

**SESSION
06**

IMPLEMENTING CRUD OPERATIONS



OBJECTIVES

- Explain different aspects of MVC design patterns
- Understand user inputs and its validation
- Implement Session, Request, and Response
- Understand concept of Paginations
- Explain the importance of Authentication



INTRODUCTION

Controllers

- Move requests and responses from the routes file to their own separate files and functions
- Organizes Web applications
- Easy to implement Create, Read, Update, and Delete (CRUD) operations
- Can be initiated using: Single action controllers and resource controllers

FROM ROUTES TO CONTROLLERS

```
php artisan make:controller EmployeeController
```

```

p<?php
namespace HRM\Http\Controllers;
use Illuminate\Http\Request;
class EmployeeController extends Controller
{
    //
}

```

```
rm app/Http/Controllers/EmployeeController
```

WORKING WITH RESOURCE CONTROLLER (1-6)

```
php artisan make:controller  
EmployeeController
```

```
<?php  
namespace HRM\Http\Controllers;  
use Illuminate\Http\Request;  
class EmployeeController extends Controller  
{  
    /**  
     * Display a listing of the resource.  
     *  
     * @return \Illuminate\Http\Response  
     */  
    public function index()  
    {  
        //  
    }  
    /**  
     * Show the form for creating a new resource.  
     *  
     * @return \Illuminate\Http\Response  
     */  
    public function create()  
    {  
        //  
    }  
}
```

WORKING WITH RESOURCE CONTROLLER (2-6)

```
/**
 * Store a newly created resource in storage.
 *
 * @param \Illuminate\Http\Request $request
 * @return \Illuminate\Http\Response
 */
public function store(Request $request)
{
    //
}

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

/**
 * Show the form for editing the specified resource.
 *
 * @param int $id
 * @return \Illuminate\Http\Response
 */
public function edit($id)
{
    //
}
```

```
/**
 * Update the specified resource in storage.
 *
 * @param \Illuminate\Http\Request $request
 * @param int $id
 * @return \Illuminate\Http\Response
 */
public function update(Request $request, $id)
{
    //
}

/**
 * Remove the specified resource from storage.
 *
 * @param int $id
 * @return \Illuminate\Http\Response
 */
public function destroy($id)
{
    //
}
```

WORKING WITH RESOURCE CONTROLLER (3-6)

```
Route::resource('employee','EmployeeController');
```

```
Route::get('/', function () {  
    return redirect('employee');  
});
```

```
<?php  
/*  
-----  
/ Web Routes  
-----  
/  
/ Here is where you can register Web routes for your application. These  
/ routes are loaded by the RouteServiceProvider within a group which  
/ contains the "Web" middleware group. Now create something great!  
/  
*/  
Route::get('/', function () {  
    return redirect('employee');  
});  
Route::resource('employee','EmployeeController');
```

WORKING WITH RESOURCE CONTROLLER (4-6)

Route	Method	Function	URI	Purpose
.create	GET	create	/create	To display the form to create a record
.destroy	DELETE	destroy	//{id}	To destroy a particular record
.edit	GET	edit	{id}/edit	To display a form to edit records
.index	GET	index	/	To display the index page
.show	GET	show	//{id}	To show a particular record
.store	POST	store	/	To add a record with details from form
.update	PUT/PATCH	update	//{id}	To update a particular record

WORKING WITH RESOURCE CONTROLLER (5-6)

Index() function:

```
<body>
  <div class="container">
    <ul class="nav">
      <li class="nav-item">
        <a class="nav-link" href="{{ route('employee.index') }}">Home</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="{{ route('employee.create') }}">Create</a>
      </li>
    </ul>
    @yield('message')
    <h1>@yield('heading')</h1>

    <div>
      @yield('content')
    </div>
  </div>
</body>
</html>
```

WORKING WITH RESOURCE CONTROLLER (6-6)

Index() function:

```
public function index()
{
    $employees = Employee::all();
    return view('index')->with('employees', $employees);
}
```



CREATE() AND STORE() FUNCTIONS(1-4)

create() and store() functions:

```
public function create()
{
    return view('add');
}
```



