

Flexstore on Rails Tutorial

Автор [Christophe Coenraets](#)
Перевод [Андрей Горбатов](#)

[Оригинал статьи](#)

Обзор

Flexstore – традиционный интернет-магазин. В этом tutorialе мы создадим два модуля:

1. Административный модуль – внутреннее приложение, используемое для поддержки базы данных продукции. Он используется для создания, удаления и обновления продуктовых единиц.
2. Модуль магазина как его видит покупатель. Покупатели используют его для просмотра и фильтрации продуктов в каталоге.

В административном модуле, мы используем функцию Rails – скаффолдинг – которая автоматически обеспечивает процессы создания, просмотра, редактирования и удаления продуктов. В модуле магазина мы экспериментируем с дополнительными особенностями:

- Темплейты
- Фильтрация с помощью Ajax
- Частичные темплейты страниц
- Билдер-темплейты
- Установка Flex приложения в качестве фронт-энда приложения Ruby on Rails

В tutorialе используется база данных MySQL. Предполагается, что вы уже установили и запустили MySQL.

Создание базы данных Flexstore

1. Создайте базу данных и назовите ее "flexstore"

Откройте командную строку, перейдите в каталог bin папки вашего сервера MySQL и выполните следующую команду:

```
mysqladmin -uroot create flexstore
```

Не забудьте ввести имя пользователя (-u) и пароль (-p) если необходимо.

2. Импортируем данные
 - Загрузите flexstore.sql.zip [здесь](#)
 - Извлеките flexstore.sql в каталог bin папки сервера MySQL Server
 - Выполните следующую команду для импорта базы данных:

```
mysql -uroot flexstore < flexstore.sql
```

Установка Ruby and Rails

1. Устанавливаем Ruby
 - Загрузите инсталлятор Ruby [здесь](#)
 - Запустите инсталлятор. Принимайте все настройки по умолчанию.
2. Установка Rails

Выполните следующую команду в каталоге c:\ruby:

```
gem install rails --remote --include-dependencies
```

Создание административного модуля

1. Создание приложения flexstore

- Создайте папку rails в c:\
- Выполните следующую команду c:\rails:

```
rails flexstore
```

2. Конфигурация базы данных для приложения

- Редактируем database.yml в c:\rails\flexstore\config
- Установите в качестве database **flexstore** в development, test, и production разделах

3. Создание контроллера для административного модуля

Выполните следующую команду в c:\rails\flexstore

```
ruby script\generate controller Admin
```

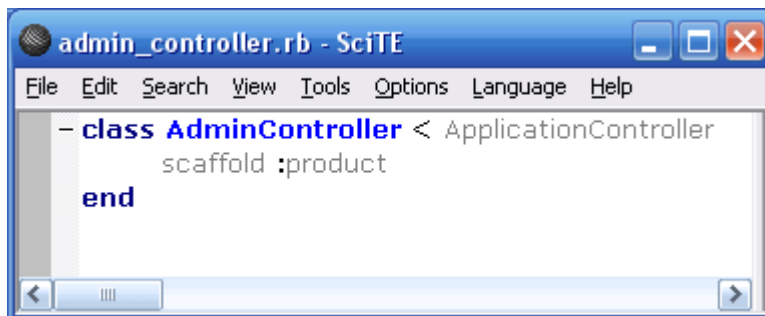
4. Создание модели для продуктов

Выполните следующую команду в c:\rails\flexstore:

```
ruby script\generate model Product
```

5. Изменение контроллера admin для включения скаффолдинга

- Откройте файл admin_controller.rb в c:\rails\flexstore\app\controllers
- Измените класс следующим образом:



6. Тестирование приложения

- Запустим сервер WEBrick, установленный с Rails

Выполните следующую команду в c:\rails\flexstore:

```
ruby script\server
```

- Откройте в браузере следующий URL:

<http://localhost:3000/admin>

Скаффолдинг Rails, определенный в контроллере Admin, автоматически создает действия и виды по умолчанию для просмотра, создания, редактирования и удаления продуктов. Каждое из этих действий или видов

может быть переопределено. Следующим шагом мы переопределим действие index.

7. Определение собственного действия index

- Откройте файл admin_controller.rb в c:\rails\flexstore\app\controllers
- Определите действие index следующим образом:



