

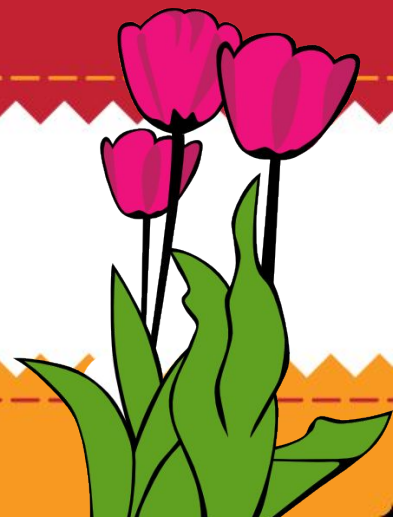
# Политика невмешательства

Как в tulpr.ru решена задача  
сохранения изменений  
состояния системы



The logo for DevConf, featuring the word "Dev" in orange and "Conf" in red. The letter "v" is stylized with a hand icon, and the letter "o" is also stylized with a hand icon.

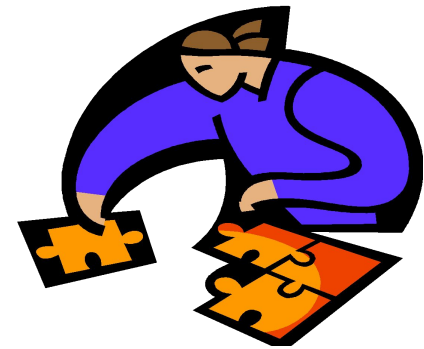
<http://www.devconf.ru>



# Постановка задачи

- Состояние системы

- Модель А
- + Модель В
- + Модель С



- Требуется просматривать

- состояние "до" изменения
- состояние "после" изменения
- служебную информацию (кто, когда, ...)

# Варианты решения


- Inline code
- Callback
  - `before_save`, `after_save`
- Observer
  - `paper_trail`, `act_as_versioned`
- Decorator
  - `method_decorators`
- Aspect
  - `aspectr`, `aquarium`



# Inline code




```
save_before( a, b, c );  
change( a, b, c );  
save_after( a, b, c );
```

- Загрязняется основная логика
- Необходимо поддерживать изменения
-  (не катит)

# Callback

```
before_save    :save_state_before  
after_save     :save_state_after
```

- Сохранение состояния 1 модели (не группы)
- Дополнительный код в моделях
- 

# AR::Observer:

## обычная реализация

```
class UserObserver < ActiveRecord::Observer
  def after_save(user)
    ...
  end
end
```


- Сохранение состояния 1 модели (не группы)

- 

# AR::Observer:

## реализация через gem

```
class User < ActiveRecord::Base
  # команда подключения функционала гема
  has_paper_trail
  # acts_as_versioned
end
```

- **Сохранение состояния 1 модели (не группы)**
- ~~Дополнительный код в моделях~~
- 

# Decorator

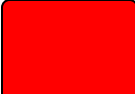


```
class UserService < MethodDecorators
```

```
gem install method_decorator
```

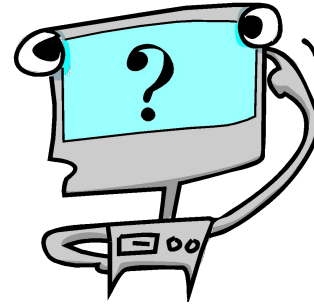
## +Log\_user\_state

```
  def approve(user_id)
    User.find(user_id).approve()
    Billing.add_user(user_id)
    Complaint.scope_user(user_id).each(&:close)
    save!
  end
end
```

- Нужно много различных декораторов
-  где есть следы (хоть и приемлемо мало)

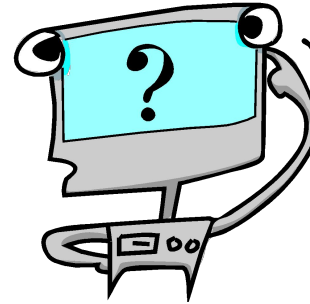


# Aspect



**КТО ЗНАЕТ,  
ЧТО ТАКОЕ АОП?**

# Aspect



- AOP
- Aspect
- Advice
- Join Point

```

    Программа
    ...
    ...
    user.set_role("admin")
    ...
    ...
  
```

```

    Aspect

    class UserAspect < AOP
    advice User, :set_role, options
    end
  
```

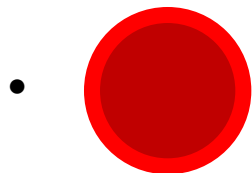
# Aspect



```
class Changelog
```

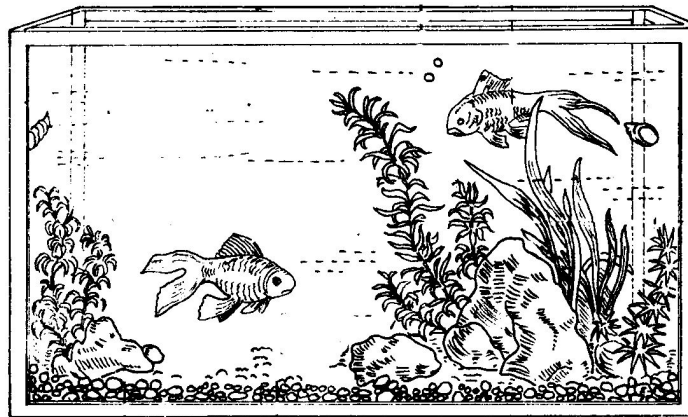
```
watch UserService, [:approve, :disapprove], {  
  state_before: lambda{ |user_id| ... },  
  state_after:  lambda{ ... },  
  state_error:  lambda{ ... }  
}  
end
```

- Нет ни единой строчки в моделях/сервисах/etc
- Все настройки – в одном месте

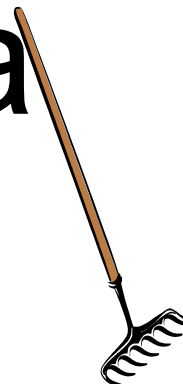


# Выбор

- AspectR (2002)
- Aquarium - gem install aquarium



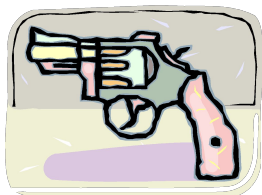
## Практика



- Aquarium

- мощный

- слишком



- Упрощение интерфейса

- более жёсткие требования

- обработка ошибок

- спокойный сон после release



A light blue curved arrow pointing from the top left towards the center text.

# Откат

в жизни модератора



- Детальная трассировка изменений
- У меня все ходы записаны (с) 12ст
- Легко откатить изменения
- Долой коррупцию, распил и откаты!

# ССЫЛКИ

- <http://bit.ly/cR400e> - AOP @ wikipedia
- <http://bit.ly/15q8bT> - Decorator @ wikipedia
- <http://bit.ly/It8d7l> - method\_decorators
- <http://bit.ly/KXu9ZA> - aquarium @ github
- <http://bit.ly/rUyqf> - aquarium @ rubyforge



**ВСЕМ**

**Спасибо за внимание**

