SQL Bros Blog

"Sarah's Thoughts"

Gage Gabaldon

Yin Yu Chen

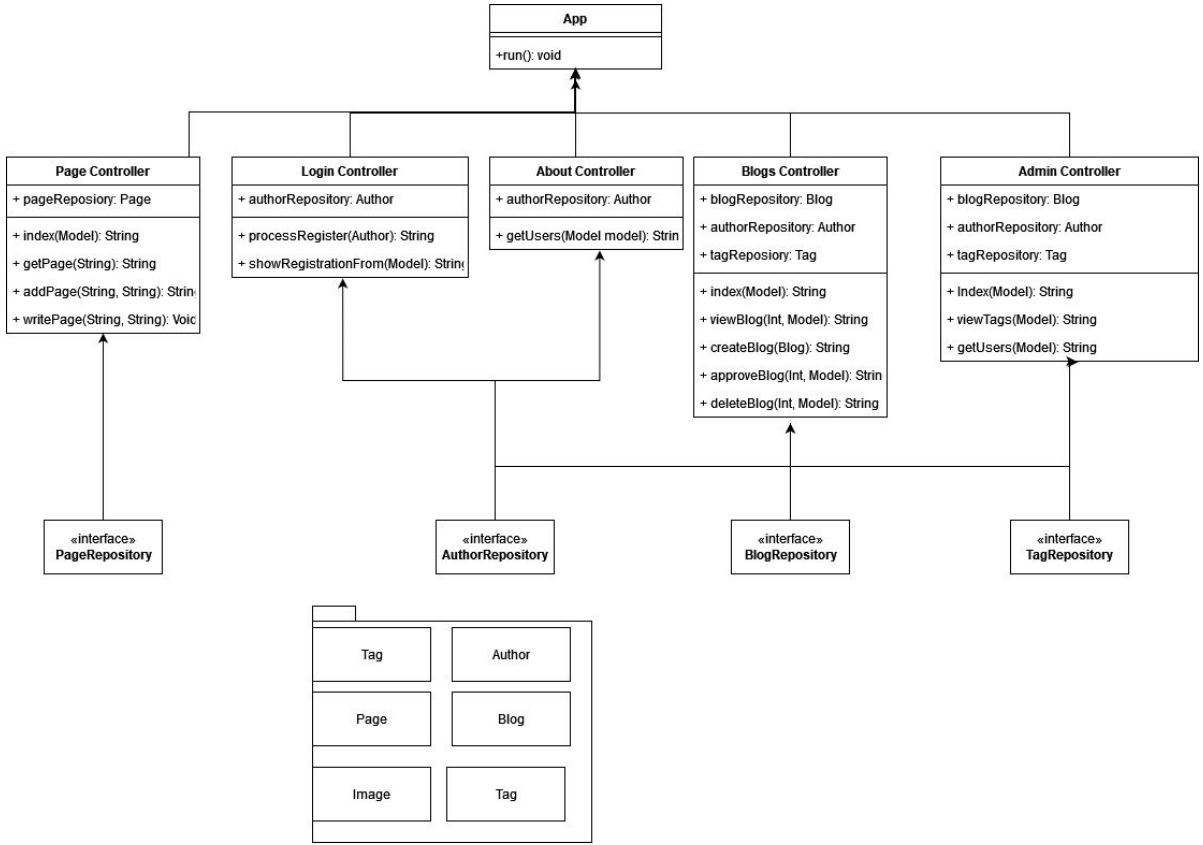
Cole Bruton

Expectations

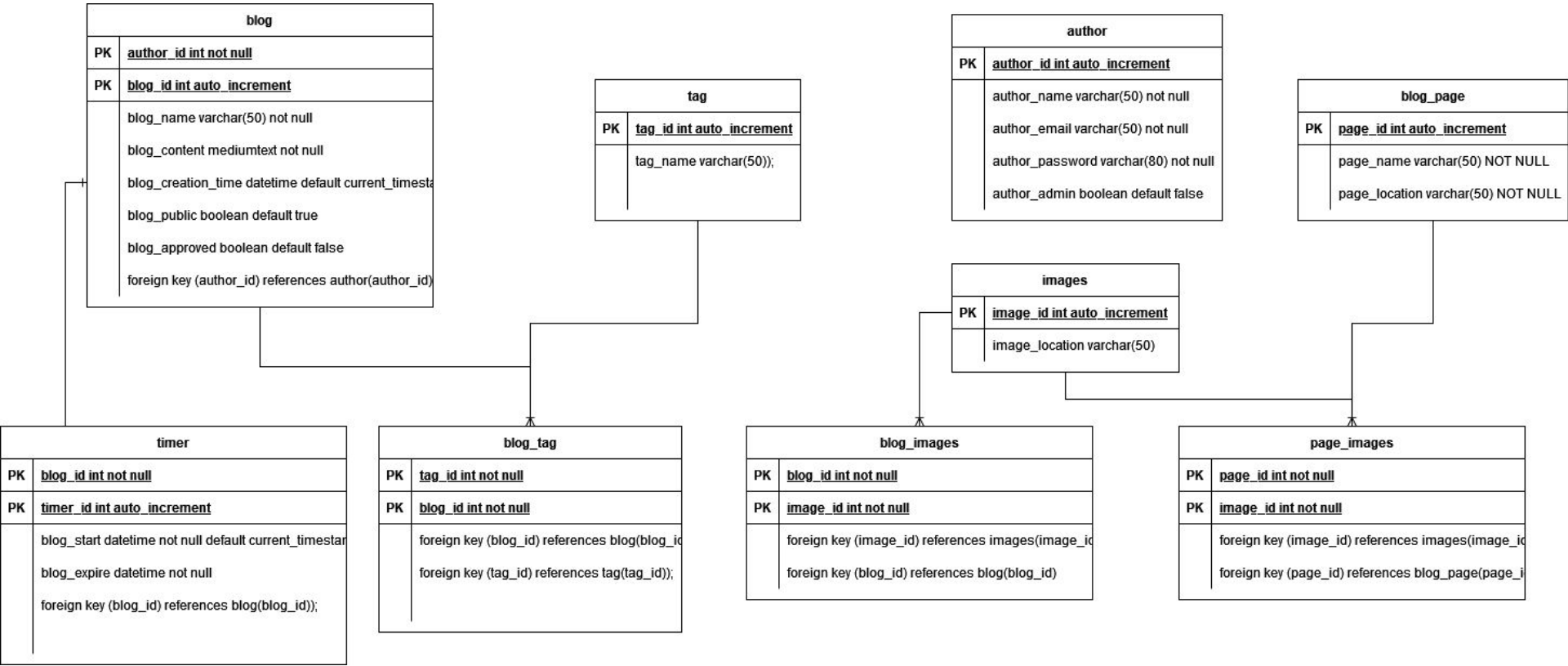
- Small company needing way to display blog posts to all visitors.
- Owner also wants a separate ability to post certain pages that only they can edit/create.
  - Static pages
- Does not want to write HTML.
  - Use of TinyMCE
- Wants #hashtag tag abilities - along with search by tag(s).
- User accounts to allow authorized users publish posts.
  - Admin approval of posts.
- Scheduled publish and removal of posts… **(WIP)** - Finished

https://the-software-guild.s3.amazonaws.com/java-ddwa/v2.7-2010/cms-blog-specification.pdf

App UML


App
+run(): void
Page Controller
+ pageReposiory: Page
+ index(Model): String
+ getPage(String): String
+ addPage(String, String): Strin
+ writePage(String, String): Voic
Login Controller
+ authorRepository: Author
+ processRegister(Author): String
+ showRegistrationFrom(Model): Strin
About Controller
+ authorRepository: Author
+ getUsers(Model model): Strin
Blogs Controller
+ blogRepository: Blog
+ authorRepository: Author
+ tagReposiory: Tag
+ index(Model): String
+ viewBlog(Int, Model): String
+ createBlog(Blog): String
+ approveBlog(Int, Model): Strin
+ deleteBlog(Int, Model): String
Admin Controller
+ blogRepository: Blog
+ authorRepository: Author
+ tagRepository: Tag
+ Index(Model): String
+ viewTags(Model): String
+ getUsers(Model): String
«interface» PageRepository
«interface» AuthorRepository
«interface» BlogRepository
«interface» TagRepository
Tag
Author
Page
Blog
Image
Tag

