Обеспечение безопасности веб-приложения


bomz.org
ЗНАНИЕ - СИЛА
незнание - рабочая сила
bomz.org

Знания

- Актуальность
- Аутентификация и авторизация
- Сессии
- Ручная аутентификация
- Аутентификация с помощью gem devise
- Каждый сверчок — знай свой шесток (gem pundit)
- Некоторые наиболее распространённые уязвимости

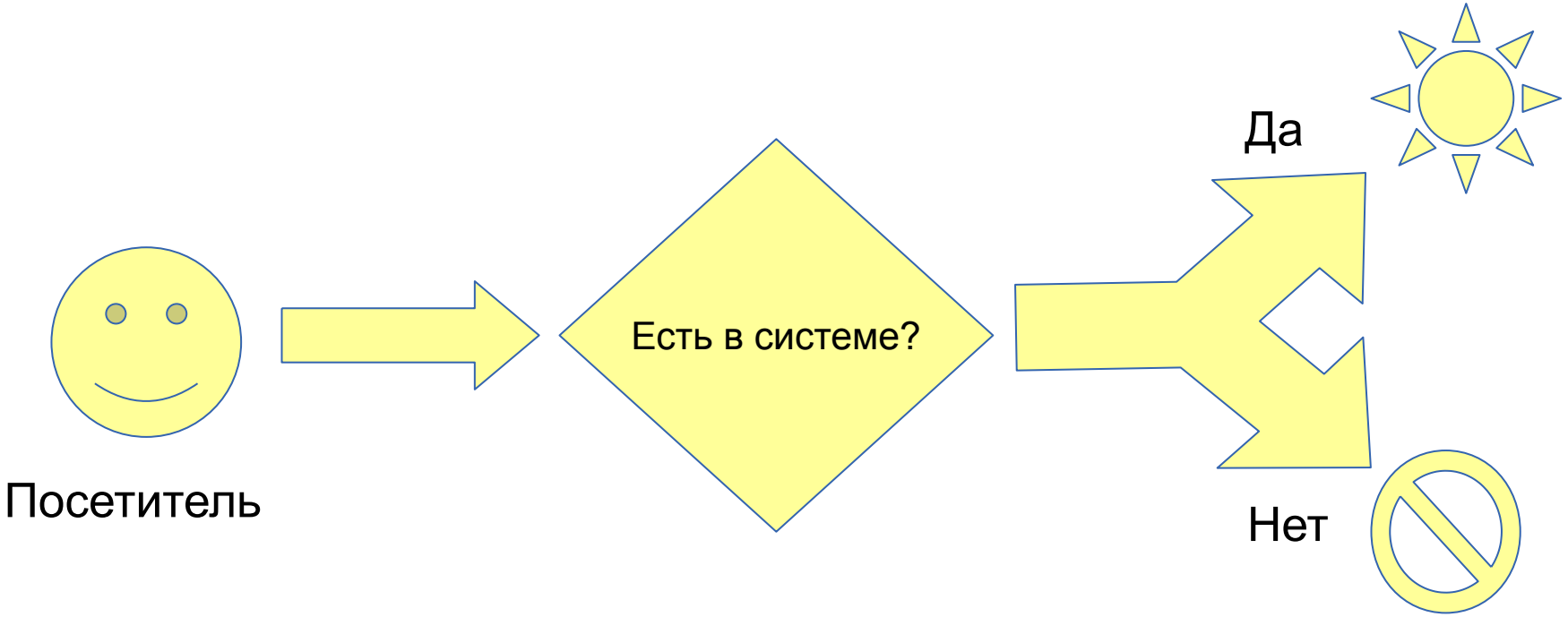
Что такое обеспечение безопасности?

Что такое обеспечение безопасности?

- Процесс устранения последствий различных «опасностей» и профилактика их причин

Почему обеспечение безопасности актуально?

