

1.3 PersonProject 初級

- パッケージの設定
- Entityの設定

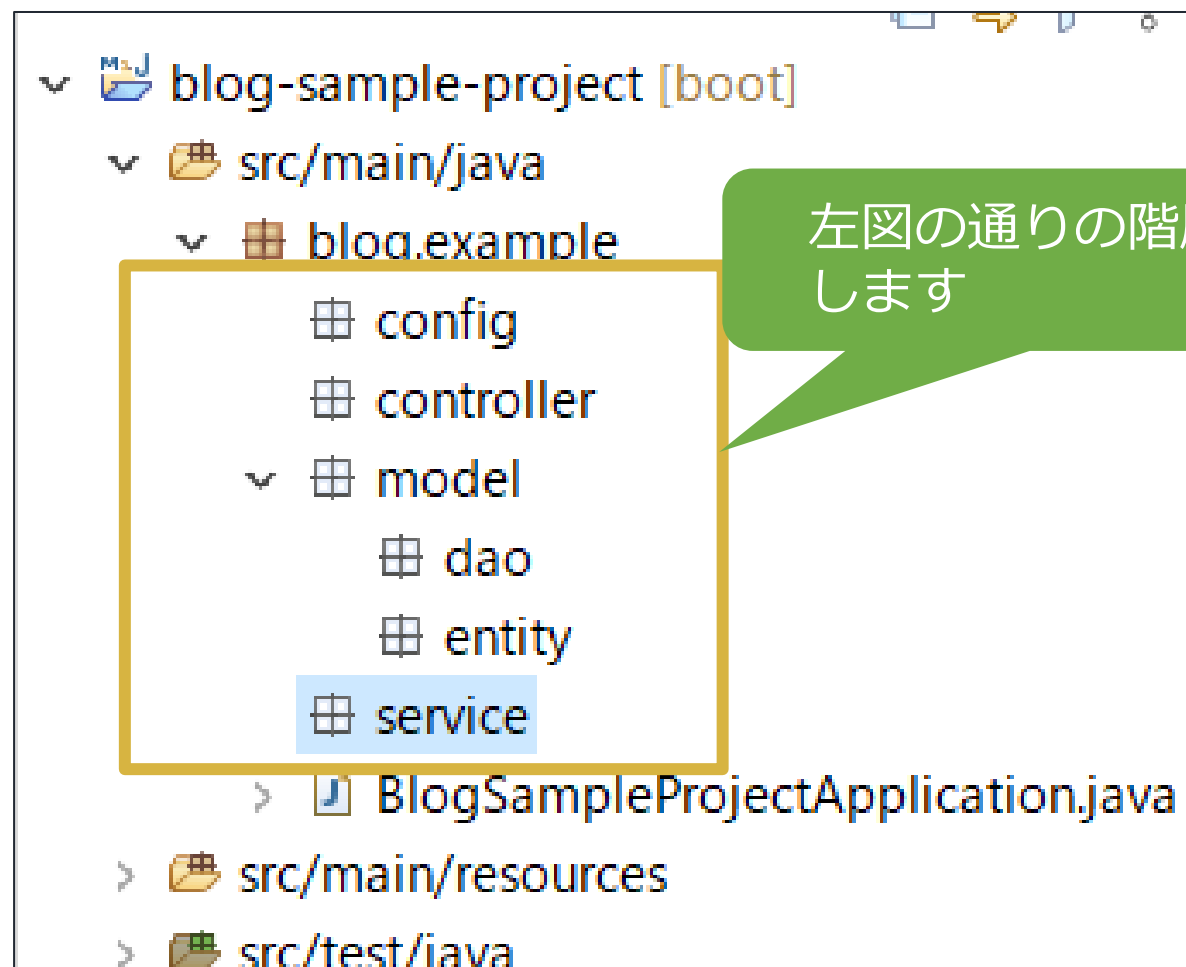
目次

1 パッケージの設定

2 Entityの生成