CREATE() AND STORE() FUNCTIONS (2-4)

create() and store() functions:

```
<!-- Stored in resources/views/add.blade.php -->

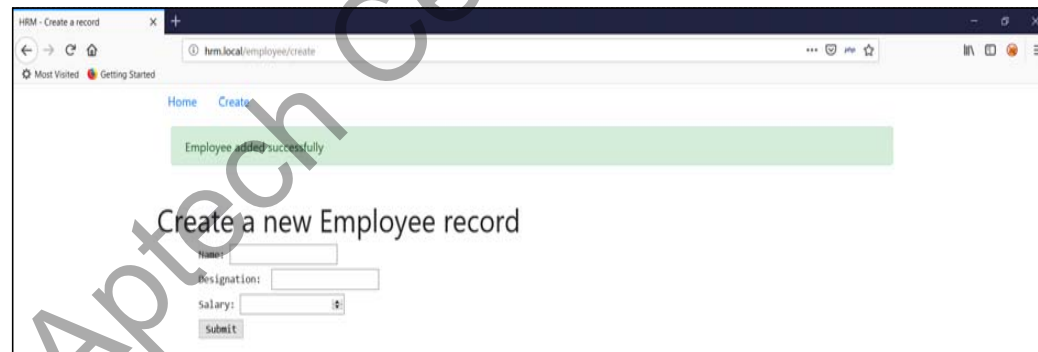
@extends('master')
@section('title')
    HRM - Create a record
@endsection
@section('heading')
    Create a new Employee record
@endsection

@section('content')
    {{ Form::open(['route'=>'employee.store','method'=>'post']) }}
    <pre>
        {{ Form::label('name','Name:') }} {{ Form::text('name') }}
        {{ Form::label('designation','Designation:') }} {{ Form::text('designation')}}
        {{ Form::label('salary','Salary:') }} {{ Form::number('salary','value') }}
        {{ Form::submit('Submit') }}
    </pre>
    {{ Form::close() }}
@endsection
@section('message')
    @if ($message = Session::get('Message'))
        <div class="alert alert-success">{{ $message }}</div>
    @endif
@endsection
```

CREATE() AND STORE() FUNCTIONS (3-4)

create() and store() functions:

```
public function store(Request $request)
{
    Employee::create($request->all());
    return redirect()->route('employee.create')->with('Message','Employee added successfully');
}
```

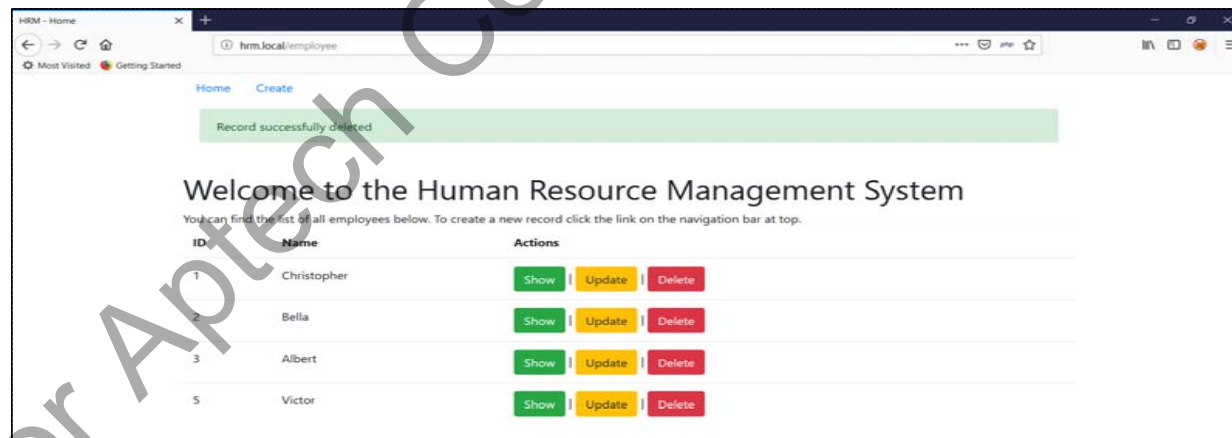


The screenshot shows a web browser window with the URL `http://localhost/employee/create`. The page has a navigation bar with 'Home' and 'Create' links. A green success message 'Employee added successfully' is displayed. Below it, the heading 'Create a new Employee record' is followed by a form with three input fields: 'Name:', 'Designation:', and 'Salary:'. A 'Submit' button is at the bottom of the form.

