LightHouse Academy

1.3 PersonProject　初級

- パッケージの設定
- Entityの設定

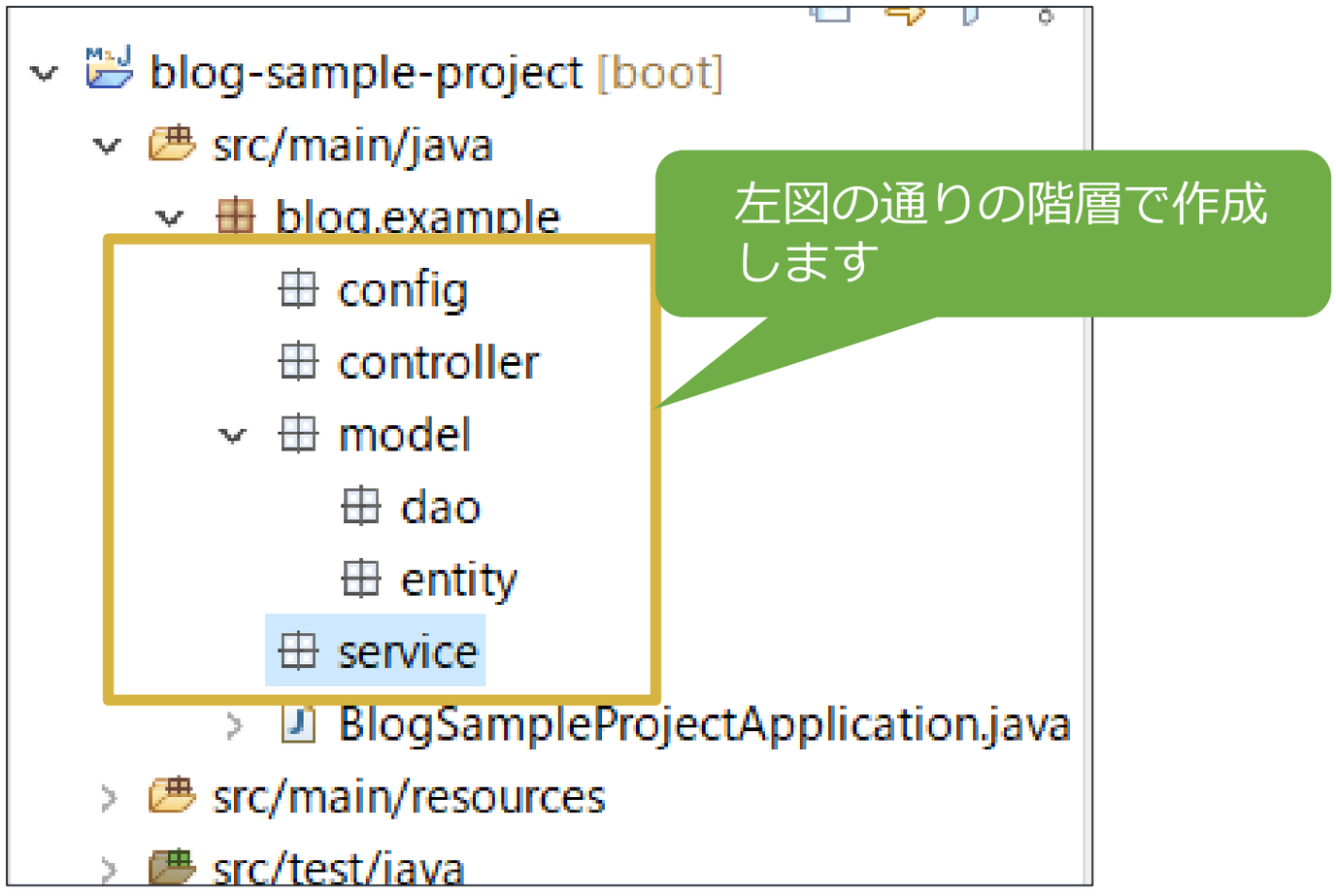
www.lighthouseit.academy

目次

© Suporich Co.Ltd.