パッケージの設定①

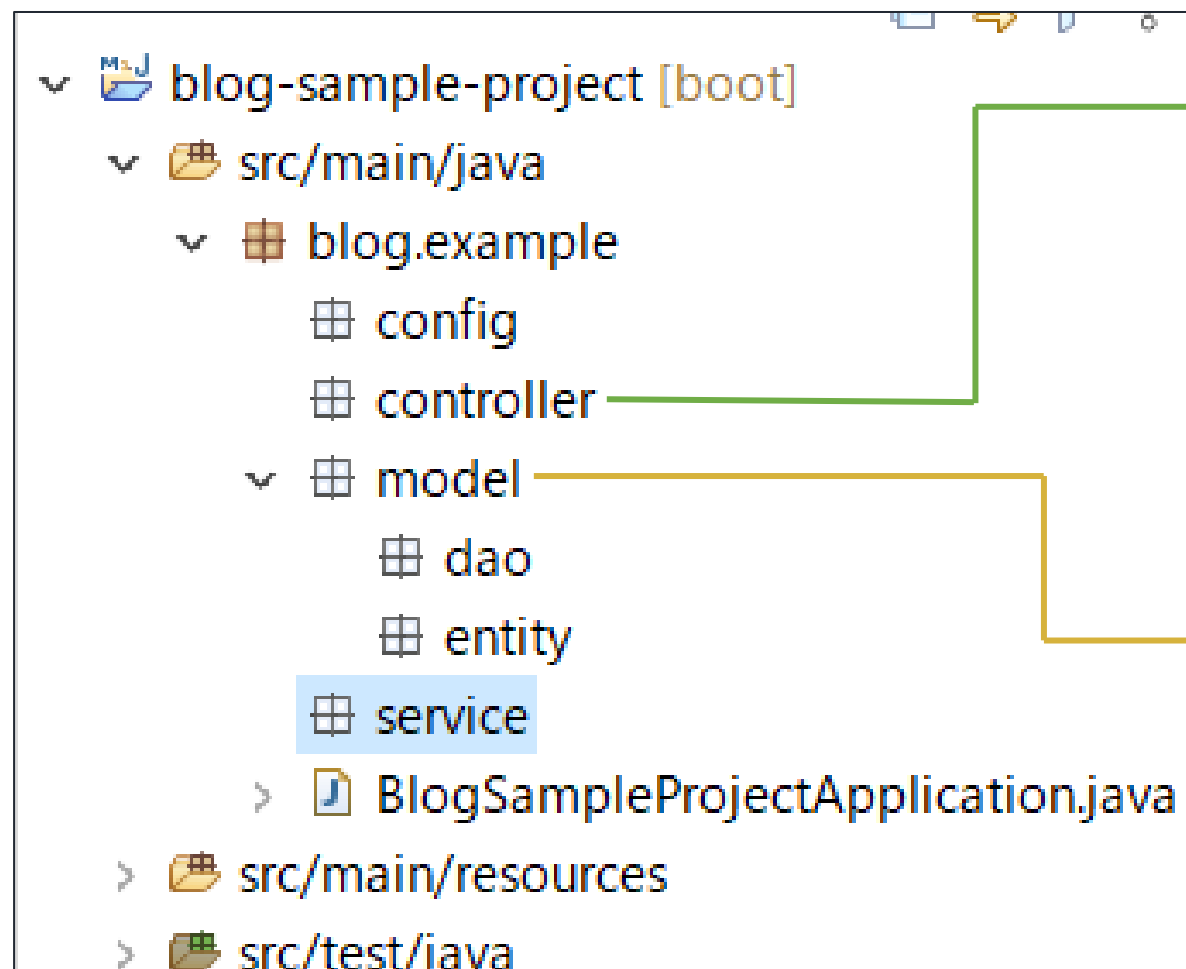
- 前回までのテキストで作成したblog-sample-projectに、パッケージ構成を作成します。



