MindVision Development Standards

Updated July 2012

All MindVision development should meet an appropriate professional standard. This document sets forth a series of minimum criteria to that end, establishing a baseline that allows us to operate a reliable production environment, and confidently support all software therein.

We develop in Ruby on Rails, and store all production data in centrally-managed PostgreSQL and MogileFS servers.

This document explicitly does *not* include specific style guidelines; while we also document such guidelines, they are treated separately, and unlike the issues listed below, are not hard acceptance requirements.

Fundamental / should-go-without-saying considerations

- Meets business requirements
- Consistent with industry best current practices
- Safe from injection attacks, etc. (SQL, HTML, JS, anything else)
- Secure against current OWASP 'Top 10'
- Secure in depth: permissions checks at all levels, not just the UI controls
- Acceptable user-level performance – we can better define this if needed

Code Structure

- Well-factored code
- Skinny controllers; no (or very minimal) logic in views
  Any conditional statement expressions that aren't a method call that ends in '?' or a boolean instance variable are candidates for delegation to the model layer (or a helper, depending on context)
- Avoid repetition in views: use partials and helpers where appropriate
- Helper methods should not:
  - access the model layer like a controller,
  - contain business logic like a model, or
  - contain vast chunks of presentation markup like a view or partial

Libraries / External Code

- Currently supported Rails version, >= 3.0
- Uses established gems instead of reimplementing
- Uses preferred gems etc where appropriate (see list)
- All external dependencies (both libraries and 3rd party services) require approval; pre-approved:
  - MindVision libraries,
  - libraries available under the MIT or (2- or 3-clause) BSD licenses, and
  - Google Analytics

Formatting / Style

- Commented where appropriate – and not where inappropriate
- Comments, UI, and variable names must all be in good English
- Descriptive method and class names
- In-code names consistent with UI/user terminology where-ever possible
- Consistent formatting: must be consistent within a project, ideally conforming to our style guidelines

Specific Considerations

- Appropriately scoped variables: avoid $global, @@class
- Appropriate level of testing: covers custom-written functionality to a level commensurate with its complexity; avoids testing behaviour of the framework or underlying libraries
- Compact asset references (one-to-few JS, one-to-few CSS, sprites where reasonable)
- Considered browser caching behaviour
- Careful use of mass assignment (config.active_record.whitelist_attributes = true)
- Slow tasks performed in background (sidekiq task queue)
- Thread-safe
- Avoid N+1 queries
- GET requests should be safe, and not change server state
- Do not store user passwords
- External code in "vendor" directories
- Database indexes defined as appropriate
- Everything should be encoded as UTF-8
- Shared-nothing architecture

Deployment and External Libraries

The production support environment comprises Linux, Varnish, Nginx, PostgreSQL, Redis, and MogileFS. Applications shall use these services as appropriate, in preference to any alternative. Configuration of these daemons is beyond the scope of the application, though its deployment documentation may, for example, request particular behaviour in the Nginx site config. Redis is suitable for ephemeral data (such as sessions and fragment caches), and unstructured low-priority production data (such as user preferences); all "primary" data must be stored in PostgreSQL, and all binary files must be stored in MogileFS.

A standard Rails deployment will consist of one or more unicorn master processes, which receive proxied requests from Nginx. An application may additionally require other daemon processes to be running, such as a clockwork ticker, or sidekiq workers. Any such requirement should be clearly documented, including whether the process can have multiple instances running (generally, true for anything but a clock process).

An application must not assume all its processes will be running on a single machine, or will have access to the same filesystem; only the three designated data stores (PostgreSQL, Redis, and MogileFS) are known to be accessible to all instances.

Outgoing email should be sent to the local sendmail process, which will queue it, and pass it through the proper channels; no application should make outgoing SMTP connections.

This following are external libraries we have used in the past, and their significance for future development.

There are several levels of library recommendation priority:

suggested	We have used this in the past, but don't care whether it's chosen for future projects.
preferred	We are familiar with this; consider using it if you don't have some other preference.
recommended	We are familiar with this; please use it, or discuss your reason for preferring something else.
required	This is always the right choice, either for consistency between our applications, or in order to work in our production environment. If you feel the need to use something else, please raise the issue immediately.

suggested	
Compass	Sass extensions, sprite management
preferred	
Factory girl	Creating data for tests; fixture replacement
Capybara	Browser testing
RSpec	Unit tests
recommended	
Acts as Archival	“Soft delete”
Cancan	Authorization
required	
Active Admin	Admin interface
MVI Active Admin	Adjustments to Active Admin
CarrierWave[1]	File upload & storage
Redis	Session & cache store
Sidekiq (if needed)	Background job queue
Unicorn	Web server
Airbrake	Exception notification
statsd	Simple site statistics

Separate documentation is available providing brief configuration details for many of the above.

1 Properly configured, CarrierWave will use MogileFS storage in the production environment; provided it is used consistently, it should be sufficient to configure it to use local files in development.