パッケージの設定①

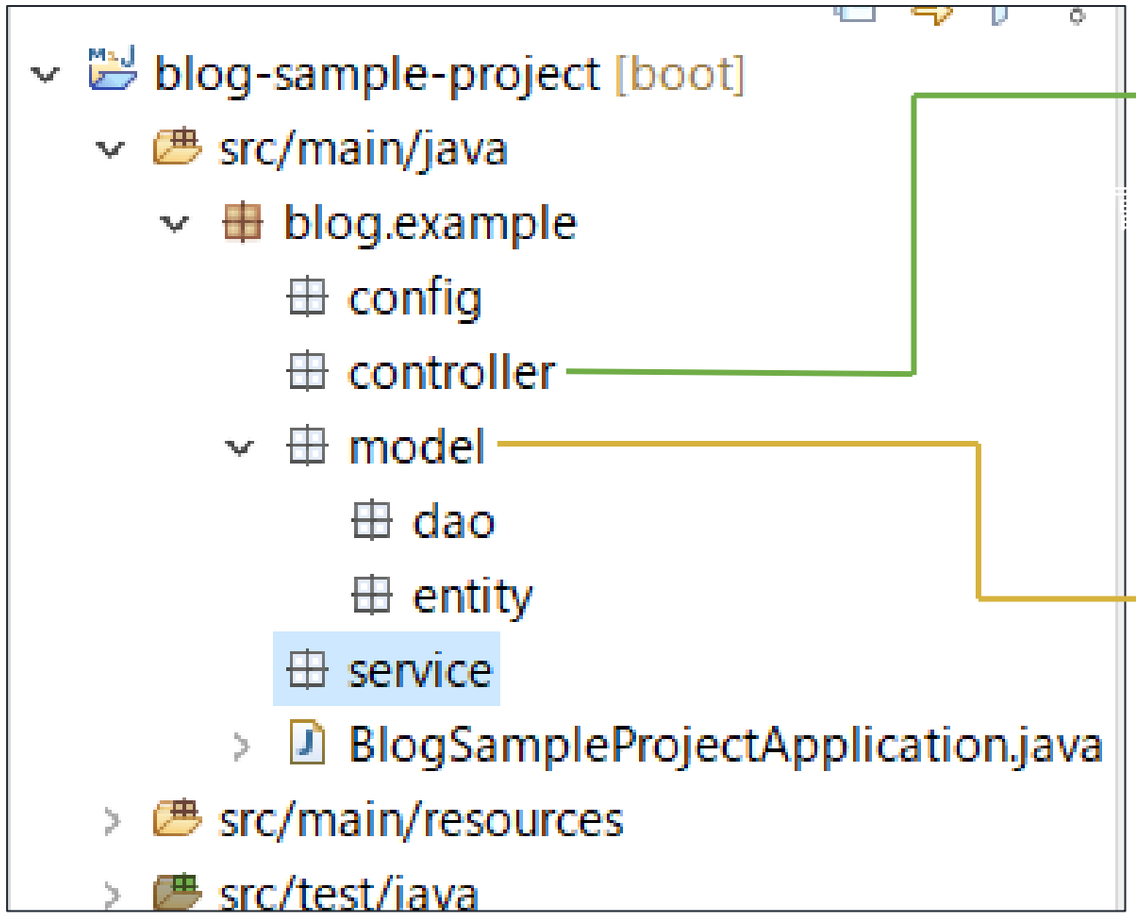
- 前回までのテキストで作成したblog-sample-projectに、パッケージ構成を作成します。



© Suporich Co.Ltd.

