Tiny Blog

# Introduction

This blog application has been created by Harsh Bhasin ([hersh\_b@yahoo.com](mailto:hersh_b@yahoo.com), <http://harshbhasin.com>) as a coding exercise.

# Setup

1. Database: this application uses SQL Express. A database backup, as well as table creation scripts has been provided in the folder called sql.
2. Register for a login and then sign in. Authorized users get to see Edit/Update/New links next to posts , and also an Admin Tab.

# Design Considerations

1. **Authentication**: I have created my own implementation of the ASP.NET Membership provider. It uses a single table (users) instead of the huge ASP.NET membership database which would have been overkill for this project.
2. **ORM**: For my ORM, I choose Massive. I could have used Code First and the Repository pattern but I like Massive a lot (https://github.com/robconery/massive/). It is written by Rob Conery, who was part of the MVC3 core team at Microsoft and now runs TekPub. It is similar to the Active Record pattern of rails and It’s design is based on the Do Not Repeat yourself (DRY ) principle. In an ORM like Code First or Entity Framework you have your object model, and your data model and a mapper in-between (Edmx or DBSet in Code First.) However in the rails Active Record world, and also in Massive, the database is your model – there is no POCO class representation of your database or any sort of mapper. So, there are lots of steps skipped and less codes to write and maintain.
3. **Validations:** The regular MVC3 way of providing validations is to decorate the POCO class model with attributes which then generates html5 data tags and then uses the jquery.validate library to validate. I find this very cumbersome and have written my own helpers to generate the required html5 data tags directly without resorting to the POCO “decoration” route. This has been implemented on the “Register” page. If you submit the “Register” for a new account page, without filling the form, you will see the validation happen. This is using my own helpers, bypassing the MVC3 decoration.
4. **Client Side Scripting using Knockoutjs, JQuery and REST:** The “Admin” page has been written entirely client side with JQuery using Knockoutjs and REST-full calls.
5. **Base Controller Class**: A base controller class (CruddyController) has been created—any controller inheriting from it gets all the RESTFul actions of the ancestor. Refer to the PostsController class that inherits from it.

# Aesthetic Considerations

**Twitter Bootstrap CSS Library used**: I have used the Twitter Bootstrap library (<http://twitter.github.com/bootstrap/>) for my CSS styling and layout.

# Time Taken

I was unable to sit more than 20 minutes at a stretch because of a Sinus Surgery I had last week. Taking frequent breaks, it took me about 6 hours with breaks to complete. I could have gone further and implemented more bonus stuff but my body rebelled.

# Coding Exercise Coverage

## Basics

All points have been covered

1. Show’s home page
2. Clicking blog title shows details
3. Can Add new blog post
4. RSS feed available
5. Styling has been done using Twitter Bootstrap.
6. Persists the data locally (to a SQL Express database.)

## Bonus

The following “bonus” points have been covered

1. #1 Admin can login to add, edit or delete posts. Admin page has been developed to be highly responsive using client-side scripting.
2. #8 Use CSS to make the home page look nice: Twitter Bootstrap used
3. #9 and # 10: Use Entity Framework/ Code First: I have used Massive and my justification listed in the “Design Consideration” section.
4. #12. Use inversion of Control to inject dependencies. I am using **Ninject** to inject the tokenStore into the controller constructor. The tokenstore is used to hold information about the logged-in user.

# Finally

It was fun doing this project as I’d not done a blog in MVC3 yet—(had done one in rails: <http://harshbhasin.com>.) Please let me have your code review feedback.

Thanks

Harsh Bhasin

Hersh\_b@yahoo.com