# ReservationController の index メソッド詳細解説

## 📝 概要

このドキュメントは、kpg-laravelシステムの ReservationController の index メソッドについて、非エンジニアでも理解できるように解説したものです。

## 🎯 このメソッドの役割

**予約一覧画面を表示する処理**

## 📋 完全なコード

<?php  
  
namespace App\Http\Controllers;  
  
use App\Models\Reservation;  
use App\Models\Calendar;  
use App\Models\Service;  
use App\Models\TmpOrderDetail;  
use App\Models\Order;  
use App\Models\OrderDetail;  
use App\Consts\ReservationConst;  
use App\Services\FreedayService;  
use Illuminate\Http\Request;  
use Auth;  
use DB;  
use Carbon\Carbon;  
  
class ReservationController extends Controller  
{  
 private $freeday\_service;  
   
 public function \_\_construct(FreedayService $freeday\_service)  
 {  
 $this->freeday\_service = $freeday\_service;  
 }  
   
 public function index(Request $request)  
 {  
 $user = Auth::user();  
  
 //2年分のFIXDAYを取得  
 $start\_date = Carbon::now()->firstOfYear();  
 $end\_date = $start\_date->copy()->addYears(2)->endOfYear();  
  
 $calendars = Calendar::where('user\_id',$user->id)  
 ->whereBetween('start\_date',[$start\_date,$end\_date])  
 ->orderBy('start\_date','asc')  
 ->get();  
  
 $freedays = $this->freeday\_service->getFreedays($user);  
  
 $reservations = Reservation::where('owner\_id',$user->id)  
 ->whereIn('status',[  
 ReservationConst::STATUS\_APPLYING,  
 ReservationConst::STATUS\_UNDER\_RESERVATION,  
 ReservationConst::STATUS\_RESERVED  
 ])->orderBy('checkin\_date','asc')->get();  
  
 return view('reservation.index',compact('calendars','freedays','reservations'));  
 }  
}

## 🔧 各部分の詳細解説

### 1. クラスとコンストラクタ

class ReservationController extends Controller  
{  
 private $freeday\_service;  
   
 public function \_\_construct(FreedayService $freeday\_service)  
 {  
 $this->freeday\_service = $freeday\_service;  
 }

#### 説明

* ReservationController というクラス（設計図）を作成
* $freeday\_service という変数を用意
* コンストラクタ（初期化処理）で FreedayService を受け取る
* クラス内で $this->freeday\_service として使えるようにする

#### わかりやすく言うと

「FREEDAY関連の便利な機能を使えるように準備する」

例えるなら：レストランで「電卓」を用意しておく → 後で会計計算に使える

### 2. ログインユーザー取得

$user = Auth::user();

#### 説明

* Auth::user() で現在ログイン中のユーザー情報を取得
* $user 変数に保存

#### わかりやすく言うと

「今、誰がログインしているか確認する」

例： - 田中太郎さんがログイン中 - → $user には田中太郎さんの情報が入る - ID - 名前 - メールアドレス - 会員種別（オーナーか一般か） - など

### 3. FIXDAY（カレンダー）取得

#### 3-1. 期間の設定

//2年分のFIXDAYを取得  
$start\_date = Carbon::now()->firstOfYear();  
$end\_date = $start\_date->copy()->addYears(2)->endOfYear();

#### 説明

**Carbon::now()** → 今日の日付を取得

**->firstOfYear()** → 今年の1月1日に変換

**$start\_date->copy()** → $start\_dateをコピー（元の値を変えないため）

**->addYears(2)** → 2年後に進める

**->endOfYear()** → その年の12月31日に設定

#### 具体例

今日が 2025年10月20日 の場合：

* $start\_date = 2025年1月1日
* $end\_date = 2027年12月31日

→ 2025年、2026年、2027年の3年分

#### わかりやすく言うと

「今年から2年後までの期間を設定する」 → この期間のFIXDAYを表示するため

#### 3-2. データベースから取得

$calendars = Calendar::where('user\_id',$user->id)  
 ->whereBetween('start\_date',[$start\_date,$end\_date])  
 ->orderBy('start\_date','asc')  
 ->get();