左図の通りの階層で作成します

パッケージの設定②

- パッケージ構成についてcontrollerとmodelから説明します。

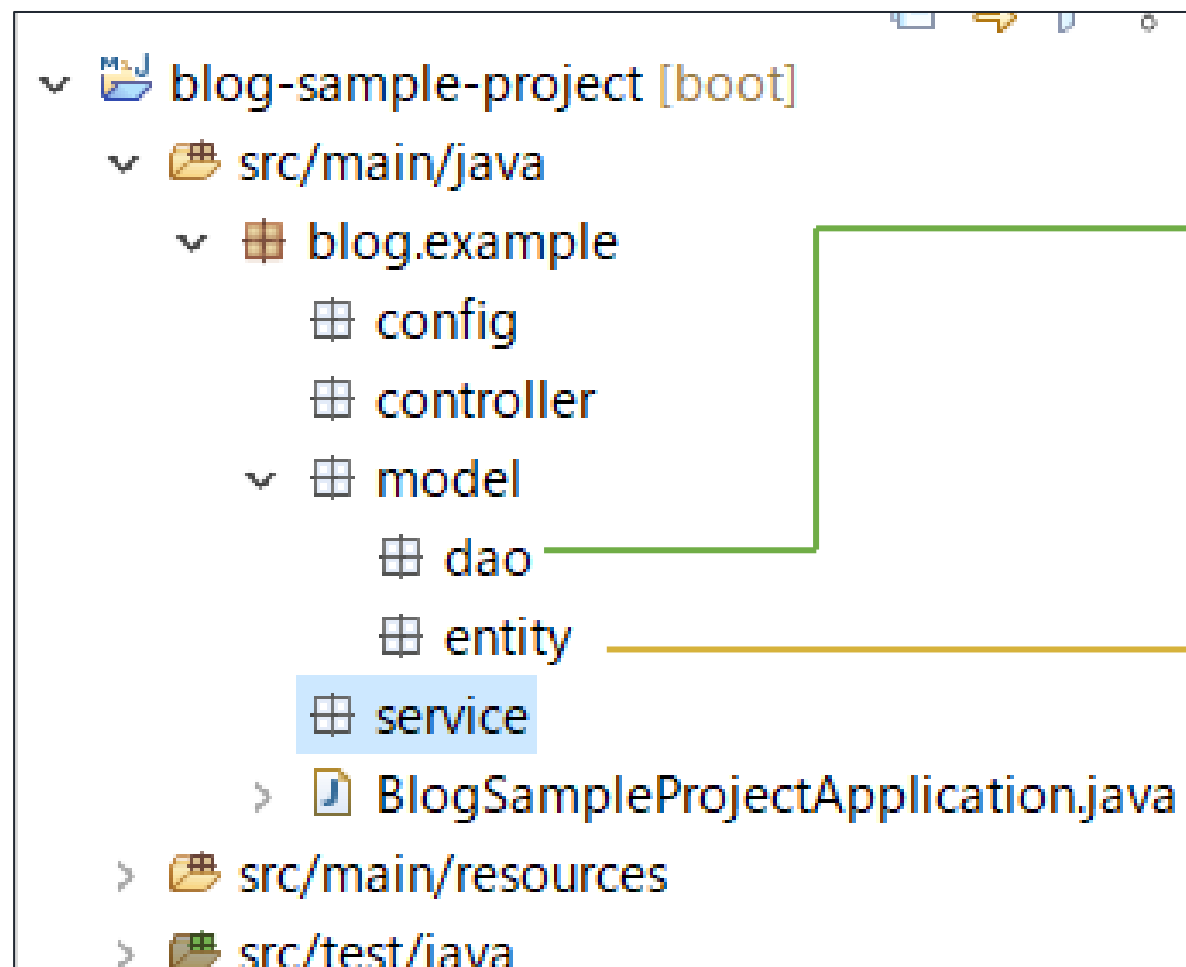


ページ遷移や、model/viewの橋渡しを担う司令塔です。

データベースとのやり取りに使用される情報や、フロントとバック間でのやり取りに使用される情報等を、役割に応じて格納するための親パッケージです。

パッケージの設定③

- model内の各パッケージを説明します。

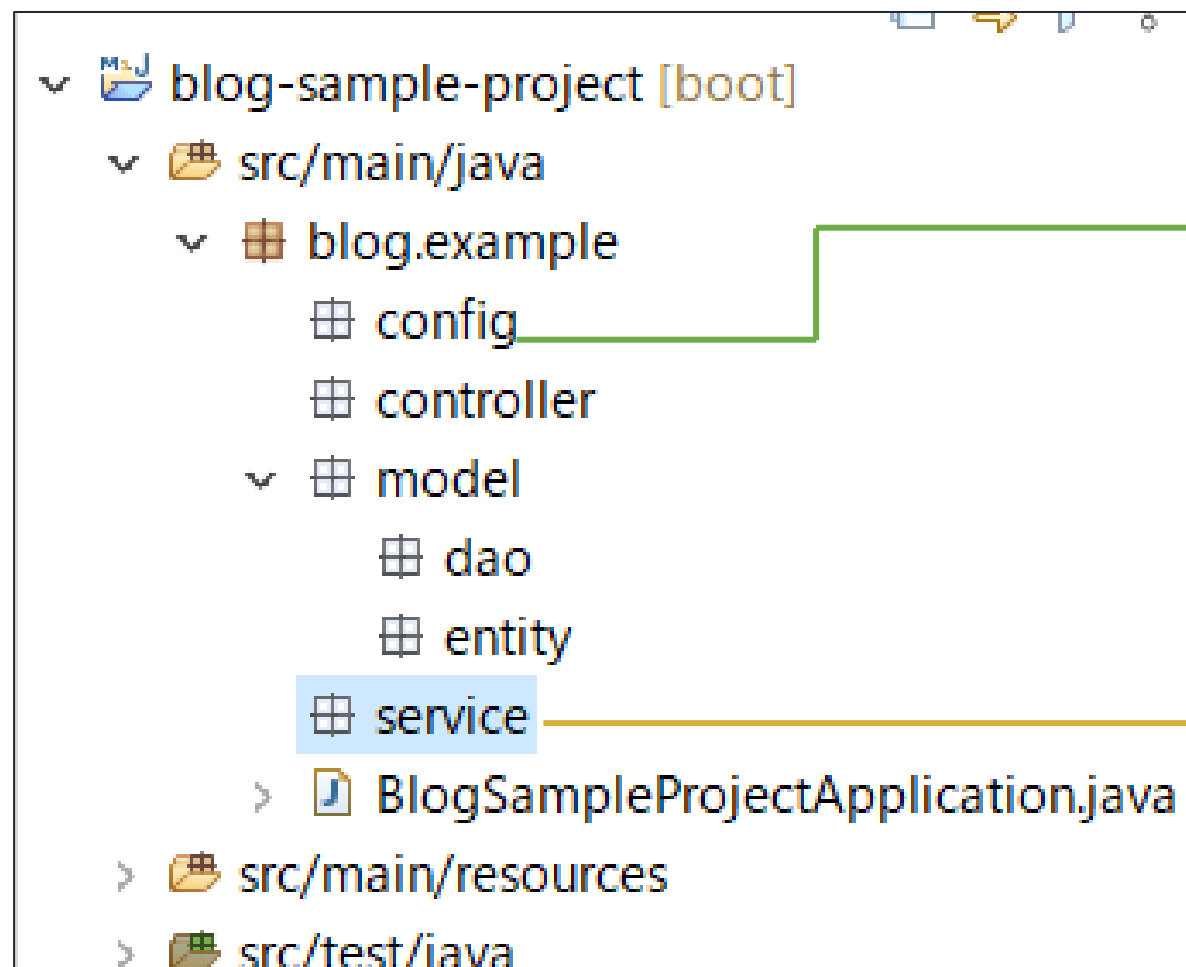


Data Access Objectの略
modelの中でも、DBにアクセスする役割をもつ、クラス・インターフェースを格納します。

Entity
DBテーブルの1レコードを合わすクラスを格納します。

パッケージの設定④