CREATE() AND STORE() FUNCTIONS (4-4)

Destroy():

```
public function destroy($id)
{
    $employee = Employee::find($id);
    $employee->delete();
    return redirect('employee')->with('Message', 'Record successfully deleted');
}
```



REQUESTS AND RESPONSE

```
public function update(Request $request, $id)
{
    $employee = Employee::find($id);
    $employee->name = $request->input('name');
    $employee->designation = $request->input('designat
ion');
    $employee->salary = $request -> input('salary');
    $employee->save();
    return redirect(action('EmployeeController@edit',$
employee->id))->with('Message','Record Modified successful
ly');
}
```



BASIC VALIDATIONS IN CONTROLLERS (1-4)

```
public function store(Request $request)
{
    $this->validate($request, [
        'name' => 'required | string | max:100',
        'designation' => 'required',
        'salary' => 'required | numeric | between:0,1000000.00',
    ]);
    Employee::create($request->all());
    return redirect()->route('employee.create')->with(
        'Message', 'Employee added successfully');
}
```


BASIC VALIDATIONS IN CONTROLLERS (2-4)

```
<!-- Stored in resources/views/create.blade.php -->
@extends('master')
@section('title')
    HRM - Create a record
@endsection

@section('heading')
    Create a new Employee record
@endsection

@section('content')
    {{Form::open(['route'=>'employee.store', 'method'=>'post'])}}
    <pre>
        {{ Form::label('name', 'Name:') }} {{ Form::text('name') }}
        {{ Form::label('designation', 'Designation:') }} {{Form::text('designation')}}
        {{ Form::label('salary', 'Salary:') }} {{ Form::number('salary', 'value') }}
        {{ Form::submit('Submit') }}
    </pre>
    {{ Form::close() }}
@endsection
```

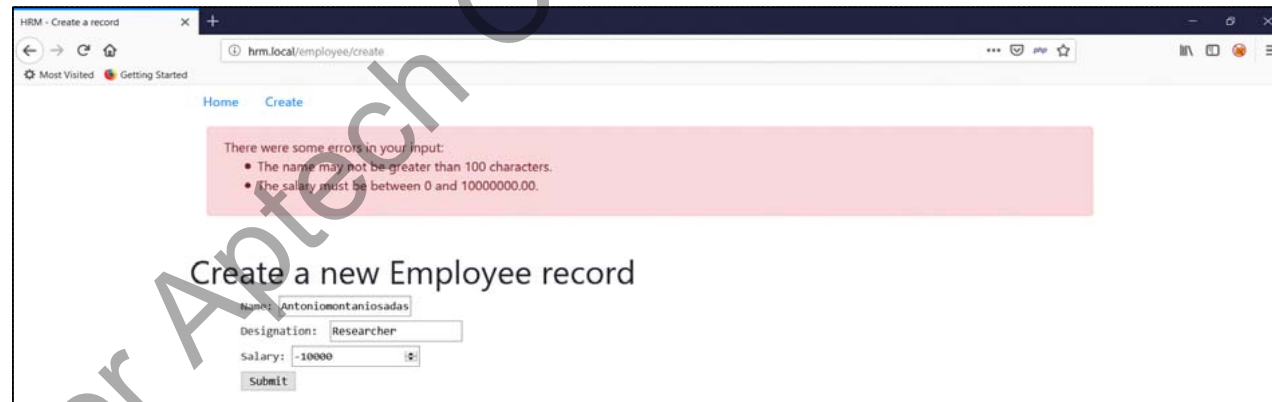
BASIC VALIDATIONS IN CONTROLLERS (3-4)

```
@section('message')
@if ($message = Session::get('Message'))
    <div class="alert alert-success">{{ $message }}</div>
@endif

@if ($errors->any())
    <div class="alert alert-danger">There were some errors in your input:
        <ul>
            @foreach ($errors->all() as $error)
                <li>{{ $error }}</li>
            @endforeach
        </ul>
    </div>
@endif
@endsection
```

BASIC VALIDATIONS IN CONTROLLERS (4-4)

```
@if ($errors->any())
    <div class="alert alert-danger">There were some errors in your input:
        <ul>
            @foreach ($errors->all() as $error)
                <li>{{ $error }}</li>
            @endforeach
        </ul>
    </div>
@endif
```



The screenshot shows a web browser window with the URL `hlm.local/employee/create`. The page has a navigation bar with 'Home' and 'Create' links. A red alert box displays the message: 'There were some errors in your input:' followed by two bullet points: '• The name may not be greater than 100 characters.' and '• The salary must be between 0 and 10000000.00.' Below the alert, the form title is 'Create a new Employee record'. The form contains three input fields: 'Name' with the value 'Antonimontanosadas', 'Designation' with the value 'Researcher', and 'Salary' with the value '-10000'. A 'Submit' button is at the bottom of the form.