Аутентификация

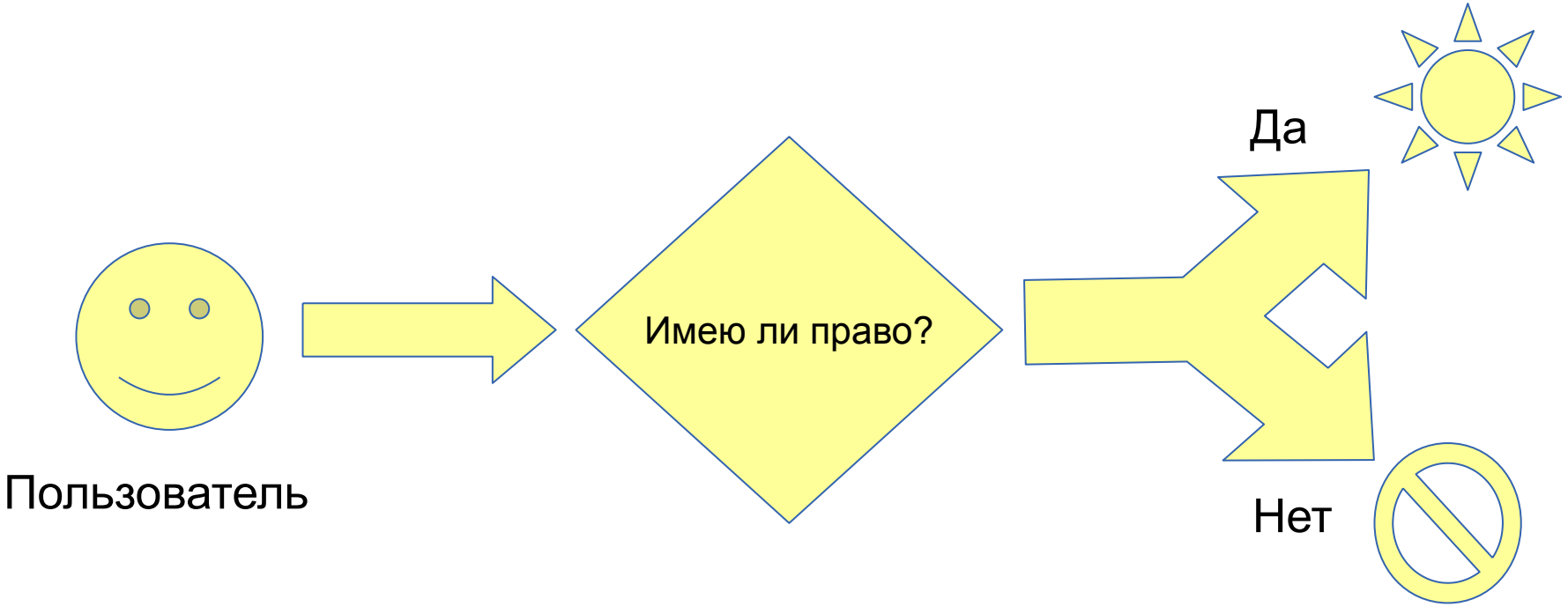


Виды аутентификации

- Логин/пароль
- Токен (для REST)
- OAuth 2.0 (соц. сети)

Авторизация

Авторизация



Сессия

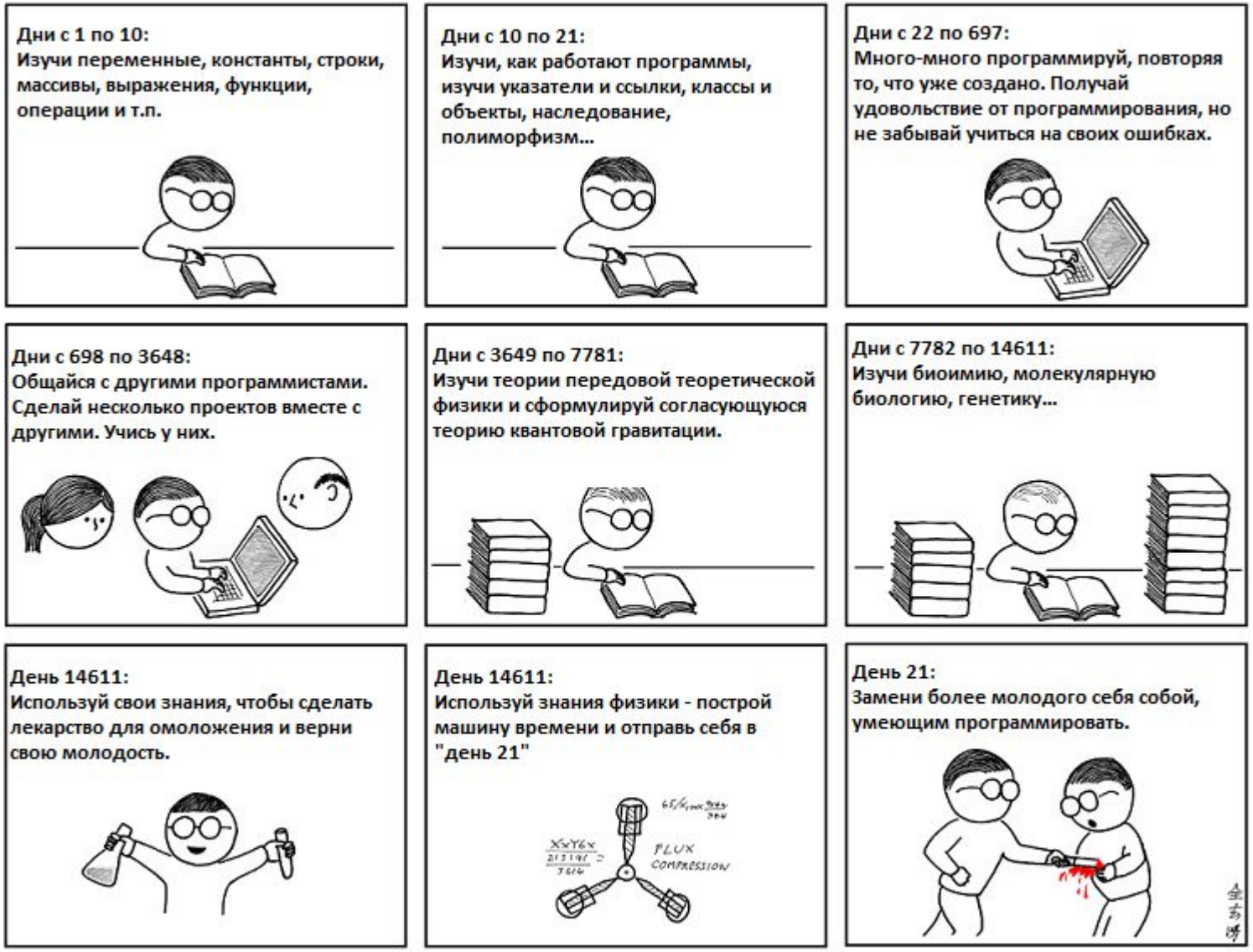
Обеспечение постоянства (обычно ActionDispatch::Session::CookieStore)

