Tutorial 9 – Blog Web Application (pt 2*)

(*) Part 1 of this Blog Web Application was done in Tutorial 8.

❖ Objectives:

- Posts CRUD (add post, list posts, edit post, delete post)
- Home page showing some latest posts with pagination
- Author profile page showing author info and his posts
- Post detail page

❖ Tasks:

1. Create an entity class named `Post` and create a *one-to-many* relationship between `User` and `Post`. (one user has many posts).
    - Suggested attributes for Post entity: `id`, `title`, `slug`, `excerpt`, `fulltext`, `viewCount`, `dateCreated`, `dateModified`.

2. Create two Thymeleaf layouts called `_mainLayout.html` and `_memberLayout.html`. The pages for blog members should use the `_memberLayout.html` layout while other pages should use the `_mainLayout.html` layout.
    - The main layout should have a header section that contains a navigation bar and a footer section at the bottom (see **Figure 1**).
    - The member layout should have a menu on the left and the main working area on the right (see **Figure 2**).

3. Create a `MemberController` class which has these controller methods:
    - `listPosts` — shows a list of posts at URL `/member/post/list`
    - `addPost` — shows a form to add a Post at URL `/member/post/add` (GET). In this page and the Edit Post page, you should integrate a *WYSIWYG editor* like TinyMCE for users to edit the post content. Check out these links:
        - `https://www.tiny.cloud/`
        - `https://stackoverflow.com/questions/47446940/how-to-get-tinymce-interface-to-appear-in-thymeleaf-form-textarea`
    - `addPostHandle` — handles the submitted form to add a post at URL `/member/post/add` (POST)
    - `deletePost` — deletes a post when user visits `/member/post/del/{id}` (GET). (*) It would be better if you shows a confirmation popup dialog (using JavaScript) before redirecting to this URL to delete the post.
    - `editPost` — shows a form to edit a Post at `/member/post/edit/{id}` (GET)

- `editPostHandle` — handles the submitted form to edit a post at the URL `/member/post/edit/{id}` (POST)

4. Create a `HomeController` class which has these controller methods:
   - `home` — shows the homepage listing 2 latest posts, showing post title, author, published time and excerpt. The post title should be a link to the post's detail page. This homepage's URL is `http://localhost:8080/`. Below the posts, there should be a "See more…" link which will lead to the next page of the homepage, showing the next 2 posts. The URL for the 2nd page of the homepage should be `http://localhost:8080/?p=2`.
   - `authorProfile` — shows the author profile page which shows author info (name, profile picture, introduction text, social links…) and all posts by an author (user), sorted by published time descending. URL: `/author/{username}`. This page should be paginated similarly to the homepage, with URLs to the pages being something like: `/author/{username}/?p=3`.
   - `postDetail` — shows all of a post's details. URL: `/post/{slug}/`. For example: `http://localhost:8080/post/worth-a-thousand-words/`