ADDING PAGINATION

Pagination

- Sends records in blocks
- Saves bandwidth
- Enhances accessibility
- Integrated with the query builder and the Eloquent ORM

```
$employee = Employee::paginate(5);
```

```
public function index()  
{  
    $employees = Employee::paginate(5);  
    return view('index')->with('employees', $employees);  
}
```

UNDERSTANDING SESSIONS (1-2)

Sessions

- Session configuration is stored at `config/session.php`
- Drivers in Laravel to store sessions:
 - array
 - cookie
 - database
 - file
 - memcached/redis

UNDERSTANDING SESSIONS (2-2)

```
'driver' => env('SESSION_DRIVER', 'file'),
```

```
'driver' => env('SESSION_DRIVER', 'database'),
```

```
php artisan session:table  
php artisan migrate
```

```
$request->session()->put('key', 'value');  
  
$value = $request->session()->get('key1');
```



AUTHENTICATION IN LARAVEL (1-3)

```
php artisan make:auth
```

```
<?php
/*
|-----
| Web Routes
|-----
|
| Here is where you can register Web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "Web" middleware group. Now create something great!
|
*/

Route::get('/', function () {
    return redirect('employee');
});

Route::resource('employee', 'EmployeeController');
Auth::routes();

Route::get('/home', 'HomeController@index')->name('home');

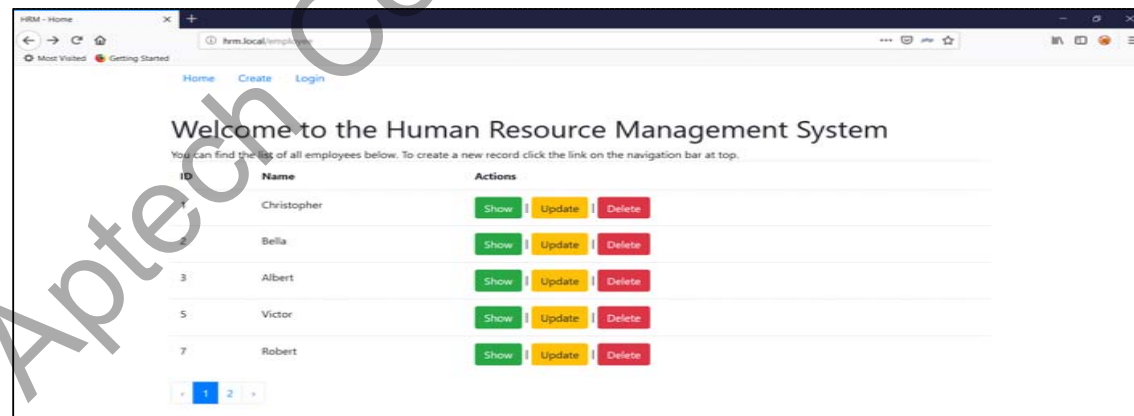
Auth::routes();

Route::get('/home', 'HomeController@index')->name('home');
```



AUTHENTICATION IN LARAVEL (2-3)

```
<ul class="nav">
  <li class="nav-item">
    <a class="nav-link" href="{{ route('employee.index') }}">Home</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="{{ route('employee.create') }}">Create</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="{{ url('/login') }}">Login</a>
  </li>
</ul>
```



AUTHENTICATION IN LARAVEL (3-3)

```
Route::resource('employee','EmployeeController')->middleware('auth');
```

```
$this->middleware('auth');
```

```
public function create()  
{  
    $this->middleware('auth');  
    return view('create');  
}
```



SUMMARY

- Controllers can be initiated using Single Action Controllers and Resource Controllers.
- A Resource Controller creates a Controller framework that is commonly used for CRUD operations.
- The resource routing feature automatically assigns routes to all functions that have been declared in the Resource Controller with a single line of code.
- The create() and store() functions are bundled together to form the Create function in the CRUD operations.
- The show() method facilitates the Read operation in the CRUD operations.
- Pagination refers to the concept of sending records in blocks.
- Laravel's paginator is integrated with the query builder and the Eloquent ORM.

