Rails worksheet 1: Playing with migrations
CSC 340

1. Create an application called ws1.
2. Make a model called Product with a name of type string and a description of type text.
3. Examine the migration and note the name of the table.
4. Apply the migration using rake.
5. Go to the console view and use Product.column_names to look at the columns, and then exit out of the console view.
6. Now, let's assume that you forgot to include an attribute in your model (a column in the table). How could you change your model? There are two ways to do it: method 1, and method 2, listed below. For this worksheet, please do ***both***. Just follow the instructions in order to try out both ways to change the model.

Method 1: Rollback the migration

1- Here, you simply un-apply the migration that you applied earlier by typing: rake db:rollback
2- Note what is written on the terminal: drop_table (products). So the table is not there anymore. (Some other files related to the model are still present, however.)
3- Go to the console view (type: rails console) and type: Product.column_names. What happens?
4- Exit out of the console view.

Method 2: Creating new migrations using the migration generator

If you need to add a new column or drop a column from your existing model, you can generate a new migration to do that.

1- We rolled back the migration we made earlier. So let's apply it again now, using rake.
2- You can check that your model is created by going to the console view and typing Product.column_names.
3- You can create a new migration of the form: "AddColumnToTable" or "RemoveColumnFromTable" followed by a list of column names and types. The migration will then be created according to your specifications. For example, to add a column called part_number to the table products, you would do:
rails generate migration AddPartNumberToProducts part_number:string.
4- Look at the generated migration file. Note that Rails automatically filled in all necessary code by intelligently parsing the migration name you provided.
5- Apply the migration using rake so that it can take effect.
6- Check the column names again and verify that part_number is now included. Then exit out of the console view.

7- Let's remove the column we just added, type: rails generate migration RemovePartNumberFromProducts part_number:string.
8- Apply the migration.
9- Examine your model and make sure that the attribute part_number was indeed removed.
10- Now, let's make a migration that adds two columns, type: rails generate migration AddDetailsToProducts part_number:string price:decimal
11- Look at the migration code to see what was generated.
12- Apply the migration.
13- Note that when you use rake db:migrate to apply migrations, only those that haven't yet been applied will be applied. So for example, since we've already applied the RemovePartNumberFromProducts migration, it won't be applied again here, even though the migration file is still in the db/migrate folder.
14- Examine your model to make sure your migration took effect.