- Serviceとconfigパッケージを説明します。

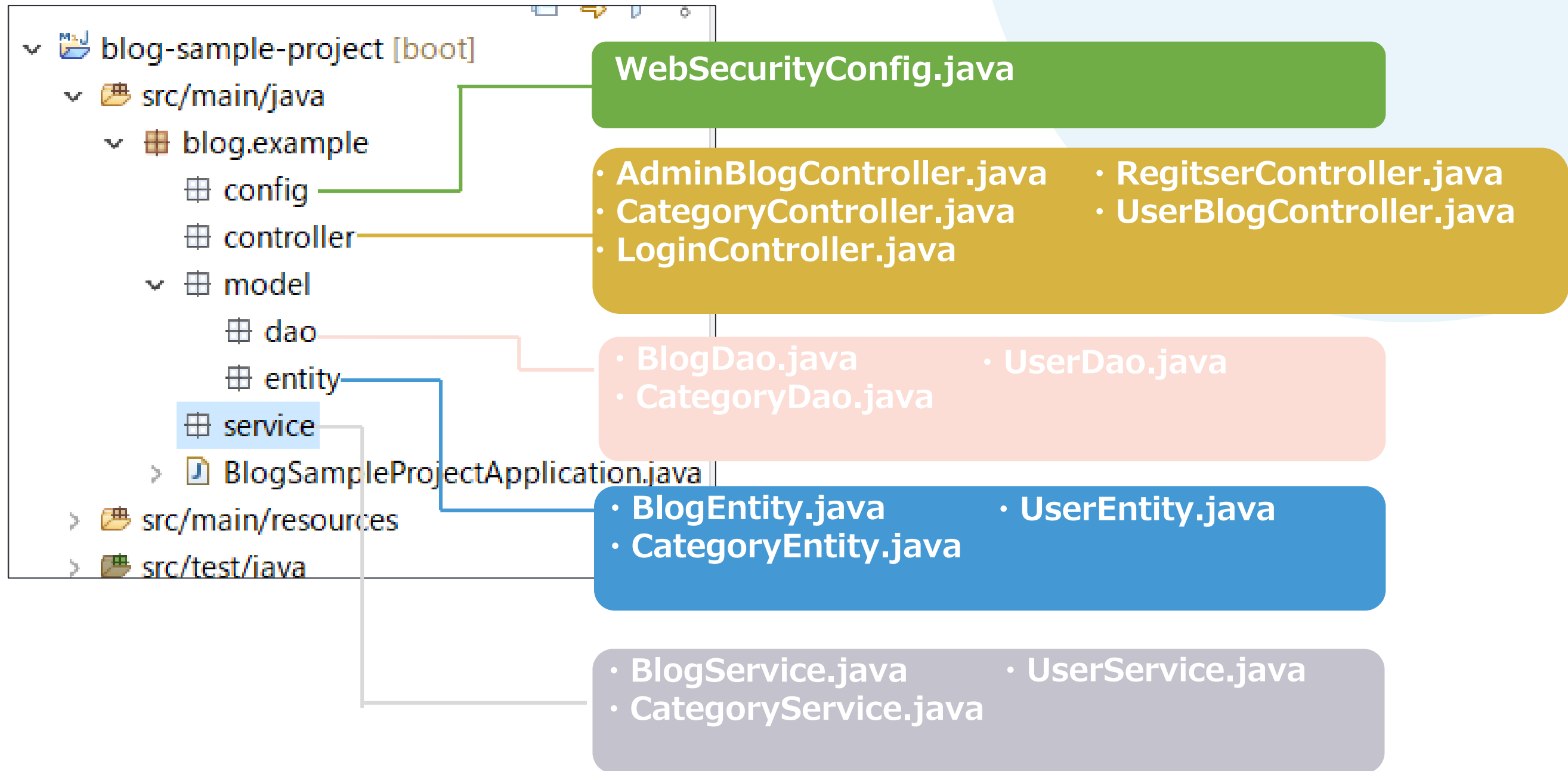


config
設定ファイル等を格納します。

DAO層のRepositoryクラスを使用してControllerが使用できる機能を実現する場所

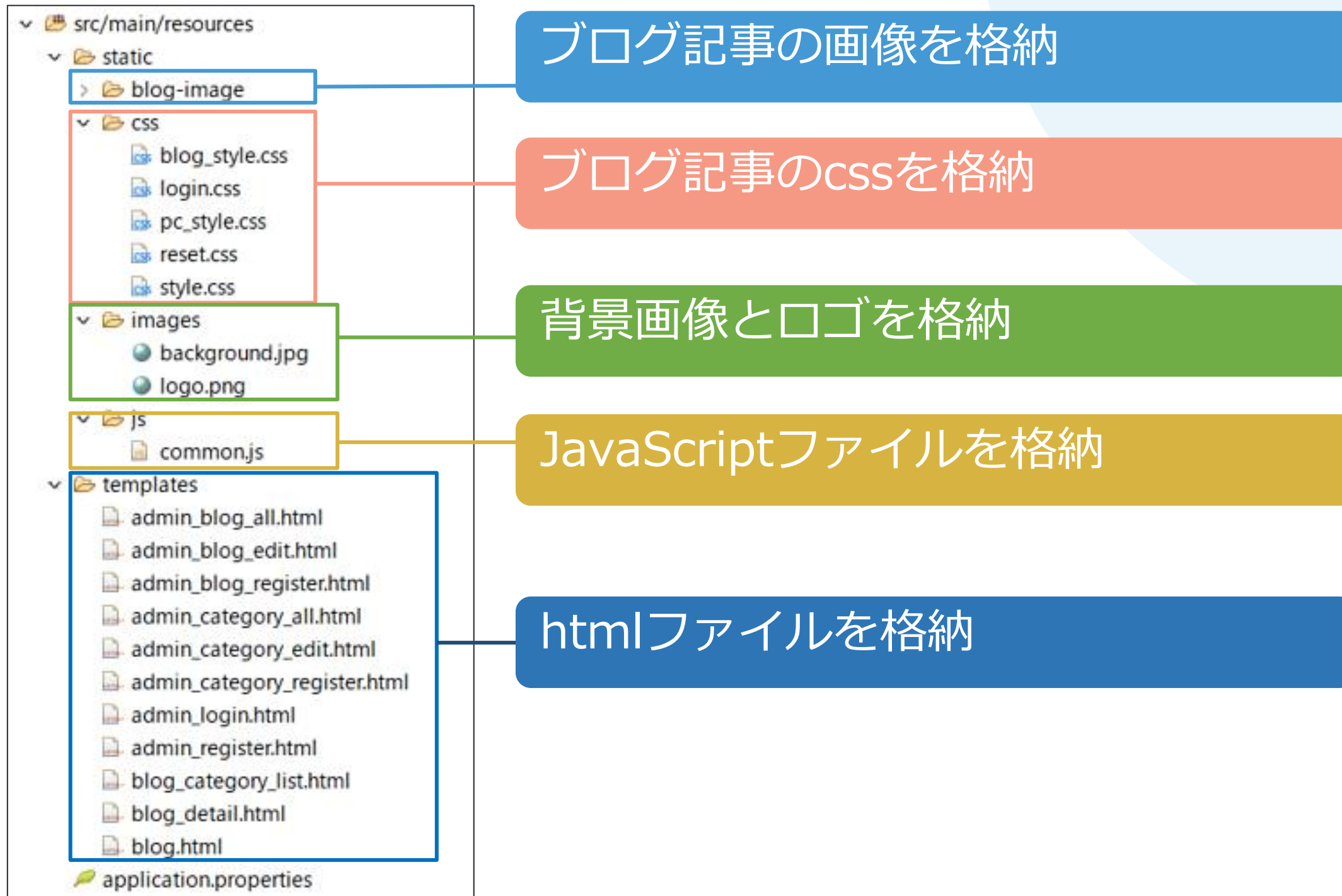
パッケージの設定⑤

- 各パッケージのどのクラスファイルが入るかを確認



パッケージの設定⑥

- 各パッケージのどのクラスファイルが入るかを確認



目次

1 パッケージの設定

2 Entityの生成

アノテーションの紹介①

```

UserEntity.java ×
1 package blog.example.model.entity;
2
3 import jakarta.persistence.Column;
4 import jakarta.persistence.Entity;
5 import jakarta.persistence.GeneratedValue;
6 import jakarta.persistence.GenerationType;
7 import jakarta.persistence.Id;
8 import jakarta.persistence.Table;
9 import lombok.Data;
10 import lombok.NoArgsConstructor;
11 import lombok.NonNull;
12 import lombok.RequiredArgsConstructor;
13
14
15 @Entity
16 @Data
17 @NoArgsConstructor
18 @RequiredArgsConstructor
19 @Table(name="account")
20 public class UserEntity {
21     @Id
22     @Column(name="user_id")
23     @GeneratedValue(strategy = GenerationType.AUTO)
24     private Long userId;
25
26     @NonNull
27     @Column(name="user_name")
28     private String userName;
29
30     @NonNull
31     @Column(name="user_email")
32     private String userEmail;
33
34     @NonNull
35     @Column(name="password")
36     private String password;
37 }
38
39