8. Создание вида для действия index

- Создайте файл index.rhtml в c:\rails\flexstore\app\views\admin
- Введите следующий код в index.rhtml:

```
<html>
<head>
<title>Product List</title>
</head>
<body>

<table>
<tr>
<td>Name</td>
<td>Price</td>
</tr>

<% @products.each do |product| %>
<tr>
<td><%= link_to product.name, :action => "show", :id => product.id %></td>
<td align="right"><%= sprintf("$%0.2f", product.price) %></td>
</tr>
<% end %>
</table>
<p><%= link_to "Create new product", :action => "new" %></p>

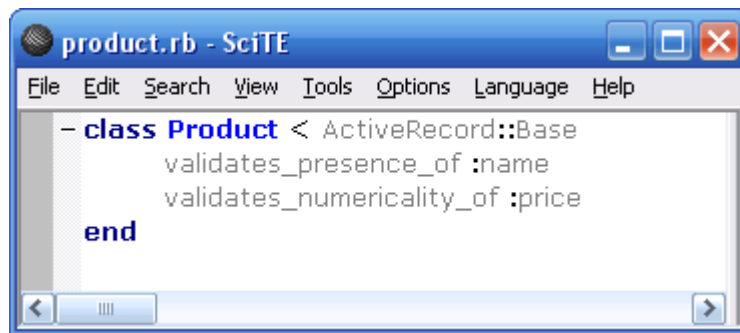
</body>
</html>
```

9. Тестирование приложения. Откройте в браузере следующий URL:

<http://localhost:3000/admin>

10. Валидация

- Откройте файл product.rb в c:\rails\flexstore\app\models
- Измените класс следующим образом:



```
product.rb - SciTE
File Edit Search View Tools Options Language Help
- class Product < ActiveRecord::Base
  validates_presence_of :name
  validates_numericality_of :price
end
```

11. Протестируйте приложение еще раз. Попробуйте добавить продукт без имени или введя стоимость не в числах.

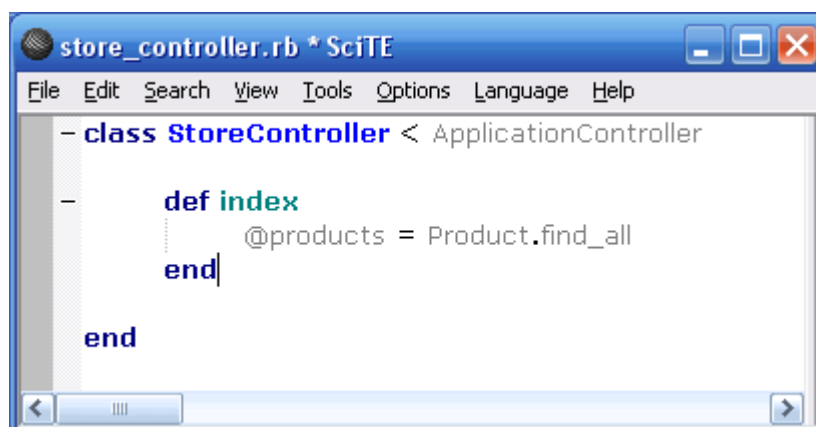
Создание модуля магазина

1. Размещение изображений продукции и таблицу стилей
 - Загрузите assets.zip [здесь](#).
 - Разархивируйте assets.zip в c:\rails\flexstore\public.
2. Создание контроллера для модуля магазина

Выполните следующую команду в c:\rails\flexstore:

```
ruby script\generate controller Store
```

3. Определение действия для контроллера Store
 - Откройте файл store_controller.rb в c:\rails\flexstore\app\controllers
 - Определите действие index следующим образом:



```
store_controller.rb * SciTE
File Edit Search View Tools Options Language Help
- class StoreController < ApplicationController
-   def index
     @products = Product.find_all
   end
end
```

4. Создание вида для действия index
 - Создайте файл index.rhtml в c:\rails\flexstore\app\views\store
 - Введите следующий код в index.rhtml:

```
<html>
<head>
<title>Flexstore on Rails</title>
<%= stylesheet_link_tag "flexstore", :media => "all" %>
</head>

<body>
```

```

<!-- begin catalog -->
<div id="catalog">
<% for product in @products %>

  <!-- begin thumbnail -->
  <div class="thumbnail">
  <strong><%= product.name %></strong>
  
  <div>
  <font color="#CC6600"><b><%= sprintf("$%0.2f", product.price) %></b></font>
  <p>
  <%= product.camera==1?'Camera<br />':" %>
  <%= product.video==1?'Video<br />':" %>
  <%= product.triband==1?'Triband':" %>
  </p>
  </div>
  </div>
  </div>
  <!-- end thumbnail -->

  <% end %>
</div>
<!-- end catalog -->

</body>
</html>

```

12. Тестирование приложения. Откройте в браузере следующий URL:

<http://localhost:3000/store>

Использование частичного темплейта страницы

В реальной жизни вы вероятно захотите отображать превьюшки продуктов в различных частях приложения. Например, нам это необходимо при создании модуля фильтрации, описанного в следующем разделе. Чтобы избежать повтор кода, мы сохраним часть кода, отображающего превьюшки в частичном темплейте страницы.