App ERD


blog
PK author_id int not null
PK blog_id int auto_increment
blog_name varchar(50) not null
blog_content mediumtext not null
blog_creation_time datetime default current_timesta
blog_public boolean default true
blog_approved boolean default false
foreign key (author_id) references author(author_id)
tag
PK tag_id int auto_increment
tag_name varchar(50));
author
PK author_id int auto_increment
author_name varchar(50) not null
author_email varchar(50) not null
author_password varchar(80) not null
author_admin boolean default false
blog_page
PK page_id int auto_increment
page_name varchar(50) NOT NULL
page_location varchar(50) NOT NULL
images
PK image_id int auto_increment
image_location varchar(50)
timer
PK blog_id int not null
PK timer_id int auto_increment
blog_start datetime not null default current_timestar
blog_expire datetime not null
foreign key (blog_id) references blog(blog_id));
blog_tag
PK tag_id int not null
PK blog_id int not null
foreign key (blog_id) references blog(blog_id
foreign key (tag_id) references tag(tag_id));
blog_images
PK blog_id int not null
PK image_id int not null
foreign key (image_id) references images(image_id
foreign key (blog_id) references blog(blog_id)
page_images
PK page_id int not null
PK image_id int not null
foreign key (image_id) references images(image_id
foreign key (page_id) references blog_page(page_i

TinyMCE

- Open source Rich-text editor
- Allows image placement and formatting of text.
  - Publish written content to HTML format
  - No need for knowledge of HTML
- Freemium app with API key
- Works with many commonly used libraries and Frameworks.

```
<head>
  <script
src="https://cdn.tiny.cloud/1/2soxcti9aqufj4t771jzv
g7m6y0p9m3ta63t8t3i3xtyk8au/tinymce/5/tinymce.min.j
s" referrerpolicy="origin"></script>
</head>
<body>
  <textarea>
    Welcome to TinyMCE!
  </textarea>
  <script>
    tinymce.init({
      selector: 'textarea',
      plugins: 'a11ychecker advcode casechange
export formatpainter linkchecker autolink lists
checklist media mediaembed pageembed permanentpen
powerpaste table advtable tinycomments
tinymcespellchecker',
      toolbar: 'a11ycheck addcomment showcomments
casechange checklist code export formatpainter
pageembed permanentpen table',
      toolbar_mode: 'floating',
      tinycomments_mode: 'embedded',
      tinycomments_author: 'Author name',
    });
  </script>
</body>
```

HTML

- Creation of a boilerplate template that can be the basis for all pages.
  - Navigation bar at top with website Title and links
  - Footer at bottom with links to social media accounts.
- All blog posts created through the TinyMCE will be added to blog page with same formatting.
- Static pages created through TinyMCE will be created and stored with added HTML formatting to match.

Approved Blogs

First Blog

Follow on Social!

HTML - Thymeleaf

- Use of Thymeleaf in HTML to make calls to information stored in the models/database.
- Also used to add created information to the models and database for storage.
- Thymeleaf Attributes used within existing HTML tags.
    - Generally start with “th:” followed by the attribute.