```

@Entityアノテーションを付与すると、Springの機能により 当該クラスはEntityとして振る舞うようになります。

lombokのアノテーション

アノテーション	機能
@NoArgsConstructor	パラメーターなしのコンストラクターの生成
@RequiredArgsConstructor	必須変数がパラメータなコンストラクタの生成
@AllArgsConstructor	全変数がパラメータなコンストラクタの生成
@ToString	toString() メソッドの生成
@EqualsAndHashCode	equals() および hashCode() の生成
@Data	クラスに @ToString、@EqualsAndHashCode、@RequiredArgsConstructor を追加し、全ての変数に @Getter と @Setter を追加
@Log (@Slf4j)	log という (SLF4J) ロガーの生成

@Tableアノテーションは、DBにある「どのテーブルの実体 なのか」を指定するものです。

アノテーションの紹介②

```

UserEntity.java ×
1 package blog.example.model.entity;
2
3 import jakarta.persistence.Column;
4 import jakarta.persistence.Entity;
5 import jakarta.persistence.GeneratedValue;
6 import jakarta.persistence.GenerationType;
7 import jakarta.persistence.Id;
8 import jakarta.persistence.Table;
9 import lombok.Data;
10 import lombok.NoArgsConstructor;
11 import lombok.NonNull;
12 import lombok.RequiredArgsConstructor;
13
14
15 @Entity
16 @Data
17 @NoArgsConstructor
18 @RequiredArgsConstructor
19 @Table(name="account")
20 public class UserEntity {
21     @Id
22     @Column(name="user_id")
23     @GeneratedValue(strategy = GenerationType.AUTO)
24     private Long userId;
25
26     @NonNull
27     @Column(name="user_name")
28     private String userName;
29
30     @NonNull
31     @Column(name="user_email")
32     private String userEmail;
33
34     @NonNull
35     @Column(name="password")
36     private String password;
37 }
38
39

```

DBテーブル (account)にある各カラムを、フィールドとして宣言しています。

@Id→プライマリーキーであることを指定します。

@Column→テーブルのどのカラムとマッピングするかを指定します。

@GeneratedValue→IDフィールドの振る舞いを指定します。
IDが自動追加されるように設定しています。

@NotNull→Nullでないことを検証します。

Entityクラスの作成 (UserEntity) ①

```

UserEntity.java ×
1 package blog.example.model.entity;
2
3 import jakarta.persistence.Column;
4 import jakarta.persistence.Entity;
5 import jakarta.persistence.GeneratedValue;
6 import jakarta.persistence.GenerationType;
7 import jakarta.persistence.Id;
8 import jakarta.persistence.Table;
9 import lombok.Data;
10 import lombok.NoArgsConstructor;
11 import lombok.NonNull;
12 import lombok.RequiredArgsConstructor;
13
14
15 @Entity
16 @Data
17 @NoArgsConstructor
18 @RequiredArgsConstructor
19 @Table(name="account")
20 public class UserEntity {
21     @Id
22     @Column(name="user_id")
23     @GeneratedValue(strategy = GenerationType.AUTO)
24     private Long userId;
25
26     @NonNull
27     @Column(name="user_name")
28     private String userName;
29
30     @NonNull
31     @Column(name="user_email")
32     private String userEmail;
33
34     @NonNull
35     @Column(name="password")
36     private String password;
37 }
    
```

UserEntityクラスを作成し、ソースを書き写してください。

blog-# ¥d account

列	タイプ	照合順序	Null 値を許容
user_id	bigint		not null
user_name	character varying(255)		
user_email	character varying(255)		
password	character varying(255)		

インデックス:
"account_pkey" PRIMARY KEY, btree (user_id)

テーブルとマッピングしている

Entityクラスの作成 (BlogEntity) ②

BlogEntityクラスを作成し、ソースを書き写してください。

```
package blog.example.model.entity;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Table;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.NonNull;
import lombok.RequiredArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor
@RequiredArgsConstructor
@Entity
@Table(name="blog")
public class BlogEntity {

    @Id
    @Column(name="blog_id")
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long blogId;

    @NonNull
    @Column(name="blog_title")
    private String blogTitle;

    @NonNull
    @Column(name="blog_image")
    private String blogImage;

    @NonNull
    @Column(name="category_Name")
    private String categoryName;
```

blog-# %d blog

列	タイプ	照合順序	Null 値を許容	デフォルト
blog_id	bigint		not null	
blog_image	character varying(255)			
blog_title	character varying(255)			
category_name	character varying(255)			
message	character varying(255)			
user_id	bigint			