1. Создайте файл `_product.rhtml` (частичный темплейт страницы) в `c:\rails\flexstore\app\views\store`
2. Откройте `index.rhtml` в `c:\rails\flexstore\app\views\store`
3. Скопируйте фрагмент HTML, относящийся к диву `thumbnail` и вставьте его в `_product.rhtml`
4. В `index.rhtml`:
 - Удалите содержимое дива `catalog` (включая цикл и див `thumbnail`)
 - Добавьте следующий код в див `catalog`:

```
<%= render(:partial => "product", :collection => @products) %>
```

Так как коллекция (список продуктов) у нас является параметром, то частичный темплейт страницы будет повторяться для каждого элемента коллекции.

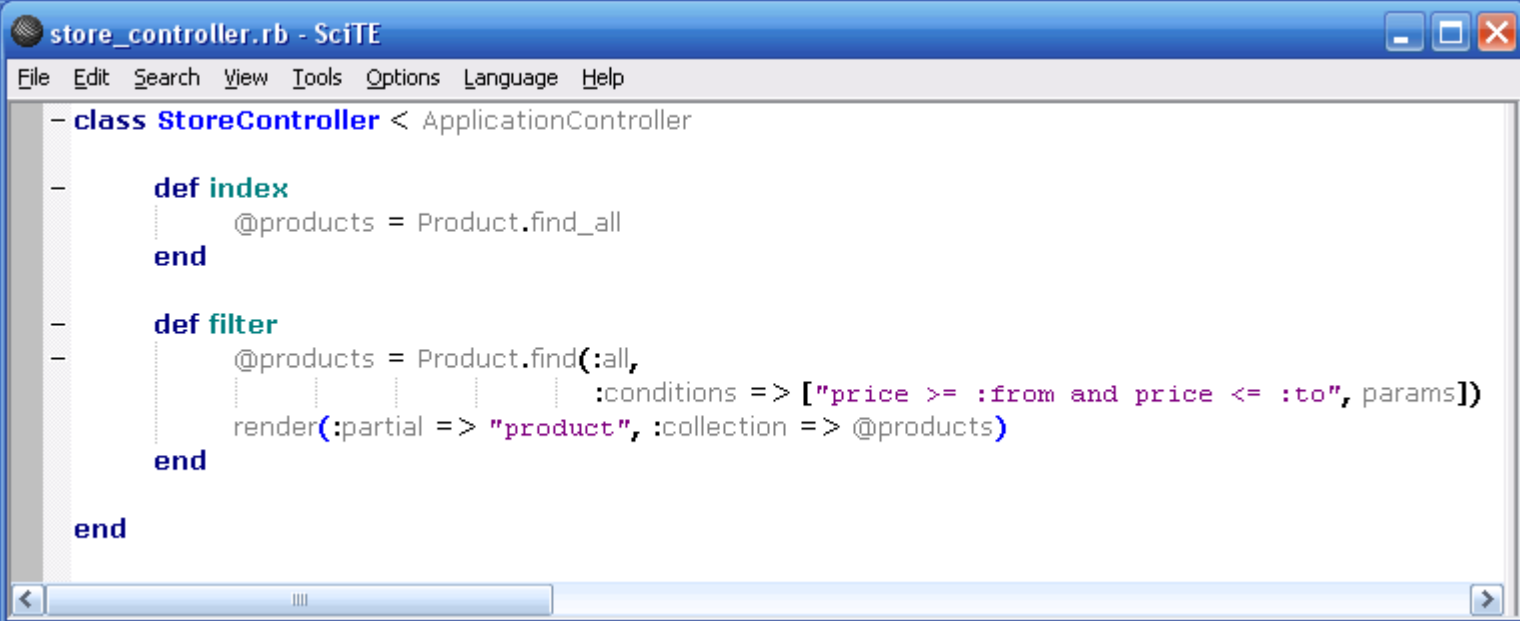
5. Тестирование приложения. Каталог продукции должен отображаться также как и в предыдущем разделе:

<http://localhost:3000/store>

Фильтрация с помощью Ajax

В этом разделе мы добавим возможность фильтрации продуктов по цене. В ответ на выбор пользователя, каталог продуктов будет обновляться и отображать телефоны только выбранной ценовой категории. Каталог будет обновляться без обновления страницы в целом. Это будет достигнуто с использованием встроенной поддержки Ajax.

3. Добавление действия filter в контроллер store
 1. Откройте файл store_controller.rb в c:\rails\flexstore\app\controllers
 2. Определим действие filter следующим образом:



```
store_controller.rb - SciTE
File Edit Search View Tools Options Language Help
- class StoreController < ApplicationController
-
-   def index
-     @products = Product.find_all
-   end
-
-   def filter
-     @products = Product.find(:all,
-                             :conditions => ["price >= :from and price <= :to", params])
-     render(:partial => "product", :collection => @products)
-   end
-
- end
```

Обратите внимание, что мы используем частичный темплейт страницы для отображения списка продуктов выбранной ценовой категории..