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。



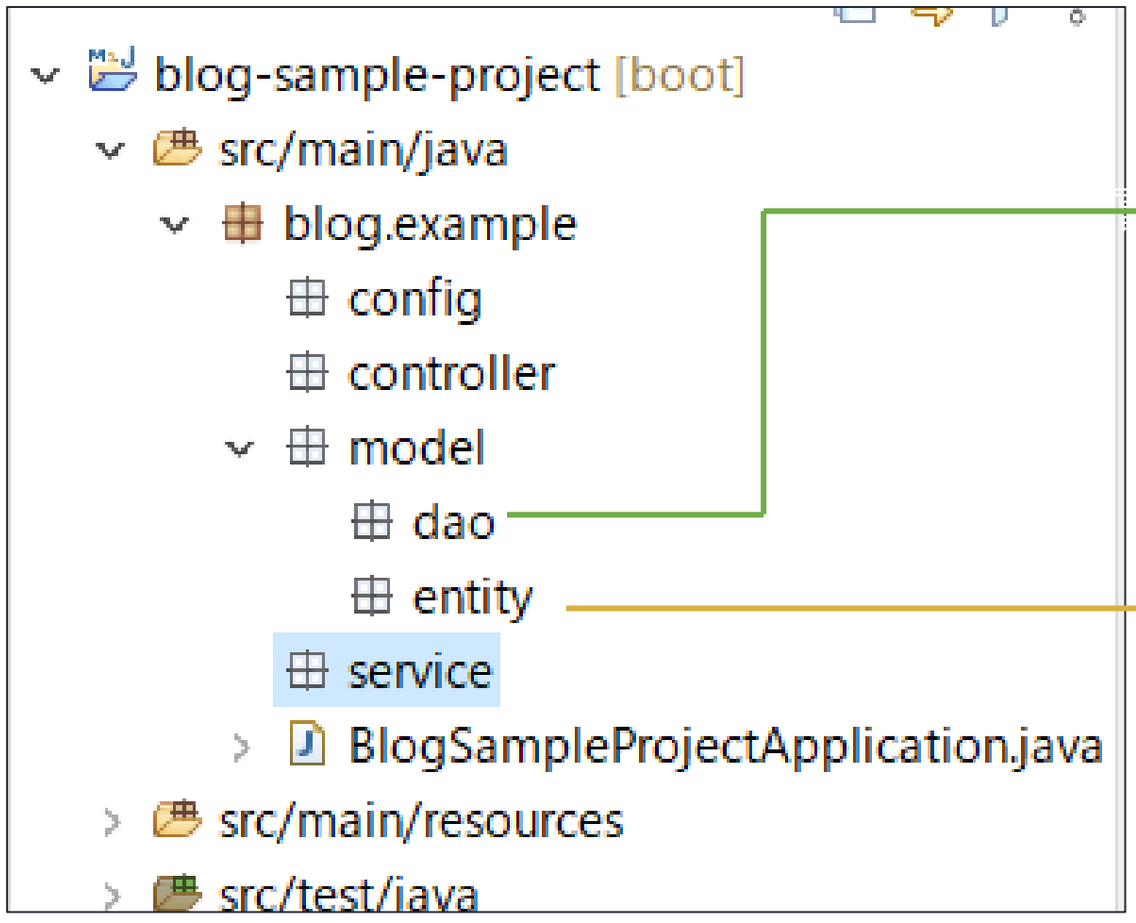
ページ遷移や、model/viewの橋渡しを担う司令塔です。

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。

© Suporich Co.Ltd.

