データベースの処理 - 条件参照

クエリビルダー

Laravelクエリビルダーは、PDOパラメーターバインディングを使用して、SQLインジェクション攻撃からアプリケーションを保護してくれるため、クエリビルダーに渡たす文字列をクリーンアップやサニタイズする必要はありません。

Laravelのデータベースクエリビルダーは、データベースクエリを作成・実行するための便利なインターフェイスを提供してくれます。

Laravelがサポートするすべてのデータベースシステムで完全に機能します。

テーブルのIDを使った条件抽出

resourceオプションでCRUDに対応したメソッドが用意されましたが、「 **show** 」メソッドが単体検索用のメソッドとなります。

showメソッドはテーブルの主キーを指定する引数（**$id**）を持ちます。

```
/**
* Display the specified resource.
*
* @param  int  $id
* @return \Illuminate\Http\Response
*/
public function show($id)
{
    //
}
```

テーブルのカラム名がIDの場合は、Eloquentの「 **find** 」メソッドを使うことで簡単に単一のレコードを取得することが出来ます。

```
/**
* Display the specified resource.
*
* @param  int  $id
* @return \Illuminate\Http\Response
*/
public function show($id)
{
    $article = Article::find( $id );
}
```

列の名前がID以外の場合は、Eloquentの「 **where** 」メソッドでカラム名と値を指定します。

```
/**
* Display the specified resource.
*
* @param  int  $id
```

```
* @return \Illuminate\Http\Response
*/
public function show($id)
{
    $article = Article::find( $id );
    $thumbnails = Thumbnail::where( "article_id", $article->id )->get();
}
```

クエリビルダーが生成したSQLを確認したい場合は、「 toSql 」メソッドを使うことで確認することができます。

```
// select * from `articles` where `articles`.`deleted_at` is null
$article = Article::find( $id )->toSql();

// select * from `articles` where `title` = ? and `articles`.`deleted_at` is null
$articleDao->where( "title", "Sample" )->toSql();

// select * from `articles` where (`title` = ?) and `articles`.`deleted_at` is null
$articleDao->where( [ "title" => "Sample" ] )->toSql();

// select `id`, `title` from `articles` where (`title` = ?) and `articles`.`deleted_at` is null
$articleDao->select( [ "id", "title" ] )->where( [ "title" => "Sample" ] )->toSql();

// select `id`, `title` from `articles` where (`title` = ?) and `articles`.`deleted_at` is null order by `created_at` desc
$articleDao->select( [ "id", "title" ] )->where( [ "title" => "Sample" ] )->orderBy( "created_at", "desc" )->toSql();
```

課題

- kadai07_1