4. Изменение вида index для добавления возможности ввода цены
 1. Откроем файл index.rhtml в c:\rails\flexstore\app\views\store
 2. Добавляем тег включения JavaScript-файла сразу после тега со ссылкой на таблицу стилей.

```
<%= javascript_include_tag "prototype" %>
```

3. Добавьте следующий html-код сразу после дива catalog.

```
<div id="left">
<%= form_remote_tag(:update => "right", :url => {:action => :filter}, :loading =>
"$('right').innerHTML=""") %>
Select your price range:<br />
<br />
From:<br />
<%= text_field_tag("from", "0") %>
<br />
<br />
To:<br />
<%= text_field_tag("to", "1000") %><br />

<br />
```

```
<%= submit_tag "Filter" %>
```

```
<%= end_form_tag %>
```

```
</div>
```

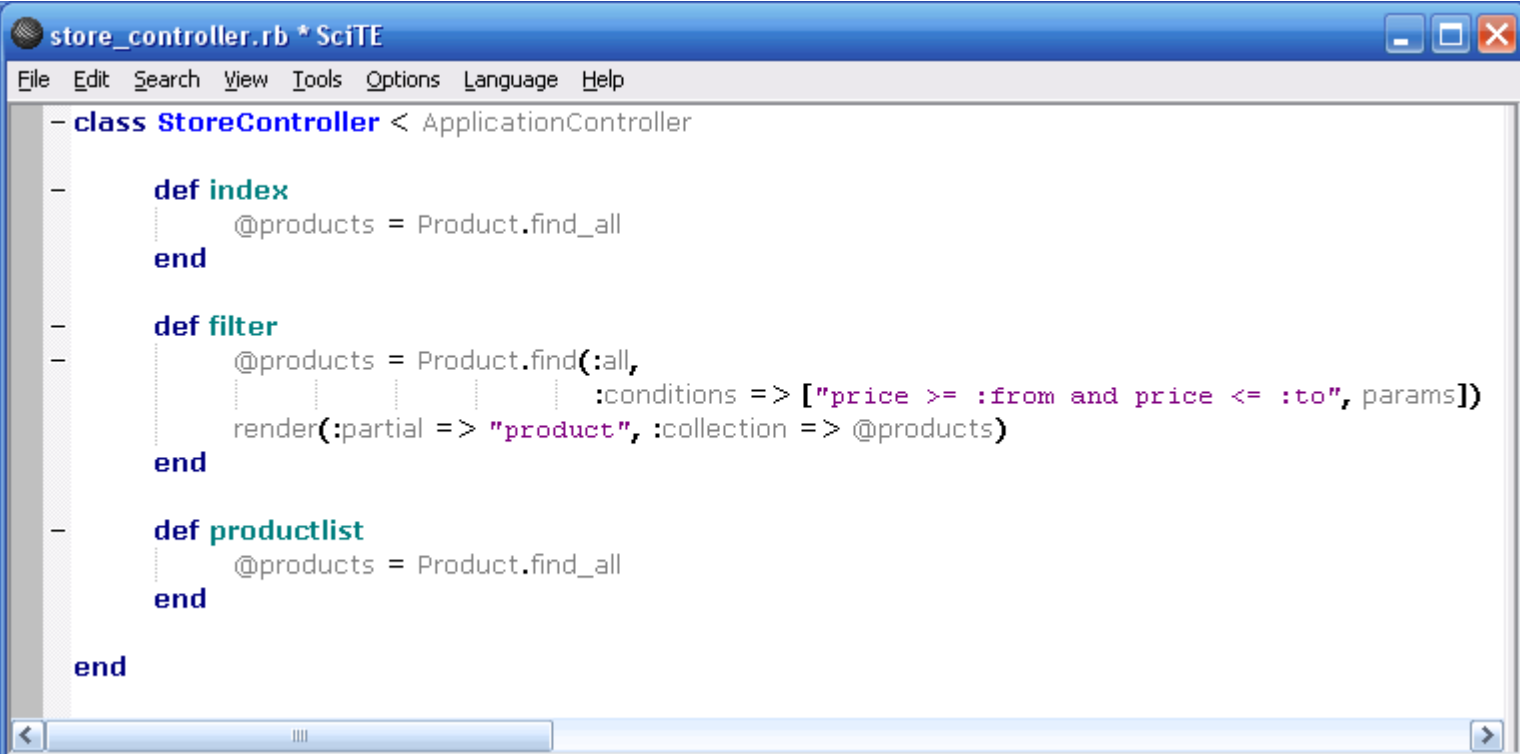
5. Измените id дива catalog на "right". Это спозиционирует каталог справа от панели фильтрации.
6. Тестируем приложение

<http://localhost:3000/store>

Билдер-темплейты (Builder templates)

С помощью билдер-темплейтов вы можете динамически генерировать XML документы. Это обеспечивает точку объединения Rails с другими технологиями и увеличивает производительность фреймворка. В этом разделе мы создадим темплейт, генерирующий XML документ для каталога продуктов.