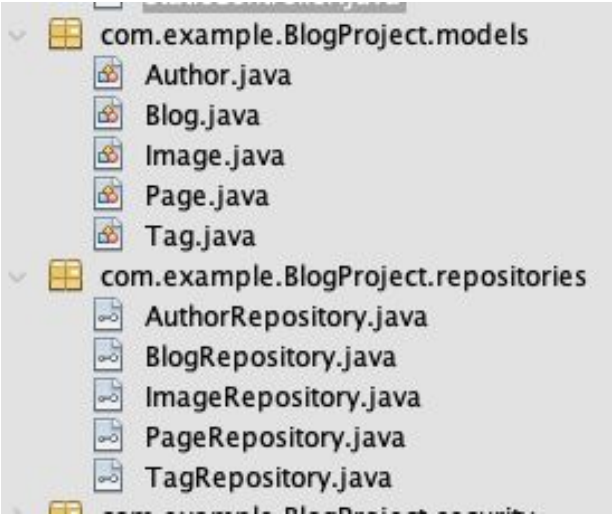
Controller

- The App is using Java with the spring framework to deliver the html web pages to the user.
  - Processes requests and redirects to appropriate website pages.
- Individual controllers are used to process the GET and POST requests to the server.
  - Blog Controller for blog posts
  - Admin Controller for processing user accounts and admin processes
  - Login Controller for redirecting to proper page for login or account creation.
  - Static Controller for processing admin creations of non-editable pages.

```
@Controller
public class BlogsController {

    @Autowired
    BlogRepository blogs;
```

Model

Blog	Object used to store information about each blog created.
Image	Object to store image information
Tag	References each tag and blogs that are linked.
Author	All the information regarding the author
Page	Used for the static pages that only the owner can create.



Model

Each of the classes contains a basic setup like this.

Use of annotations equates the entities and variables to their corresponding tables in the database.

Each model has a Repository class for interacting with the database itself.

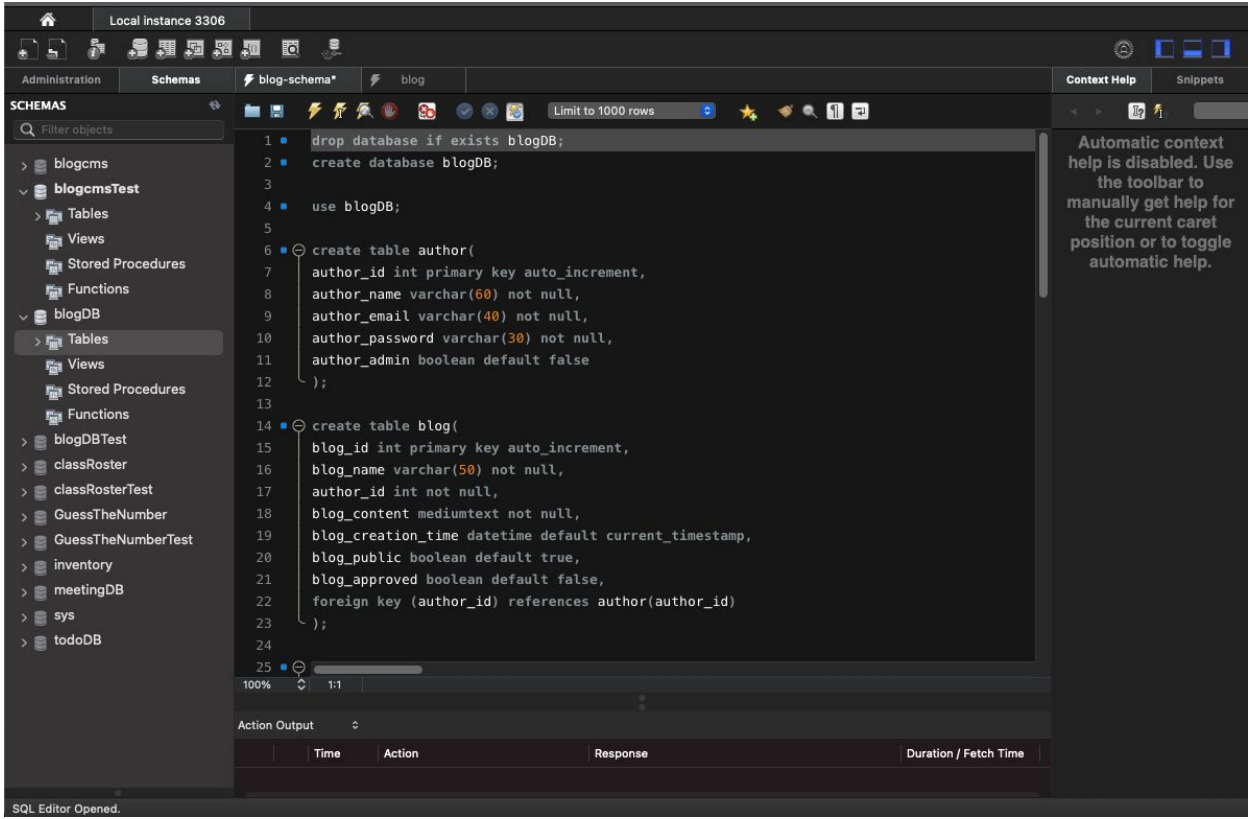
```java
@Entity
public class Tag {
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Id
    private int tagId;

    @ManyToMany
    @JoinTable(name = "blog_tag",
            joinColumns = {@JoinColumn(name = "tag_id")},
            inverseJoinColumns = {@JoinColumn(name = "blog_id")})
    private List<Blog> blog;
```

```java
@Entity
public class Blog {
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Id
    private int blogId;

    @Column(nullable = false)
    private String blogName;

    @Column(nullable = false)
    private String blogContent;

    @Column
    private LocalDateTime blogCreationTime;

    @Column
    private boolean blogPublic;

    @Column
    private boolean blogApproved;

    @ManyToOne
    @JoinColumn(name="authorId", nullable=false)
    private Author author;

    @ManyToMany(mappedBy = "blog")
    private List<Image> image;

    @ManyToMany(mappedBy = "blog")
    private List<Tag> tag;
```