パッケージの設定③

- model内の各パッケージを説明します。



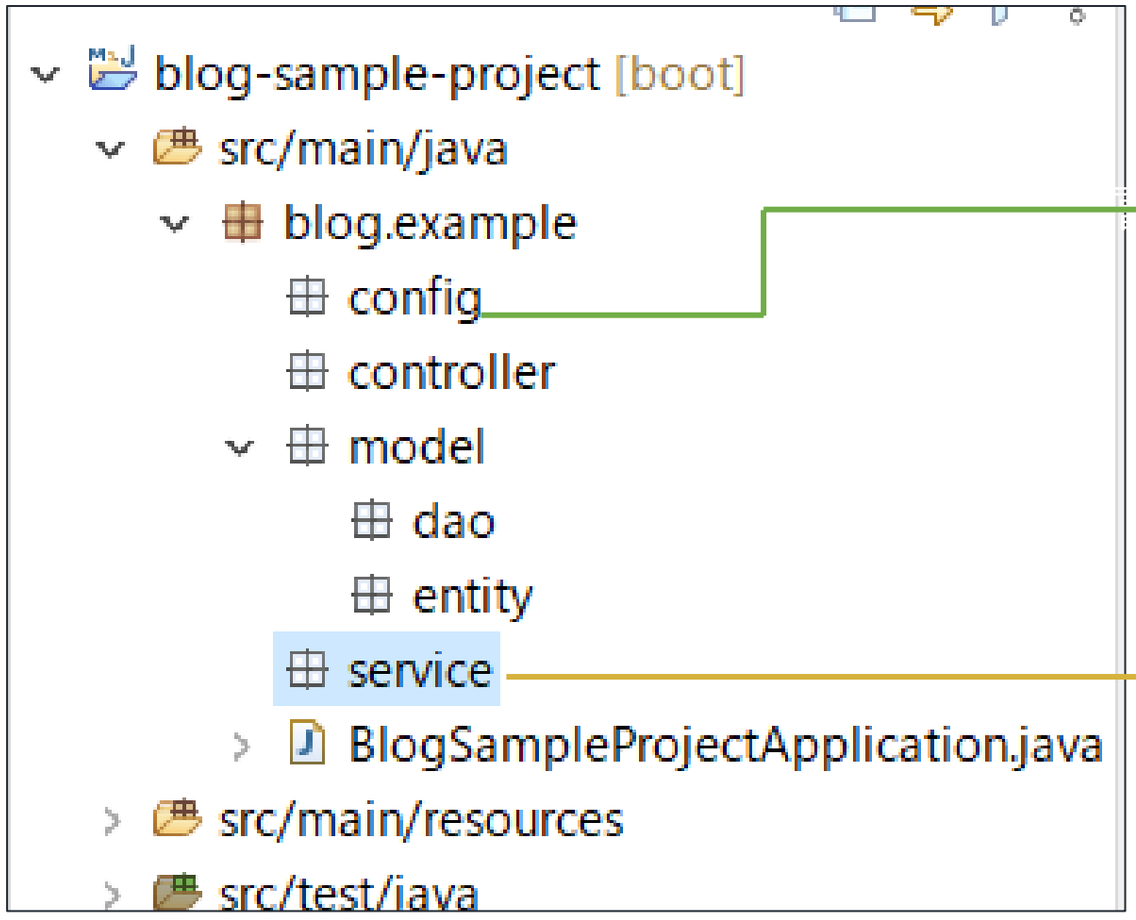
Data Accecc Objectの略
modelの中でも、DBにアクセスする役割をもつ、クラス・インターフェースを格納します。

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。

© Suporich Co.Ltd.