6. Добавление действия productlist в контроллер store
 - Откройте файл store_controller.rb в c:\rails\flexstore\app\controllers
 - Определите действие productlist следующим образом:



```
store_controller.rb * SciTE
File Edit Search View Tools Options Language Help
- class StoreController < ApplicationController
-   def index
-     @products = Product.find_all
-   end
-   def filter
-     @products = Product.find(:all,
-                             :conditions => ["price >= :from and price <= :to", params])
-     render(:partial => "product", :collection => @products)
-   end
-   def productlist
-     @products = Product.find_all
-   end
- end
```

7. Создание билдер-темплейта
 - Создайте файл productlist.rxml в c:\rails\flexstore\app\views\store
 - Отредактируйте файл productlist.rxml следующим образом:

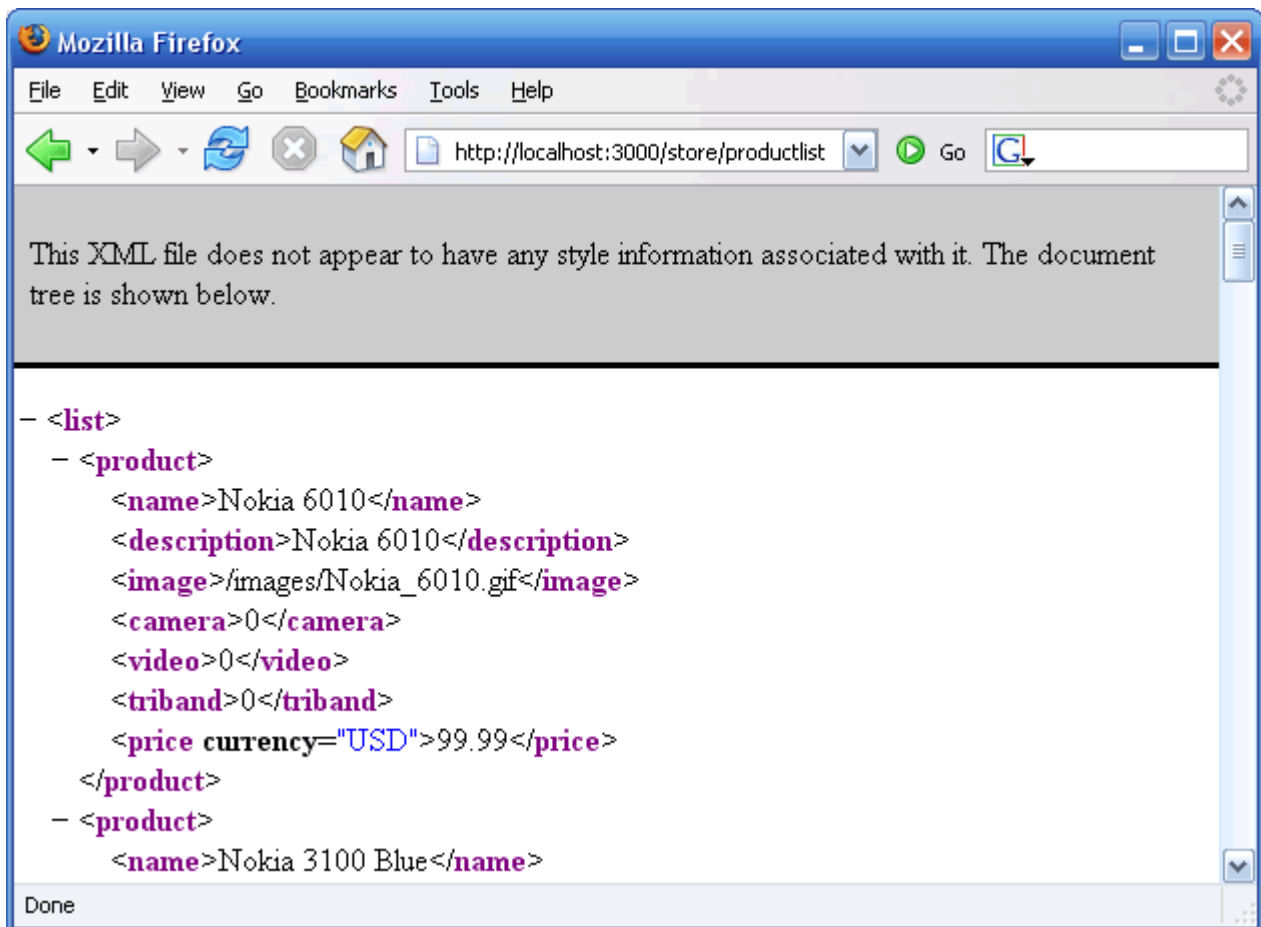
```
xml.list do
  @products.each do |product|
    xml.product do
      xml.name(product.name)
      xml.description(product.name)
```

```
xml.image(product.image)
xml.camera(product.camera)
xml.video(product.video)
xml.triband(product.triband)
xml.price(product.price, :currency => "USD")
end
end
end
```

