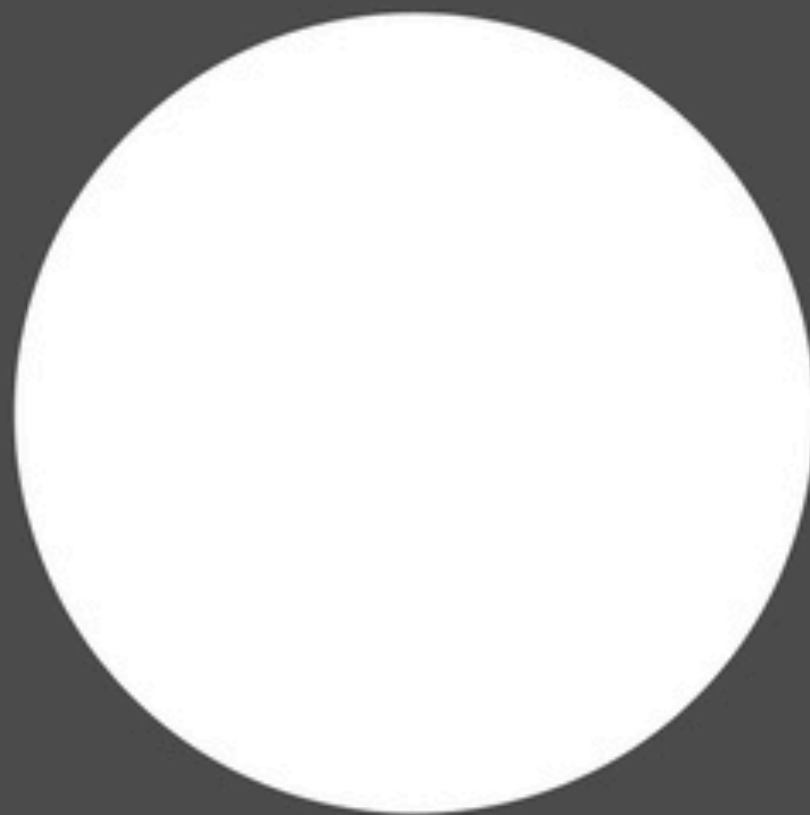






Well hello, 1280 x 720



I am a test circle here to make sure screen
aspect ratios, color, and borders are happy for all.
Nothing to see here. Please move along.



Ungemifying your projects

(one repo to rule them all)

theScore[®]

Sportsball! Also eSports.
(we're hiring!)

let's talk about

MONORAILS

What's a monorail?

A rails app that does *everything*

1. The website
2. API
3. Background jobs
4. Search
5. 3rd party service interaction
6. Kitchen sink duties



Why are monorails bad?

- No clear separation of concerns
- All code is loaded, even if you don't use it
- Scaling concerns (naïve scaling means everything is scaled)
- Test suite runs everything by default

What can we do?

Let's split up app into separate pieces!

Models are usually the only thing needed in each component, so let's just gemify that.

How gemification looks

downstream repo: mycompany-api

downstream repo: mycompany-search

upstream repo: mycompany-models

The code

```
# Gemfile
gem 'mycompany-models',
  git: 'git@github.com:mycompany/mycompany-models.git',
  branch: 'master'

# Gemfile.lock
GIT
remote: git@github.com:mycompany/mycompany-models.git
revision: 976fea8d33e36ce3d63b08375626b6475b1f8ac2
branch: master
specs:
  mycompany_models (1.6.11)
    activerecord (~> 3.2.6)
    activesupport (~> 3.2.6)
```

Workflow

1. Push code to master branch of mycompany-models
2. Update downstream projects:

```
# in terminal
```

```
bundle update mycompany-models
```

Downsides of gemifying

- Code changes between gemified projects and downstream projects are tied together
- Multiple pull requests for the same issue!
- Order of merging pull requests is important
- Can forget about some pull requests

Model

scoremedia / [redacted] PRIVATE

Add Soccer::PlayerRecord.played scope #515

Merged Nitrodist merged 1 commit into master from CORE-1016-played-scope-for-soccer-player-records on Jun 6, 2014

Conversation 4Commits 1Files changed 2

Nitrodist commented on Jun 5, 2014

CORE-1016 [redacted]

This commit introduces a new scope called .played that will list any players that had any minutes during the game (indicating that they played during the game).

Very similar to Soccer::GoalieRecord.played (64f4b22).

Add Soccer::PlayerRecord.played scope ... d83f6b5

Downstream Project

```
commit 4f4e7d98aec55689f91ca503446efbceea39e7d
Author: Mark Campbell <nitrodist@gmail.com>
Date: Fri Jun 6 10:52:12 2014 -0400

Bump [redacted]
```

scoremedia / [redacted] PRIVATE

Add the 'clean_sheet' alert for soccer goalies #212

Merged Nitrodist merged 1 commit into master from CORE-1016-clean-sheet-alert on Jun 2, 2014

Conversation 6Commits 1Files changed 11

Nitrodist commented on May 23, 2014

CORE-1016 [redacted]

The logic for generating a clean sheet alert for a goalie is based off the fact that:

1. The team finished with a score of 0 (status of game is final)

2. The goalie was the only goalie with minutes played on their team

The Solution

One repo
Multiple projects

Directory structure

```
$ ls -al mycompany
mycompany-api/
mycompany-models/
mycompany-search/
circle.yml
README.md
specs.sh # run all the tests!
.gitignore
.hound.yml
```

The better code

```
# Gemfile in mycompany-api
gem 'mycompany-models', path: '../mycompany-models'

# Gemfile.lock in mycompany-api
PATH
  remote: ../mycompany-models
  specs:
    mycompany_models (1.6.11)
      activerecord (~> 4.2.0)
      activesupport (~> 4.2.0)
```


Benefits

- One pull request per change
- No 'bumping' the model gem
- Projects never accidentally use 'old' version of models
- New developers don't WTF
- Save GitHub \$\$\$!

How to

Three ways:

1. Don't gemify in the first place
2. New git repo with code copied directly in
3. New git repo with fancy history preserved (do this!)

Preserve history

Don't break the SHA1 hashes!

Modifying history means that the SHA1 hash of the commit changes (rebase, commit --amend, etc.) -- let's avoid it!

High level steps

1. Make new repo
2. Add projects as remotes
3. Checkout each project locally
4. Move files into top-level directory called \$project
5. Merge in projects one-by-one
6. Modify each Gemfile to use relative paths

Code

`http://bit.ly/1IGxkR8`

[link](http://bit.ly/1IGxkR8)

Going further

Why stop at gems? Works just as well for SOA/Microservices!

Questions?

Credit goes to

- Luke Reeves
- Scott Walkinshaw
- Thuva Tharma
- Nate Smith (actually wrote the shell script!)

Thanks!

@Nitrodist on Twitter