パッケージの設定④

- Serviceとconfigパッケージを説明します。



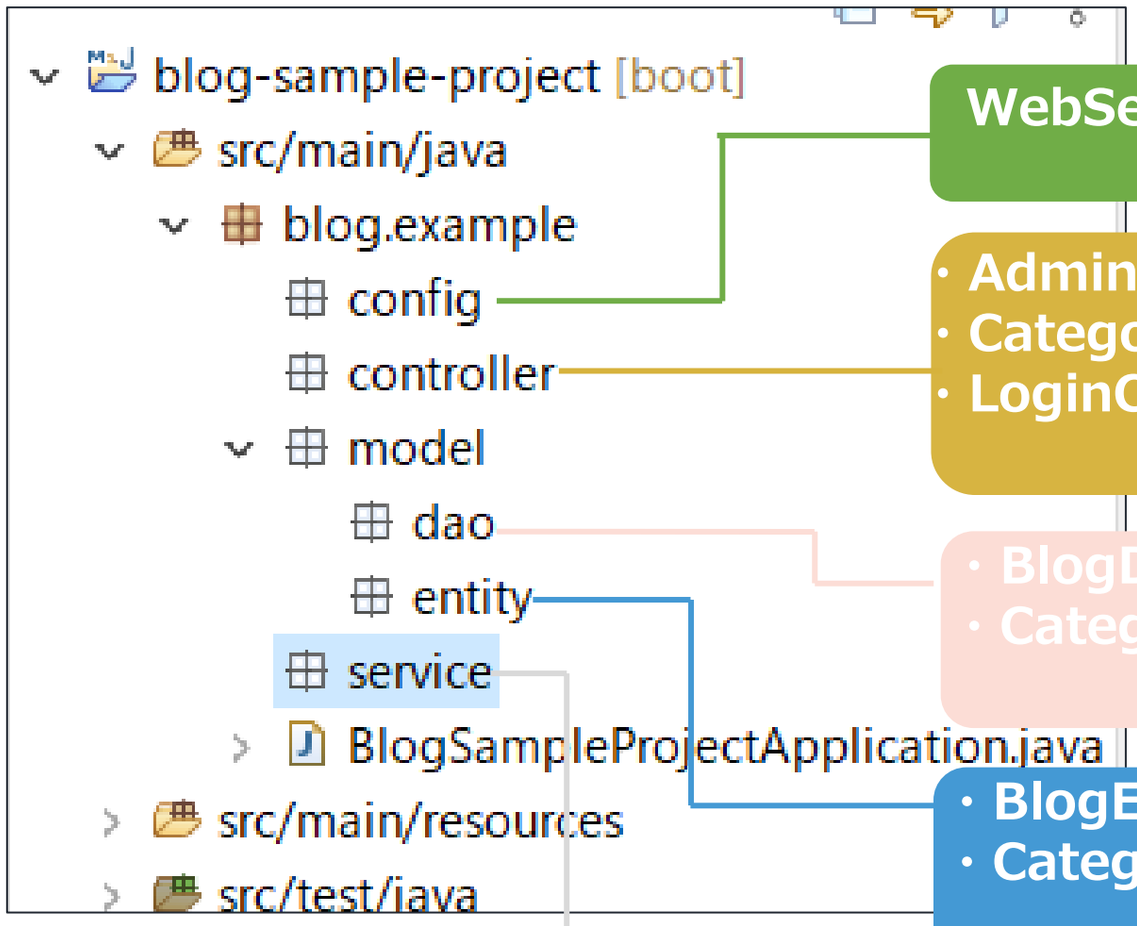
config
設定ファイル等を格納します。

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

© Suporich Co.Ltd.

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



WebSecurityConfig.java

- AdminBlogController.java
- CategoryController.java
- LoginController.java
- RegitserController.java
- UserBlogController.java

- BlogDao.java
- CategoryDao.java
- UserDao.java

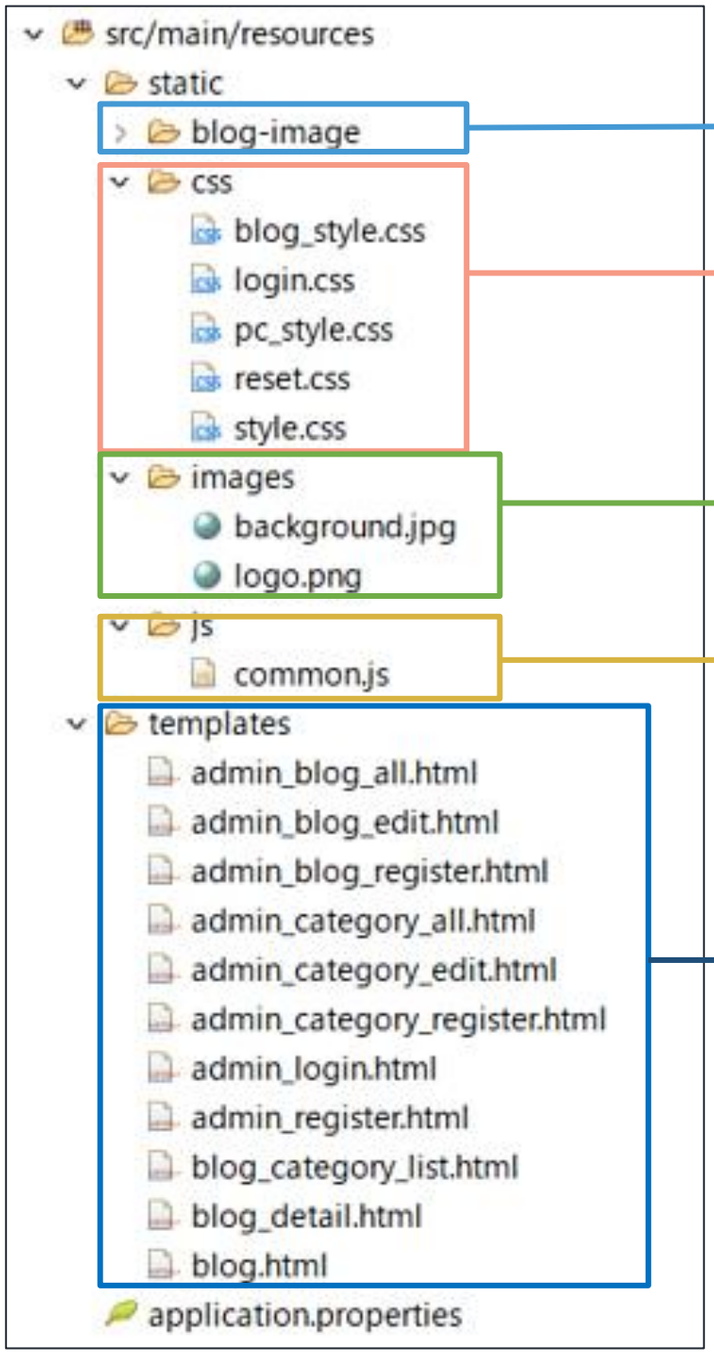
- BlogEntity.java
- CategoryEntity.java
- UserEntity.java