8. Тестирование темплейта билдера. Откройте следующий URL:

<http://localhost:3000/store/productlist>

Вы увидите XML документ как на рисунке ниже:

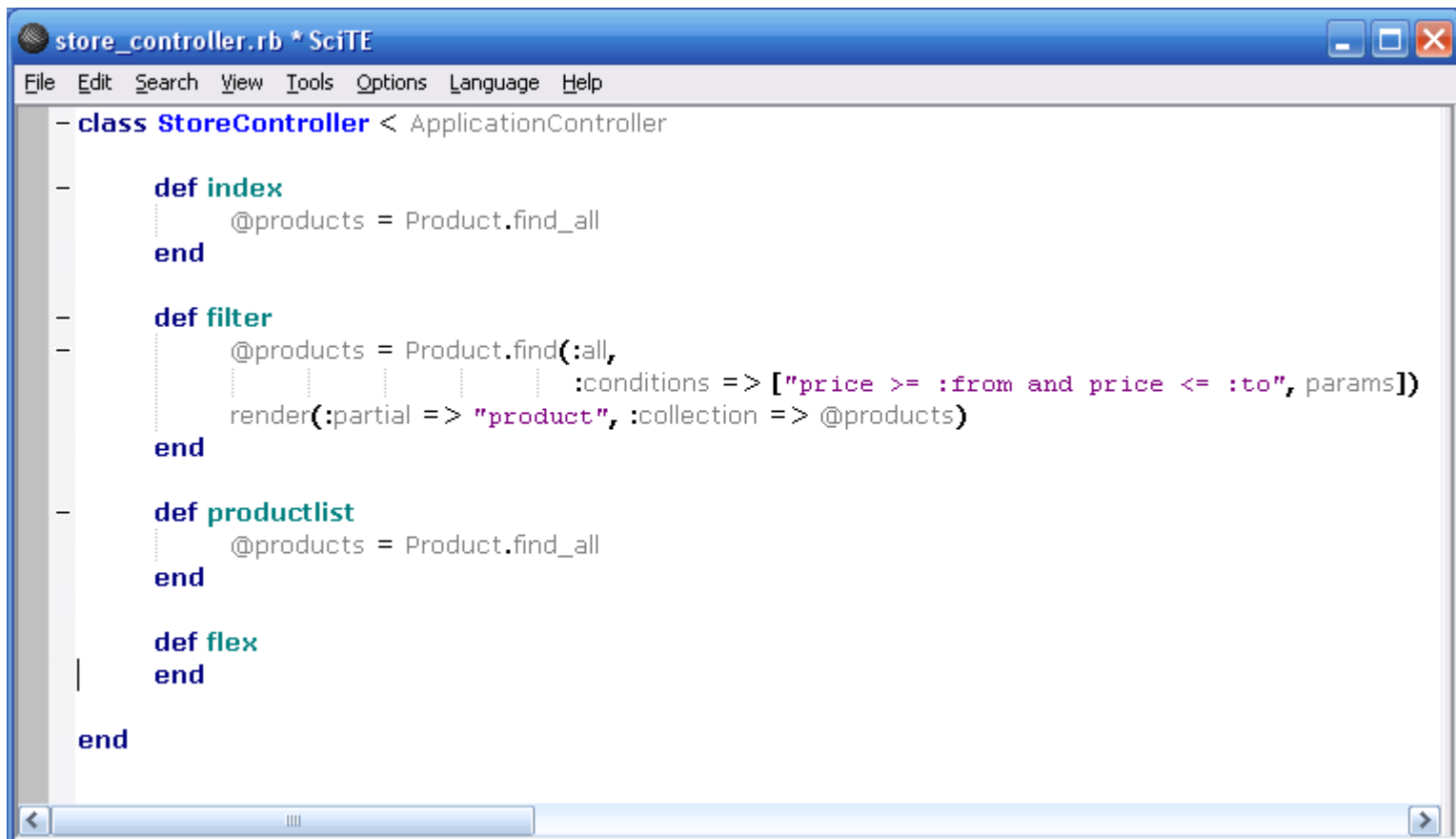


Добавление Flex приложения в качестве фронт-энда Rails приложения

В этом разделе мы добавим Flex приложение, чтобы улучшить внешний вид каталога. Фильтрация все еще реализована в HTML.

Основанный на Flex каталог продуктов будет использовать билдер-темплейты, созданные в предыдущем разделе, для извлечения данных из каталога. Использование эффектов и прозрачности обеспечивает более мягкие переходы между выбором продуктов. Эти переходы реализуют лучшую практику «визуальной непрерывности» в проектировании интерфейса пользователя и включают визуальные эффекты, показывающие какие продукты были убраны в результате фильтрации, а какие наоборот были добавлены.