Figure 1: Suggested homepage layout

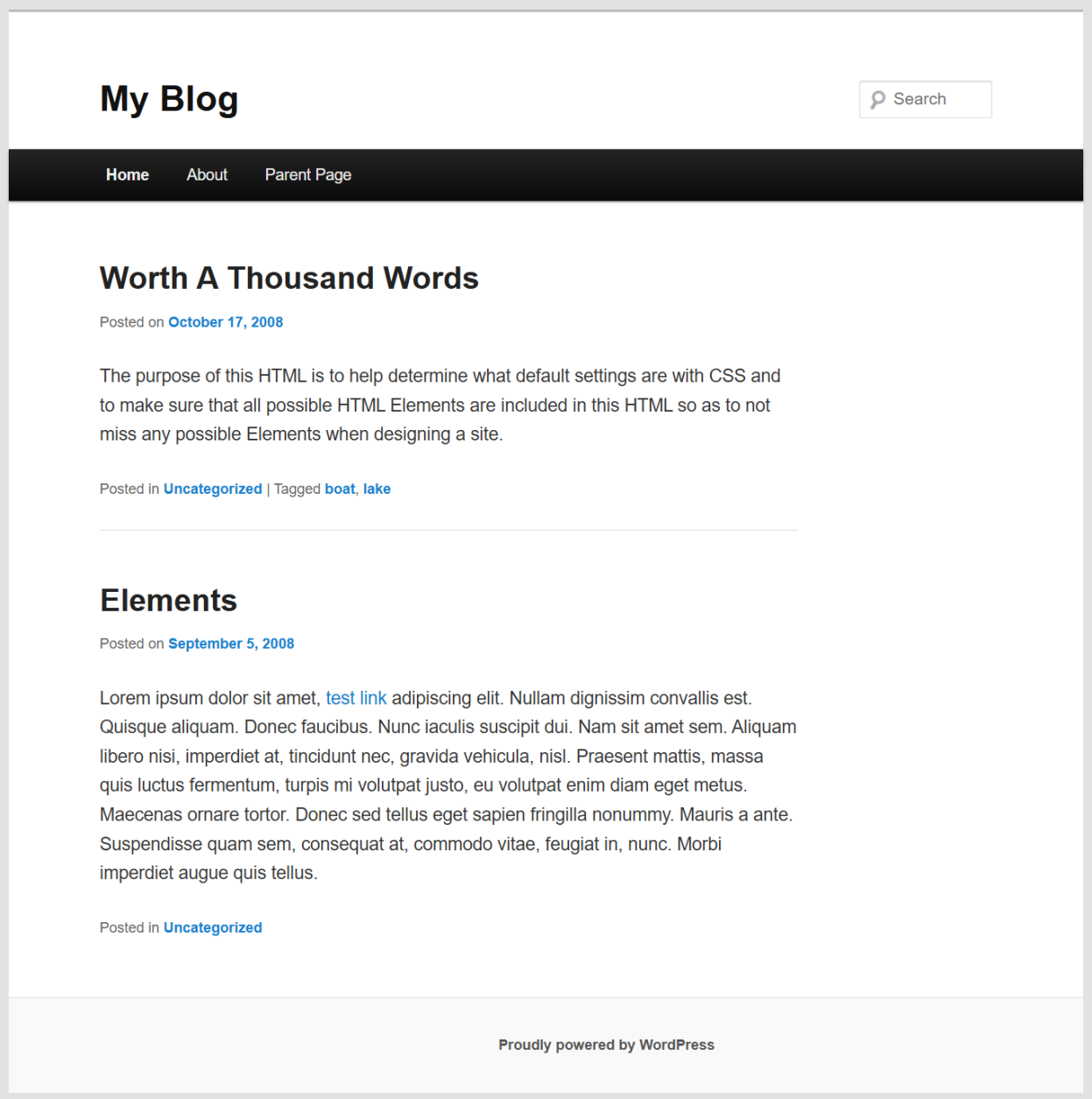
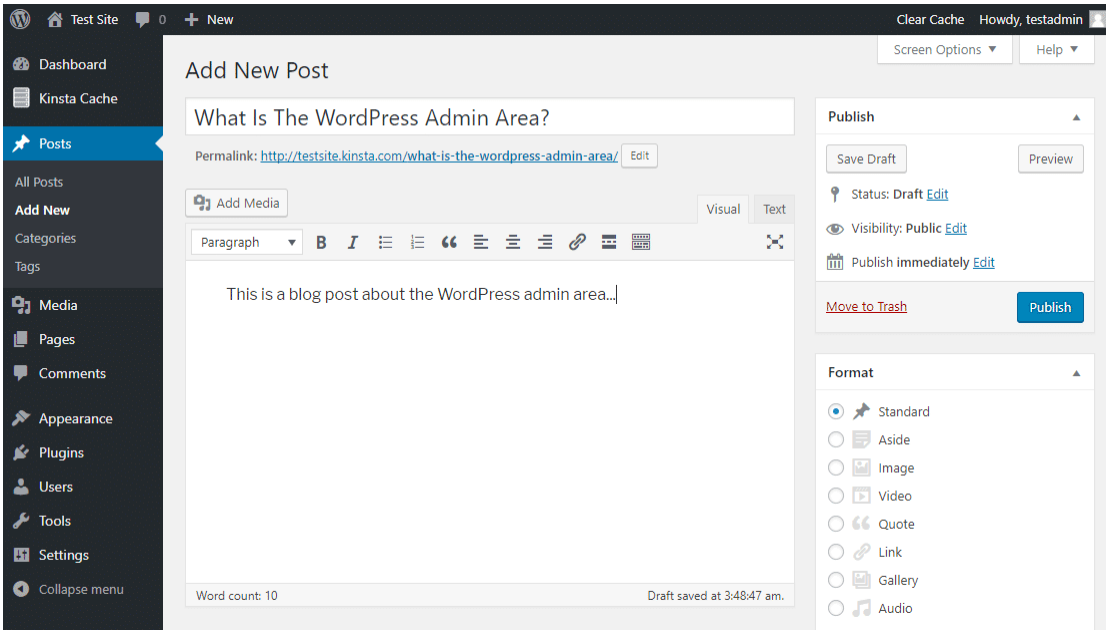


Figure 2: Suggested member layout



❖ Run, debug & submit your project

Once finished, compress your project into a `.zip` file and submit the file to the tutorial's submission box.