**New:**[**Ruby on Rails 4.0 version of this book!**](http://www.amazon.com/Ruby-on-Rails-4-0-ebook/dp/B00E25KVLW?tag=xyzpubcom02-20)

**4.5. Populating the Database with seeds.rb**

With the file db/seeds.rb, the Rails gods have given us a way of feeding default values easily and quickly to a fresh installation. This is a normal Ruby program within the Rails environment. You have full access to all classes and methods of your application.

So you do not need to enter everything manually with **rails console** in order to make the records created in [the section called “create”](http://www.xyzpub.com/en/ruby-on-rails/3.2/activerecord_datensaetze_hinzufuegen.html#activerecord_create)available in a new Rails application, but you can simply use the following file db/seeds.rb:

Country.create(name: 'Germany', population: 81831000)

Country.create(name: 'France', population: 65447374)

Country.create(name: 'Belgium', population: 10839905)

Country.create(name: 'Netherlands', population: 16680000)

You then populate it with data via **rake db:seed**. To be on the safe side, you should always set up the database from scratch with **rake db:setup** in the context of this book and then automatically populate it with the file db/seeds.rb. Here is what is looks like:

$ rake db:setup

db/development.sqlite3 already exists

-- create\_table("countries", {:force=>true})

-> 0.0175s

-- initialize\_schema\_migrations\_table()

-> 0.0005s

-- assume\_migrated\_upto\_version(20121114110230, ["/Users/xyz/sandbox/europe/db/migrate"])

-> 0.0006s

$

I use the file db/seeds.rb at this point because it offers a simple mechanism for filling an empty database with default values. In the course of this book, this will make it easier for us to set up quick example scenarios.

**It's all just Ruby code**

The db/seeds.rb is a Ruby program. Correspondingly, we can also use the following approach as an alternative:

country\_list = [

[ "Germany", 81831000 ],

[ "France", 65447374 ],

[ "Belgium", 10839905 ],

[ "Netherlands", 16680000 ]

]

country\_list.each do |name, population|

Country.create( name: name, population: population )

end

The result is the same. I am showing you this example to make it clear that you can program completely normally within the filedb/seeds.rb.

**Generating seeds.rb From Existing Data**

Sometimes it can be useful to export the current data pool of a Rails application into a db/seeds.rb. While writing this book, I encountered this problem in almost every chapter. Unfortunately, there is no standard approach for this. I am showing you what you can do in this case. There are other, more complex scenarios that can be derived from my approach.

We create our own little rake task for that. That can be done by creating the file lib/tasks/export.rake with the following content:

namespace :export do

desc "Prints Country.all in a seeds.rb way."

task :seeds\_format => :environment do

Country.order(:id).all.each do |country|

puts "Country.create(#{country.serializable\_hash.delete\_if {|key, value| ['created\_at','updated\_at','id'].include?(key)}.to\_s.gsub(/[{}]/,'')})"

end

end

end

Then you can call the corresponding rake task with the command **rake export:seeds\_format**:

$ rake export:seeds\_format

Country.create("name"=>"Germany", "population"=>81831000)

Country.create("name"=>"France", "population"=>65447374)

Country.create("name"=>"Belgium", "population"=>10839905)

Country.create("name"=>"Netherlands", "population"=>16680000)

$

You can either expand this program so that the output is written directly into the db/seeds.rb or you can simply use the shell:

$ rake export:seeds\_format > db/seeds.rb

$

**UTF-8**

If you want to use UTF-8 characters in your db/seeds.rb, then you need to enter the line

# ruby encoding: utf-8

at the beginning of the file.

Example:

# ruby encoding: utf-8

Country.create(name: 'Germany', population: 81831000)

Country.create(name: 'France', population: 65447374)

Country.create(name: 'Belgium', population: 10839905)

Country.create(name: 'Netherlands', population: 16680000)

Country.create(name: 'Austria', population: 8440465)

Country.create(name: 'Republika e Shqipërisë', population: 2831741)

29May2013

* [Wireframes](http://blog.dynamic50.com/category/wireframes/)
* [development](http://blog.dynamic50.com/tag/development/),
* [dynamic50](http://blog.dynamic50.com/tag/dynamic50/),
* [Training](http://blog.dynamic50.com/tag/training/),
* [Wireframes](http://blog.dynamic50.com/tag/wireframes/)

Wireframes are used to design the basic structure of a website service using simple line drawings. The wireframes are used to show the placements of elements and functionality. Using wire framing is a great way of starting a project as you can edit and make changes to the layout before the more complex design and coding is underway.

Wireframes commonly take into account user needs and user journey through the page, online tool, or mobile app. It is an important part of the “interactive design process” and allows the client to focus on the layout without the distraction of colour, type and other design and marketing elements. It is a good way of working closely with the client to create a visual understanding of a page. This is important at an early stage in a project to ensure the web service meets the stakeholders approval before the coding and creative process has begun.

Quicker to adapt:

A wireframe is quicker and cheaper to adapt than a page where all the coding and design elements have been put into place. Being part of an iterative process allows the client and design team to make alterations until the page meets the outlined objectives.

Project lifecycle:

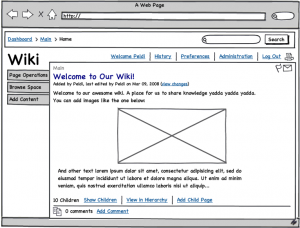
Wireframing is used at the early stages of the design process. A wireframe prototype can also be used for usability testing and gain insightful feedback prior to development.

Wireframes can be sketched out on paper or can be put together using Adobe photoshop, illustrator, or other graphic software. There is some good software that allows you to do this easily, here at [Dynamic50](http://dynamic50.com/) we use specialist wire framing software [Balsamiq Mockups](http://www.balsamiq.com/" \o "Balsamiq Mockups" \t "_blank).

Wireframe elements:

All of the important elements of a web page can be depicted in the wireframe. Using basic shapes and labels rather than graphics. These elements may include:

* - Simple layout structure such as navigation buttons for the main areas of the site.
* - A shape or wording to show the company logo
* - Content areas for articles and other features. How will they be arranged on the page?
* - Images and video placeholders
* - Calls to action such as user login area and search box

[](http://blog.dynamic50.com/wp-content/uploads/2013/05/website-wireframe-example.png)

Conclusion:

Wireframes should be used early in the project lifecycle to gain client approval on the layout and key navigation for the page. This will allow the project to be managed effectively and will save both time and money in changes that would otherwise need to be made at later stages. This allows for a more accurate and productive project outcome.

If you’re interested in learning more take a look at our [portfolio](http://dynamic50.com/portfolio/clients), or if you would like to talk to us about a new or existing project then [contact us](http://dynamic50.com/contacts/new)Â to see how we can help you.