- Обычный хеш:
  - session[:user_id] = user.id
  - session[:user_id] = nil
- Храниться (Rails.application.config.session_store):
  - В файлах на сервере
  - На клиенте
  - В БД
  - Протокол DRB
  - Memcached / Redis

Ручная аутентификация

- Используется фильтр в контроллере:
  - before_action :authorize
- Проверяется соответствие логина и пароля данным из БД
- Пароль шифрован и проверяется с помощью метода authenticate из gem Bcrypt (заменить password на password_digest)
- Вход/выход — через собственный контроллер SessionController
- В случае успеха запоминаем результат в сессии
- Создаём для удобства ряд вспомогательных методов (current_user, sign_in, sign_out)

Сколько нужно времени, чтобы изучить язык программирования?



Насколько я знаю - это самый простой способ "*Изучить C++ за 21 день*"

Аутентификация с devise

- Различные виды аутентификации
- Используется ряд модулей — для запоминания входа, подтверждения аккаунта и т. д.
- Дополняется модель User (убрать password_digest)
- Добавляются ряд хелперов:
  - user_signed_in?
  - current_user
  - authenticate_user!
  - ...
- Ряд готовых к использованию представлений

Pundit:
каждый сверчок —
знай свой шесток

Авторизация с pundit

- Контроль ролей — в контроллерах и представлениях
- Ролевую модель мы создаём сами
- Используются политики (обычный Ruby-класс)

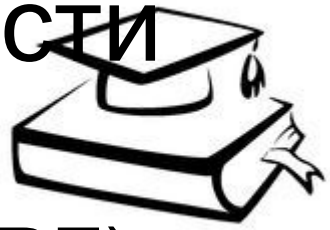
Соглашения в политиках

Название класса — это название модели + постфикс Policy.
Первый аргумент контроллера — пользователь (обычно текущий), второй — объект, который мы хотим проверить.
Обычно есть несколько методов-запросов (т. е. с ? в конце), названия которых соответствуют названиям действий в контроллере.
Объект модели называется record.

Пример политики

```
class PostPolicy
  attr_reader :user, :post

  def initialize(user, post)
    @user = user
    @post = post
  end

  def update?
    user.admin? or not post.published?
  end
end
```

Некоторые наиболее распространённые уязвимости



- Подделка межсайтовых запросов (CSRF) — ссылка на страницу веб-приложения, где пользователь аутентифицирован
- SQL-инъекции — внедрение SQL-запроса в параметры запроса
- Межсайтовый скриптинг (XSS) — внедрение вредоносного кода на стороне клиента


Ты молод, креативен, талантлив? Амбициозен, уверен в себе, полон свежих идей? А делать хоть что-нибудь умеешь?!
Atkritka.com

Умения

- Реализовать ручную аутентификацию пользователя
- Реализовать аутентификацию с помощью gem devise
- Разделить пользователей по ролям с помощью pundit
- Обеспечить прохождение тестов после внедрения аутентификации

- Реализовать аутентификацию с помощью gem devise

- $ rails g devise:install
- $ rails g devise User
- $ rails g devise:views:bootstrap_templates (gem 'devise-bootstrap-views')
- def after_sign_(in|out)_path_for(resource)
- before_action :authenticate_user!

