Именно поэтому вызов

```
User.find(params[:id])
```

в листинге 7.5 обнаруживает пользователя с идентификатором 1.

7.1.3. Отладчик

В разделе 7.1.2 мы видели, как отладочная информация помогает понять, что происходит в приложении. Начиная с Rails 4.2 появился еще более непосредственный способ получения отладочной информации с помощью гема byebug (листинг 3.2). Чтобы увидеть, как он работает, просто добавьте строку debugger в приложение, как показано в листинге 7.6.

Листинг 7.6 ❖ Контроллер Users с отладчиком (app/controllers/users_controller.rb)

```
class UsersController < ApplicationController

  def show
    @user = User.find(params[:id])
    debugger
  end

  def new
  end
end
```

Если теперь посетить адрес /users/1, сервер Rails вернет приглашение byebug:

```
(byebug)
```

Его можно рассматривать как аналог консоли Rails и выполнять команды, чтобы узнать состояние приложения:

```
(byebug) @user.name
"Example User"
(byebug) @user.email
"example@railstutorial.org"
(byebug) params[:id]
"1"
```

Чтобы выйти и продолжить выполнение приложения, нажмите **Ctrl-D**, затем удалите строку debugger из метода show (листинг 7.7).

Листинг 7.7 ❖ Контроллер Users с удаленной строкой отладчика (app/controllers/users_controller.rb)

```
class UsersController < ApplicationController

  def show
    @user = User.find(params[:id])
  end

  def new
  end
end
```

Всякий раз, когда появляются какие-то сомнения, добавьте вызов `debugger` близко к той части кода, которая, по вашему мнению, вызывает проблемы. Исследование состояния системы с помощью `byebug` – весьма мощный метод выявления ошибок и интерактивной отладки приложения.

7.1.4. Аватар и боковая панель

Определив в предыдущем разделе заготовку страницы пользователя, теперь немного улучшим ее, добавив изображение пользователя и начальную реализацию боковой панели. Начнем с добавления «глобально распознаваемого аватара», или *граватара* (http://gravatar.com), к профилю пользователя[1]. Gravatar – это бесплатная служба, позволяющая пользователям загружать изображения и связывать их со своими адресами электронной почты. То есть это удобный способ добавить изображение пользователя, не связываясь с проблемами выгрузки изображений, их обрезкой и хранением; все, что нам нужно, – это создать правильный адрес URL изображения в службе Gravatar, используя адрес электронной почты, и соответствующий аватар появится автоматически. (Как реализовать выгрузку собственных изображений, рассказывается в разделе 11.4.)

Нам нужно определить вспомогательную функцию `gravatar_for`, которая будет возвращать аватар для данного пользователя, как показано в листинге 7.8.

Листинг 7.8 ❖ Представление, отображающее имя пользователя с и его аватар (app/views/users/show.html.erb)

```
<% provide(:title, @user.name) %>
<h1>
  <%= gravatar_for @user %>
  <%= @user.name %>
</h1>
```

По умолчанию методы, определенные в любом вспомогательном файле, автоматически доступны в любом представлении, но для удобства мы поместим `gravatar_for` в файл для вспомогательных функций, связанных с контроллером Users. Как отмечено в документации к Gravatar (http://en.gravatar.com/site/implement/hash/), адреса URL для доступа к аватарам основаны на MD5-хэше (https://ru.wikipedia.org/wiki/MD5) адреса электронной почты пользователя. В Ruby алгоритм MD5-хэширования реализуется с помощью метода `hexdigest`, который является частью библиотеки Digest:

```
>> email = "MHARTL@example.COM".
>> Digest::MD5::hexdigest(email.downcase)
=> "1fda4469bcbec3badf5418269ffc5968"
```

Поскольку адреса электронной почты нечувствительны к регистру (раздел 6.2.4), в отличие от MD5-хэшей, мы использовали метод `downcase`, чтобы

---

[1] В индуизме аватаром называют проявление божества в человеческом или животном образе. В более широком смысле термин аватар используется для обозначения некоего представления личности, особенно в виртуальной сфере.