#### 説明（1行ずつ）

**1行目: Calendar::where('user\_id', $user->id)**

Calendarテーブルから検索を開始

条件: user\_id がログイン中のユーザーIDと一致 → 自分のFIXDAYのみ

例： - 田中さん（user\_id = 5）がログイン - → user\_id = 5 のカレンダーのみ検索

**2行目: ->whereBetween('start\_date', [$start\_date, $end\_date])**

さらに条件追加： start\_date が $start\_date と $end\_date の間

例： - 2025-01-01 〜 2027-12-31 の間 - → この範囲のFIXDAYのみ

**3行目: ->orderBy('start\_date', 'asc')**

並び替え： start\_date で昇順（古い順）

‘asc’ = ascending（昇順） → 日付が早いものから順番に

結果： 2025年8月 → 2025年12月 → 2026年8月… の順

**4行目: ->get()**

実際にデータベースから取得して結果を返す

$calendars に配列として保存される

#### わかりやすく言うと

「このユーザーの今後2年分のFIXDAYを日付が早い順に取得する」

結果イメージ：

$calendars = [  
 [id=1, start\_date='2025-08-15', end\_date='2025-08-17', ...],  
 [id=2, start\_date='2025-12-30', end\_date='2026-01-02', ...],  
 [id=3, start\_date='2026-08-15', end\_date='2026-08-17', ...],  
]

### 4. FREEDAY取得

$freedays = $this->freeday\_service->getFreedays($user);

#### 説明

FreedayService の getFreedays メソッドを実行

内部で以下を実施： 1. このユーザーのFREEDAYを検索 2. end\_date >= 今日（有効期限内）のもののみ 3. 結果を返す

#### わかりやすく言うと

「このユーザーの有効なFREEDAYを取得する」

有効 = 期限切れしていないもの

例： - 前期FREEDAY: - 残り3泊 - 有効期限: 6月30日 - 今日が5月15日 → ✅ 有効（取得される） - 今日が7月1日 → ❌ 期限切れ（取得されない）

### 5. 予約一覧取得

$reservations = Reservation::where('owner\_id',$user->id)  
 ->whereIn('status',[  
 ReservationConst::STATUS\_APPLYING,  
 ReservationConst::STATUS\_UNDER\_RESERVATION,  
 ReservationConst::STATUS\_RESERVED  
 ])->orderBy('checkin\_date','asc')->get();

#### 説明（1行ずつ）

**1行目: Reservation::where('owner\_id', $user->id)**

Reservationテーブルから検索

条件: owner\_id が このユーザー → このユーザーがオーナーの予約

注: user\_id ではなく owner\_id → オーナーとゲストの両方の予約を含む

**2-5行目: ->whereIn('status', [...])**

ステータスが以下のいずれかのもの：

* STATUS\_APPLYING (1) = 未確定
* STATUS\_UNDER\_RESERVATION (2) = 申込中
* STATUS\_RESERVED (3) = 確定済み

含まれないもの： - キャンセル済み (9) - チェックイン済み (4) - チェックアウト済み (5) など

**6行目: ->orderBy('checkin\_date', 'asc')**

チェックイン日が早い順に並べる

**7行目: ->get()**

取得して $reservations に保存

#### わかりやすく言うと

「このユーザーの今後の予約（キャンセルしていない）をチェックイン日が早い順に取得する」

結果イメージ： - 11月5日チェックイン（申込中） - 12月20日チェックイン（確定済み） - 2026年3月15日チェックイン（未確定）

### 6. ビュー（画面）に渡す

return view('reservation.index', compact('calendars', 'freedays', 'reservations'));