- BlogService.java
- CategoryService.java
- UserService.java

© Suporich Co.Ltd.

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



ブログ記事の画像を格納

ブログ記事のcssを格納

背景画像とロゴを格納

JavaScriptファイルを格納

htmlファイルを格納

© Suporich Co.Ltd.

目次

© Suporich Co.Ltd.

アノテーションの紹介①

UserEntity.java

```java
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.Data;
import lombok.NoArgsConstructor;
import lombok.NonNull;
import lombok.RequiredArgsConstructor;


@Entity
@Data
@NoArgsConstructor
@RequiredArgsConstructor
@Table(name="account")
public class UserEntity {
    @Id
    @Column(name="user_id")
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long userId;

    @NonNull
    @Column(name="user_name")
    private String userName;

    @NonNull
    @Column(name="user_email")
    private String userEmail;

    @NonNull
    @Column(name="password")
    private String password;
}
```

@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。

lombook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。

© Suporich Co.Ltd.

アノテーションの紹介②

UserEntity.java

```java
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.Data;
import lombok.NoArgsConstructor;
import lombok.NonNull;
import lombok.RequiredArgsConstructor;


@Entity
@Data
@NoArgsConstructor
@RequiredArgsConstructor
@Table(name="account")
public class UserEntity {
    @Id
    @Column(name="user_id")
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long userId;

    @NonNull
    @Column(name="user_name")
    private String userName;

    @NonNull
    @Column(name="user_email")
    private String userEmail;

    @NonNull
    @Column(name="password")
    private String password;
}
```

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。

© Suporich Co.Ltd.

Entityクラスの作成（UserEntity）①

UserEntity.java

```java
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.Data;
import lombok.NoArgsConstructor;
import lombok.NonNull;
import lombok.RequiredArgsConstructor;


@Entity
@Data
@NoArgsConstructor
@RequiredArgsConstructor
@Table(name="account")
public class UserEntity {
    @Id
    @Column(name="user_id")
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long userId;

    @NonNull
    @Column(name="user_name")
    private String userName;

    @NonNull
    @Column(name="user_email")
    private String userEmail;

    @NonNull
    @Column(name="password")
    private String password;
}
```

UserEntityクラスを作成し、ソースを書き写してください。

```
blog-# ¥d account
                テーブル"public.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

© Suporich Co.Ltd.

Entityクラスの作成（BlogEntity）②

```java
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;
```

BlogEntityクラスを作成し、
ソースを書き写してください。

```
blog-# ¥d blog
                テーブル"public.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

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

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

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


© Suporich Co.Ltd.

