name
      2. prod\_name\_long
      3. prod\_name\_short
      4. company\_name
      5. primary\_color
      6. secondary\_color
      7. custom\_css
      8. background\_img\_fd
      9. logo\_img\_fd
   2. The following keys in specific should not be settable:
      1. id
      2. active
      3. custom\_css\_path
      4. header\_path
      5. background\_path
      6. css\_hash
      7. mod\_time
      8. create\_time
      9. media\_version
   3. Any other keys given should raise a \_\_\_Exception.
2. Media hash tests
   1. Changing the following keys should NOT change the media hash:
      1. name
      2. prod\_name\_long
      3. prod\_name\_short
      4. company\_name
   2. But changing these keys SHOULD change the media hash:
      1. primary\_color
      2. secondary\_color
      3. custom\_css
      4. background\_img\_fd
      5. logo\_img\_fd
   3. If any of the media-hash-changing keys was included in the new\_settings dictionary, but the value is not actually changing, then the media hash also should not change.
3. All of the tests listed under the "new" function must be retested here.

## Activate

**def** **activate**(conns, brand\_id):

  """ Make the given brand the active brand in favor of all others.

@return: the metadata of the activated brand.

"""

1. NoSuchBrandError is thrown if the given brand ID cannot be found.
2. When the machine is cleared and no brand has been explicitly activated yet, the "SparkWeave default" brand is considered active.
3. If None is given as the brand\_id, then the "SparkWeave default" brand is activated. The previously activated brand is not longer considered active.
4. If an existing brand\_id is given, then than brand becomes active. The previously activated brand is not longer considered active.

## Delete

**def** **delete**(conns, brand\_id):

  """ Deletes the given brand. If this brand was the active brand, the

"SparkWeave default" brand becomes the active brand.

@return: the metadata that the given brand had before it was deleted.

"""

1. DeletionCriteriaViolation is thrown if None ("SparkWeave default") is given as the brand ID.
   1. Afterwards, the default brand is not deleted.
   2. The default brand is not deactivated when previously active.
2. NoSuchBrandError is thrown if the given brand ID cannot be found.
3. list() does not include the deleted brand after a successful delete.
4. If a brand was active when it was deleted, the "SparkWeave default" brand becomes the active one afterwards.