#### 説明

**view(‘reservation.index’, …)** → resources/views/reservation/index.blade.php を表示

**compact(‘calendars’, ‘freedays’, ‘reservations’)** → 以下の3つの変数を画面に渡す： - $calendars （FIXDAY一覧） - $freedays （FREEDAY一覧） - $reservations （予約一覧）

#### わかりやすく言うと

「集めた3つの情報を予約一覧画面に渡して表示する」

画面側で使える変数：

@foreach($calendars as $calendar)  
 → FIXDAY一覧を表示  
   
@foreach($freedays as $freeday)  
 → FREEDAY一覧を表示  
   
@foreach($reservations as $reservation)  
 → 既存予約一覧を表示

## 🎯 処理の全体フロー

【ユーザーがアクセス】  
http://localhost:8081/reservation/index  
  
↓  
  
【1】ログイン確認  
Auth::user()  
→ 田中太郎さんとわかる  
  
↓  
  
【2】FIXDAY取得  
Calendar テーブルから  
田中さんの2025-2027年のFIXDAYを取得  
→ $calendars に保存  
  
↓  
  
【3】FREEDAY取得  
Freeday テーブルから  
田中さんの有効なFREEDAYを取得  
→ $freedays に保存  
  
↓  
  
【4】予約取得  
Reservation テーブルから  
田中さんの今後の予約を取得  
→ $reservations に保存  
  
↓  
  
【5】画面表示  
reservation/index.blade.php に  
3つのデータを渡して表示

## 🖼️ 画面表示イメージ

┌────────────────────────────────────┐  
│ 予約管理画面 │  
├────────────────────────────────────┤  
│ │  
│ ■ FIXDAY（固定日予約） │  
│ ┌──────────────────────────────┐ │  
│ │ 2025/8/15〜8/17（2泊） │ │  
│ │ ステータス: 予約可 │ │  
│ │ [予約する]ボタン │ │  
│ └──────────────────────────────┘ │  
│ ┌──────────────────────────────┐ │  
│ │ 2025/12/30〜2026/1/2（3泊） │ │  
│ │ ステータス: 確定済み │ │  
│ │ [詳細を見る]ボタン │ │  
│ └──────────────────────────────┘ │  
│ ↑ $calendars で表示 │  
│ │  
│ ■ FREEDAY（自由日予約） │  
│ ┌──────────────────────────────┐ │  
│ │ 利用可能: 3泊 │ │  
│ │ 有効期限: 2025年6月末まで │ │  
│ │ [予約する]ボタン │ │  
│ └──────────────────────────────┘ │  
│ ┌──────────────────────────────┐ │  
│ │ 利用可能: 4泊 │ │  
│ │ 有効期限: 2025年12月末まで │ │  
│ │ [予約する]ボタン │ │  
│ └──────────────────────────────┘ │  
│ ↑ $freedays で表示 │  
│ │  
│ ■ 今後の予約 │  
│ ┌──────────────────────────────┐ │  
│ │ 11/5〜11/6（1泊） │ │  
│ │ ステータス: 申込中 │ │  
│ │ 大人2名 │ │  
│ └──────────────────────────────┘ │  
│ ┌──────────────────────────────┐ │  
│ │ 12/20〜12/22（2泊） │ │  
│ │ ステータス: 確定済み │ │  
│ │ 大人2名、子供1名 │ │  
│ └──────────────────────────────┘ │  
│ ↑ $reservations で表示 │  
│ │  
└────────────────────────────────────┘

## 💡 重要ポイント

### ポイント1: 期間指定

$end\_date = $start\_date->copy()->addYears(2)->endOfYear();

* 2年分のFIXDAYを表示
* 古い年のFIXDAYは表示されない
* → FIXDAYは溜まらない

### ポイント2: 有効期限チェック

$this->freeday\_service->getFreedays($user);

内部で以下をチェック：

->where('end\_date', '>=', Carbon::now())