- Реализовать авторизацию с помощью gem pundit

- $ rails g pundit:install
- authorize **Competence**, **:create?**
- **<% if policy(*Competence*).create? %>**

• Обеспечить прохождение функциональных тестов

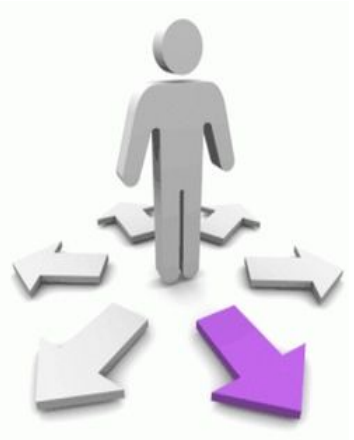
```ruby
# test_helper.rb
# role — название фабрики пользователя с соответствующей ролью
# пользователь с такими email/password уже должен быть создан
# например, sign_in(:cosmonaut)
def sign_in(role)
  post user_session_path, params: { user: attributes_for(role) }
end

# пример фабрики
factory :user do
  password { '123456' }

  factory :cosmonaut do
    role { User.roles[:cosmonaut] }
    email { "cosmonaut@mail.bro" }
  end

end
```

Обеспечить прохождение приёмочных тестов

```ruby
# test_helper.rb
# role — название фабрики пользователя с соответствующей ролью
# например, sign_in(:cosmonaut)
def sign_in(role)
  user = create(role)

  visit new_session_url

  fill_in 'Email', with: user.email
  fill_in 'Password', with: user.password

  click_on 'Sign in'
end
```

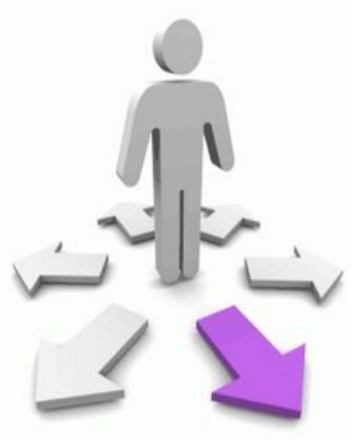

ТУДА
100м
СЮДА
200м
ВЕЧНАЯ НЕОПРЕДЕЛЕННОСТЬ!
КАК ЖИТЬ?
DEMOTIVATORS.RU

Неопределённости



- Разница между аутентификацией и авторизацией?
- Есть ли альтернативы devise?
- Есть ли альтернативы pundit? Да, cancancan.
- Маршруты пользователя нужно прописать после devise_for

Альтернативы devise



- github.com/NoamB/sorcery — от EvilMartians
- authlogic
- railscasts.com/episodes/270-authentication-in-rails-3-1

Результат


Это же так просто!
Я это сделаю!
Я могу это сделать!
Я попытаюсь...
Как мне это сделать?
Я хочу это сделать!
Я не могу это сделать (не умею, не получится)
Я не буду этого делать.

Результат

- Изучена ручная аутентификация
- Изучена аутентификация с помощью devise
- Изучена авторизация с помощью pundit
- Прокачали различные аспекты обеспечения безопасности веб-приложения на Rails

Самостоятельно

- Использование bcrypt для шифрования
- Граватары
- Strong parameters
- sanitizing
- …

Граватар

- Глобально распознаваемый аватар
- Можно использовать на разных сайтах (брендинг)
- Предоставляется изображение по умолчанию
- Можно зарегистрироваться и сохранить необходимый (info@profport.org)

Алгоритм получения

- Перевести email в нижний регистр
- Получить id с помощью Digest::MD5::hexdigest
- Использовать id для получения ссылки на граватар:
  - "https://secure.gravatar.com/avatar/#{id}"
- Использовать в image_tag (или где надо)

Список источников

- Основное
- Сессии и флеш-сообщения
- Использование bcryptдля аутентификации
- Gem Devise
- Gem pundit
- Дополнительное
- Аутентификация по токену
- Аспекты безопасности Rails