インデックス:
"blog_pkey" PRIMARY KEY, btree (blog_id)

続き

```
    @NonNull
    @Column(name="category_name")
    private String categoryName;

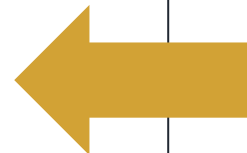
    @NonNull
    @Column(name="message")
    private String message;

    @Column(name="user_id")
    private Long userId;
}
```


Entityクラスの作成 (CategoryEntity) ③

CategoryEntityクラスを作成し、ソースを書き写してください。

```
blog-# ¥d category
      テーブル"public.category"
      列      |      タイプ      |      照合順序      |      Null 値を許容      |
      -----+-----+-----+-----+
category_id | bigint           |                    | not null                |
category_name | character varying(255) |                    |                          |
インデックス:
"category_pkey" PRIMARY KEY, btree (category_id)
```



```
CategoryEntity.java ×
1 package blog.example.model.entity;
2
3 import jakarta.persistence.Column;
4 import jakarta.persistence.Entity;
5 import jakarta.persistence.GeneratedValue;
6 import jakarta.persistence.GenerationType;
7 import jakarta.persistence.Id;
8 import jakarta.persistence.Table;
9 import lombok.AllArgsConstructor;
10 import lombok.Data;
11 import lombok.NoArgsConstructor;
12 import lombok.NonNull;
13 import lombok.RequiredArgsConstructor;
14
15 @Data
16 @NoArgsConstructor
17 @AllArgsConstructor
18 @RequiredArgsConstructor
19 @Entity
20 @Table(name="category")
21 public class CategoryEntity {
22     @Id
23     @Column(name="category_id")
24     @GeneratedValue(strategy = GenerationType.AUTO)
25     private Long categoryId;
26
27     @NonNull
28     @Column(name="category_name")
29     private String categoryName;
30 }
```


Entity作成完了時の動作の確認

- 以上で、Entity作成は完了です。この段階で必ず、SpringBootプロジェクトを起動し、エラーなどがないかを確認してください。

①プロジェクトを右クリック

②RunAsを選択

③SpringBootApplicationをクリック

④エラーがないかを確認

```

# jpaの設定
spring.jpa.properties.hibernate.jdbc.lob.
spring.jpa.properties.hibernate.dialect=c
spring.jpa.hibernate.ddl-auto=update
#spring.sql.init.mode=always
#spring.sql.init.data-locations=classpath

```

```

main] b.example.BlogSampleProjectApplication : Starting
main] b.example.BlogSampleProjectApplication : No acti
main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstr
main] .s.d.r.c.RepositoryConfigurationDelegate : Finishe
main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat
main] o.apache.catalina.core.StandardService : Startin
main] o.apache.catalina.core.StandardEngine : Startin
main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initial
main] w.s.c.ServletWebServerApplicationContext : Root We
main] o.hibernate.jpa.internal.util.LogHelper : HHH00002
main] org.hibernate.Version : HHH00004
main] org.hibernate.orm.deprecation : HHH90000
main] com.zaxxer.hikari.HikariDataSource : HikariP
main] com.zaxxer.hikari.pool.HikariPool : HikariP
main] com.zaxxer.hikari.HikariDataSource : HikariP
main] SQL dialect : HHH00004
main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH00004
main] j.LocalContainerEntityManagerFactoryBean : Initial
main] JpaBaseConfiguration$JpaWebConfiguration : spring.
main] ion$DefaultTemplateResolverConfiguration : Cannot
main] .s.s.UserDetailsServiceAutoConfiguration :

Using generated security password: 13a800c1-0c0b-4da0-a501-534369ab3fec

This generated password is for development use only. Your security configuration must be updated before running yo

2022-12-12T16:30:43.191+09:00 INFO 19296 --- [main] o.s.s.web.DefaultSecurityFilterChain : Will se
2022-12-12T16:30:43.282+09:00 INFO 19296 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat
2022-12-12T16:30:43.295+09:00 INFO 19296 --- [main] b.example.BlogSampleProjectApplication : Started

```



Light in Your Career.

THANK YOU!