

# Ruby *gravaty* gem

Marco Bresciani

# What is *gravaty*?

- As described in the RubyGems page:
  - This gem automagically prepares complete URIs for Gravatar, for both avatars and profiles, with all currently supported options, included the XML-RPC API, as of 2014-04-30, starting from a single email address!
- Some (hopefully useful) references:
  - <https://rubygems.org/gems/gravaty>
  - <https://savannah.nongnu.org/projects/gravaty>
  - Bugs: <https://savannah.nongnu.org/bugs/?group=gravaty>

# What is Gravatar?

- According to *Gravatar.Com* (retrieved on 2014-05-19):
  - An "avatar" is an image that represents you online—a little picture that appears next to your name when you interact with websites.
  - A Gravatar is a Globally Recognized Avatar. You upload it and create your profile just once, and then when you participate in any Gravatar-enabled site, your Gravatar image will automatically follow you there.
  - Gravatar is a free service for site owners, developers, and users. It is automatically included in every WordPress.com account and is run and supported by Automattic.
- According to *Wikipedia* (updated on 17 April 2014 at 10:22):
  - Gravatar (an abbreviation for globally recognized avatar) is a service for providing globally unique avatars which was created by Tom Preston-Werner. Since 2007, it has been owned by Automattic, who have integrated it into their WordPress blogging platform.

(see <http://en.wikipedia.org/wiki/Gravatar>)

# How to create a *gravaty* object

- Create the `Gravaty` object. The `gravatize` factory method needs your email address as parameter. There is no need to register on Gravatar.Com:
  - `Gravaty::gravatize yourem@il.address`
- Returns your email address MD5 digest:
  - `a_gravaty.digest`
- `to_s` method is also available;
- ***Bonus feature:*** every `Gravaty` is a `Comparable` object, based on `a_gravaty.email` data, your email address in small capital letters.

# The avatar image feature

- With your `Gravaty` object available, the `avatar` method provides the configuration-specific URI for your avatar:
  - `a_gravaty.avatar` returns  
`https://secure.avatar.com/avatar/HASH`  
(where `HASH` is your `a_gravaty.digest` email address MD5 digest)
- The method allows an `Hash`, with avatar configuration parameters:  
`type, pixel_size, force, secure, rating, default`.  
(see <https://en.gravatar.com/site/implement/images/> for details)
- The `avatar!` method saves the provided configuration for your `Gravaty` object's `to_s` output.

# The profile content feature

- With your `Gravaty` object available, the `profile` method provides the configuration-specific URI for your (registered) user:
  - `a_gravaty.profile` returns  
`https://secure.avatar.com/HASH`  
(where `HASH` is your `a_gravaty.digest` email address MD5 digest)
- The method allows an `Hash`, with profile configuration parameters:  
`format, secure`.  
(see <https://en.gravatar.com/site/implement/profiles/> for details)
- The `profile!` method saves the provided configuration for your `Gravaty` object's `to_s` output.

# The XML-RPC API content feature

- With your `Gravaty` object available, the `xmlrpc` method provides a reference to the remote Gravatar XML-RPC AP, for your (registered) user:
  - `a_gravaty.xmlrpc` requires the method name (`grav.test`, by default), your password and possible method parameters (whether needed) via a `Hash`.
- In order to increase security, the password is not saved inside the `Gravaty` object but has to be provided for each call;  
(see <https://en.gravatar.com/site/implement/xmlrpc/> for details)
- Communication happens through SSL: certificate might be required (see `gravaty README.md` file).

# Additional features (1/2)

- Since QRCode and JSON are becoming popular formats, `qr` and `json` methods (together with `qr!` and `json!`) allows to directly retrieve the user's profile data URI, in QRCode or JSON format, with format-specific parameters. Examples:
  - `a_gravaty.qr pixel_size: 42`
  - `a_gravaty.json! callback: 'my_method'`
- The `qr!` and `json!` methods save the provided configuration for your `Gravaty` object's `to_s` output.
- `a_gravaty.qr` corresponds to `a_gravaty.profile` format: `'qr'` while `a_gravaty.json` corresponds to `a_gravaty.profile` format: `'json'` and they both filtered the allowed parameters according to the output type.



## Additional features (2/2)

- When using the 'banged' (!) methods (`avatar!` or `profile!`, same for `qr!` and `json!`), the built URI is saved inside the `Gravaty` object and is read-only available through `to_s`.
- The `download` method, simply specifying a filename as parameter, retrieves whatever resource is indicated by such internal status and saves its content in the specified file.  
Example:
  - `a_gravaty.qr!`
  - `a_gravaty.download 'myQr.png'`
- The `reset` method clears the internal status restoring the email address output of `to_s`.

# Ruby *gravaty* gem

- See **examples** folder for more detailed usage examples.
- This work is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/4.0/>.
- Ruby *gravaty* gem by Marco Bresciani is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.
- Some (hopefully useful) references:
  - <https://rubygems.org/gems/gravaty>
  - <https://savannah.nongnu.org/projects/gravaty>