1. Определение действия flex
 - Откройте файл store_controller.rb в c:\rails\flexstore\app\controllers
 - Определите действие flex следующим образом:



```
store_controller.rb * SciTE
File Edit Search View Tools Options Language Help
- class StoreController < ApplicationController
-
-   def index
-     @products = Product.find_all
-   end
-
-   def filter
-     @products = Product.find(:all,
-                             :conditions => ["price >= :from and price <= :to", params])
-     render(:partial => "product", :collection => @products)
-   end
-
-   def productlist
-     @products = Product.find_all
-   end
-
-   def flex
-   end
-
- end
```

2. Разместите файл catalog.swf (скомпилированная версия Flex каталога продуктов).
 - Загрузите flexcatalog.zip [здесь](#).
 - Извлеките flexcatalog.zip в c:\rails\flexstore\public

Заметки:

- Приложение было создано с помощью Flex фреймворка. Загрузить исходники, переделанные для Flash Player 9 вы можете [здесь](#).
- Приложение создано с помощью Flex 2 beta 3 и требует наличия Flash Player 9 (загрузить можно [здесь](#)).

Прим. пер. Автор делал исходники и swf для Flex 2 beta 2 и Flash Player 8,5 beta 2. При желании скачать их можно [здесь\(swf\)](#) и [здесь\(mxml\)](#)

3. Первое сокращение

Первым шагом мы просто заменяем каталог продуктов в HTML на его версию во Flex.

- Создайте файл flex.rhtml в c:\rails\flexstore\views\store
- Скопируйте код ниже во flex.rhtml

```

<html>
<head>
<title>Flexstore on Rails</title>
<%= stylesheet_link_tag "flexstore", :media => "all" %>
<script type="text/javascript" src="/flex/embedflash.js" ></script>
<script type="text/javascript" src="/flex/FABridge.js" ></script>
<script>

function filter() {
var from = document.getElementById("from").value;
var to = document.getElementById("to").value;
var flexApp = FABridge.store.root();
flexApp.filter(from, to);
}

</script>
</head>

<body>

<div id="left">
Select your price range:<br />
<br />
From:<br />
<input type="text" id="from" value="0"/>
<br />
<br />
To:<br />
<input type="text" id="to" value="1000"/><br />
<br />
<input type="submit" value="Filter" onclick="filter()"/>

</div>

<div id="flex">
<script>embedFlash("flexApp", "/flex/catalog.swf", 690, 510,
"bridgeName=store");</script>
</div>

</body>
</html>

```

- Протестируем приложение

<http://localhost:3000/store/flex>

4. Оптимизация пользовательского интерфейса

Мы будем использовать слайдеры (из библиотеки Yahoo UI), чтобы выбирать ценовую категорию. Каталог продуктов будет реагировать на перетаскивание слайдеров.

- Установите слайдеры Yahoo. Загрузите slider.zip [здесь](#) и и звлеките файл в c:\rails\flexstore\public.
- Перепишите содержимое файла flex.rhtml следующим образом:

```

<html>
<head>
<title>Flexstore on Rails</title>
<%= stylesheet_link_tag "flexstore", :media => "all" %>
<script type="text/javascript" src="/flex/embedflash.js" ></script>
<script type="text/javascript" src="/flex/FABridge.js" ></script>
<script type="text/javascript" src="/yui/YAHOO.js" ></script>
<script type="text/javascript" src="/yui/event.js" ></script>
<script type="text/javascript" src="/yui/dom.js" ></script>
<script type="text/javascript" src="/yui/dragdrop.js" ></script>
<script type="text/javascript" src="/yui/slider.js" ></script>

<script type="text/javascript">

var slider1, slider2;

function init() {

document.getElementById("camera").checked = false;
document.getElementById("video").checked = false;
document.getElementById("triband").checked = false;

slider1 = YAHOO.widget.Slider.getHorizSlider("slider1", "thumb1", 0, 200, 2);
slider1.onChange = function(offsetFromStart) {
var newValue = offsetFromStart * 5;
document.getElementById("value1").innerHTML = newValue;
var flexApp = FABridge.store.root();
flexApp.setMinimum(newValue);
};
slider1.onMouseUp = function() {
var flexApp = FABridge.store.root();
flexApp.layoutTiles();
};
slider2 = YAHOO.widget.Slider.getHorizSlider("slider2", "thumb2", 200, 0, 2);
slider2.onChange = function(offsetFromStart) {
var newValue = 1000 + offsetFromStart * 5;
document.getElementById("value2").innerHTML = newValue;
var flexApp = FABridge.store.root();
flexApp.setMaximum(newValue);
};
slider2.onMouseUp = function() {
var flexApp = FABridge.store.root();
flexApp.layoutTiles();
};
}

window.onload = init;

</script>

</head>
<body style="margin-top:20px; margin-left:20px;">

Select your price range:<br />
<br />
<div id="slider1" class="sliderBG"
onkeypress="return handleHorizSliderKey(this, YAHOO.util.Event.getEvent(event))" >
Minimum: $<span id="value1" >0</span>
<div id="thumb1" class="thumb"></div>
</div>
<br /><br />