MySql aka backend

- Database implementation for long term storage of data.
- Data stored in Model objects is sent to database.
    - JPA implementation using annotations.
        - Removed the need for SQL statements.
- JDBC Template

MySQL Workbench



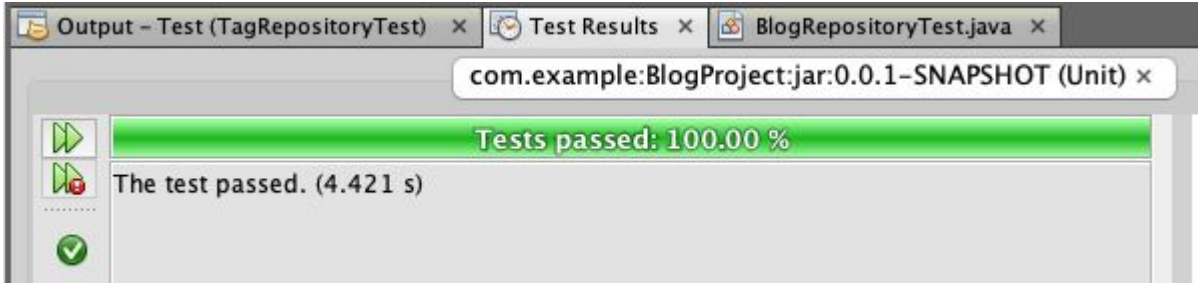
Testing

Spring testing with junit

Testing of the repositories and simple functions.

```java
@Repository
public interface BlogRepository extends JpaRepository<Blog, Integer>{
    @Query(value = "SELECT * FROM blog WHERE blog_approved = true", nativeQuery = true)
    public List <Blog> findByApproved();
    @Query(value = "SELECT * FROM blog WHERE blog_approved = false", nativeQuery = true)
    public List <Blog> findByUnApprove();
}
```

```java
@org.junit.jupiter.api.Test
public void testCreateUser() {
    Author user = new Author();
    user.setAuthorEmail("yo@gmail.com");
    user.setAuthorPassword("1234");
    user.setAuthorName("Gage Gabaldon");

    Author savedUser = repo.save(user);

    Author existUser = entityManager.find(Author.class, savedUser.getAuthorId());

    assertThat(user.getAuthorEmail()).isEqualTo(existUser.getAuthorEmail());
}
```



The End

Live Demo