Entityクラスの作成（CategroyEntity）③

CategoryEntity.java

```java
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="category")
public class CategoryEntity {
    @Id
    @Column(name="category_id")
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long categoryId;

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

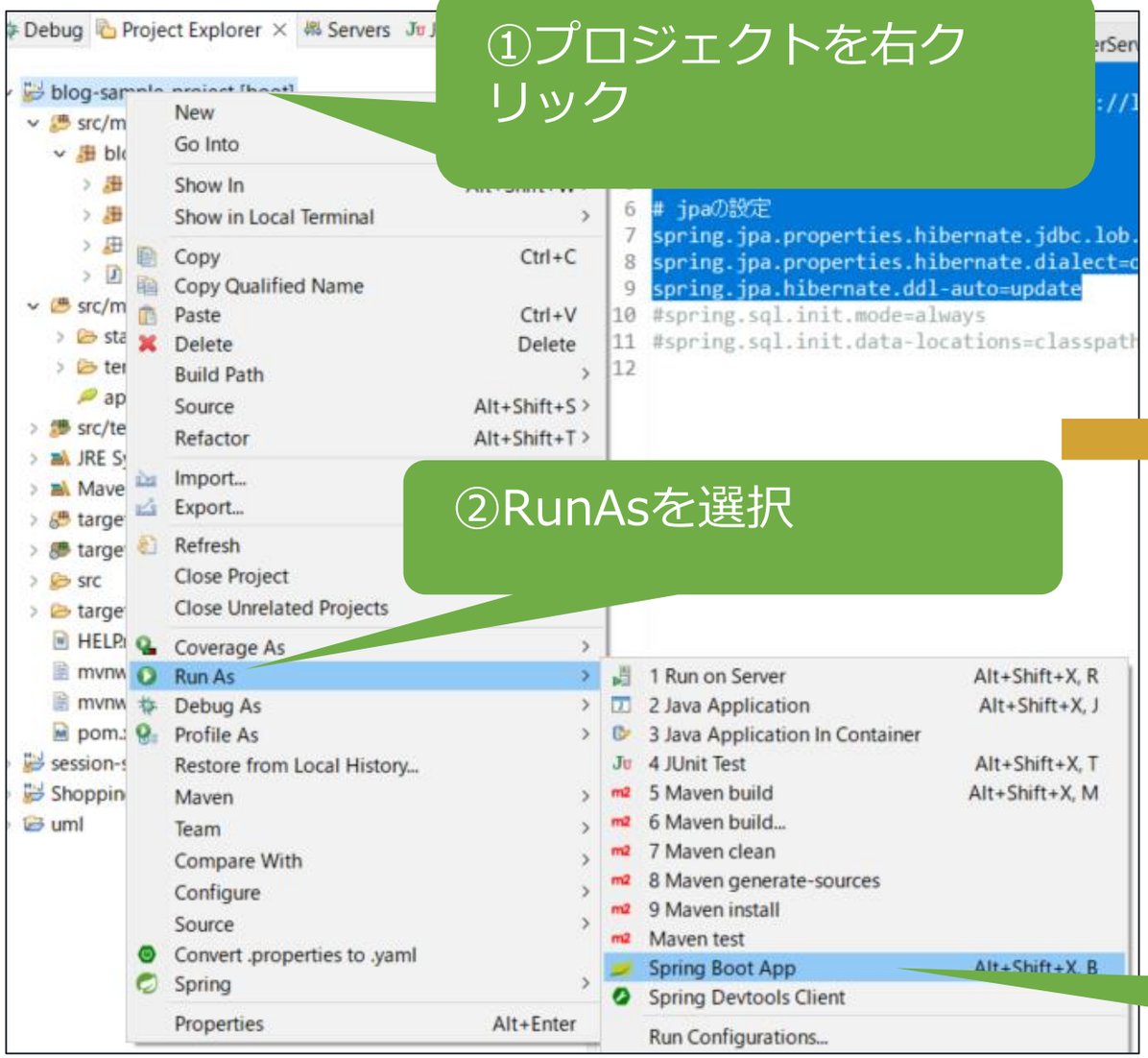
CategoryEntityクラスを作成し、
ソースを書き写してください。

```
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)
```


© Suporich Co.Ltd.

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。



© Suporich Co.Ltd.

Light in Your Career.

THANK YOU!

© Suporich Co.Ltd.