* 期限切れのFREEDAYは取得されない
* → FREEDAYは半年で失効

### ポイント3: ステータスフィルター

->whereIn('status', [1, 2, 3])

* キャンセル済み（9）は表示しない
* 今後の予約のみ表示

## 🔍 コード内の専門用語解説

### const（定数）

const STATUS\_NOT\_ACCEPTED = 0;

**意味**: - const = constant（定数）の略 - 一度決めたら変更できない値

**使い方**：

// ❌ 数値だけだとわかりにくい  
if ($status == 0) { ... }  
  
// ✅ 定数名でわかりやすい  
if ($status == ReservationConst::STATUS\_NOT\_ACCEPTED) { ... }

**利点**： 1. コードが読みやすくなる 2. 間違いが減る 3. 値を変更する時、1箇所変えるだけで済む

**例え**：

定数 = ニックネーム

* 実際の値: 0, 1, 2, 3（わかりにくい）
* 定数名: STATUS\_RESERVED（わかりやすい）

「3」と書くより「確定済み」とわかる名前で書く

### Carbon（日付操作ライブラリ）

Carbon::now()->firstOfYear()

**よく使う操作**：

| コード | 意味 | 例 |
| --- | --- | --- |
| Carbon::now() | 今日 | 2025-10-20 |
| ->firstOfYear() | 今年の1月1日 | 2025-01-01 |
| ->endOfYear() | 今年の12月31日 | 2025-12-31 |
| ->addYears(2) | 2年後 | 2027-10-20 |
| ->copy() | コピー | 元の値を変えない |

### Eloquent ORM（データベース操作）

Calendar::where('user\_id', $user->id)->get();

**翻訳すると**：

SELECT \* FROM calendars WHERE user\_id = 5;

**主なメソッド**：

| メソッド | 意味 | SQL |
| --- | --- | --- |
| where('列名', 値) | 条件指定 | WHERE 列名 = 値 |
| whereBetween('列名', [開始, 終了]) | 範囲指定 | WHERE 列名 BETWEEN 開始 AND 終了 |
| whereIn('列名', [値1, 値2]) | 複数条件 | WHERE 列名 IN (値1, 値2) |
| orderBy('列名', 'asc') | 並び替え | ORDER BY 列名 ASC |
| get() | 取得実行 | 実際にデータを取得 |

## 📊 データの流れ

┌──────────────┐  
│ ユーザー │ ログイン  
│ (田中太郎) │  
└───────┬──────┘  
 │  
 ↓  
┌──────────────────────┐  
│ ReservationController │  
│ index メソッド実行 │  
└───────┬──────────────┘  
 │  
 ├→ Auth::user()   
 │ → 田中太郎さんと特定  
 │  
 ├→ Calendar::where(...)->get()  
 │ → FIXDAYを取得  
 │ → $calendars に保存  
 │  
 ├→ $freeday\_service->getFreedays()  
 │ → FREEDAYを取得  
 │ → $freedays に保存  
 │  
 ├→ Reservation::where(...)->get()  
 │ → 予約を取得  
 │ → $reservations に保存  
 │  
 ↓  
┌──────────────────────┐  
│ ビュー（画面） │  
│ reservation/index │  
└───────┬──────────────┘  
 │  
 ↓  
 ブラウザに表示

## 🎓 まとめ

### このメソッドが行うこと

1. **ログイン中のユーザーを特定**
2. **FIXDAY（2年分）を取得**
3. **FREEDAY（有効なもの）を取得**
4. **今後の予約を取得**
5. **3つのデータを画面に渡して表示**

### 画面で表示される情報

* 自分のFIXDAY一覧（予約可能な固定日程）
* 自分のFREEDAY一覧（使える宿泊ポイント）
* 自分の今後の予約一覧（既に予約済みのもの）

### ユーザーができること

* FIXDAYから予約する
* FREEDAYから予約する
* 既存の予約を確認する

このメソッドは、**予約システムの入口となる重要な画面**を作る処理です！