```

```

<div id="slider2" class="sliderBG"
onkeypress="return handleHorizSliderKey(this, YAHOO.util.Event.getEvent(event))" >
Maximum: $<span id="value2" >1000</span>
<div id="thumb2" class="thumb" style="left:200px"></div>
</div>

```

```

<br /><br />

```

Select the required features
on your mobile device:


```

<br />
<table>
<tr><td><input type="checkbox" id="camera" onClick="var app =
FABridge.store.root();app.setCamera(this.checked);"/></td><td>Camera</td></tr>
<tr><td><input type="checkbox" id="video" onClick="var app =
FABridge.store.root();app.setVideo(this.checked);"/></td><td>Video</td></tr>
<tr><td><input type="checkbox" id="triband" onClick="var app =
FABridge.store.root();app.setTriband(this.checked);"/></td><td>Triband</td></tr>
</table>

```

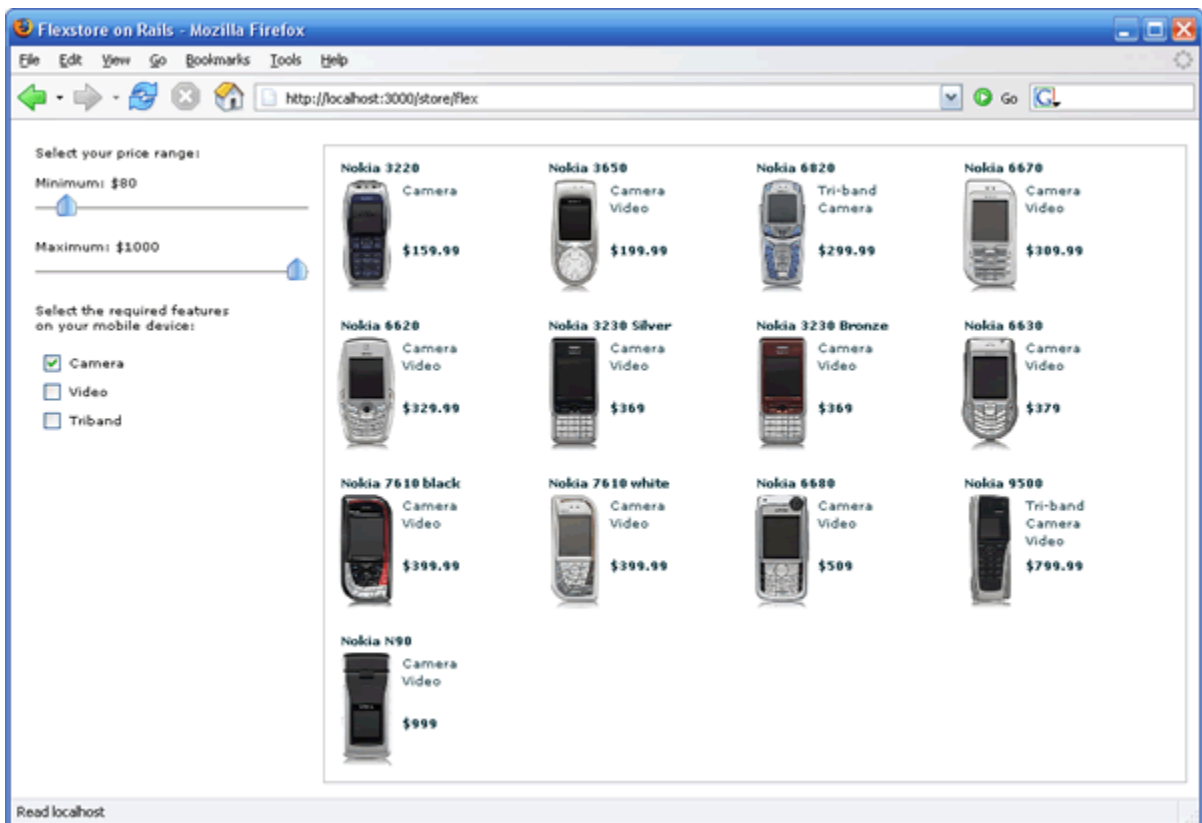
```

<div id="flex">
<script>embedFlash("flexApp", "/flex/catalog.swf", 690, 510,
"bridgeName=store");</script>
</div>
</body>
</html>

```

- Протестируйте приложение

<http://localhost:3000/store/flex>



Ресурсы:

[Ruby on Rails web site](#)

Flex on [Adobe Labs](#)