

Processing and Validating User Input



Mateo Prigl

Software developer

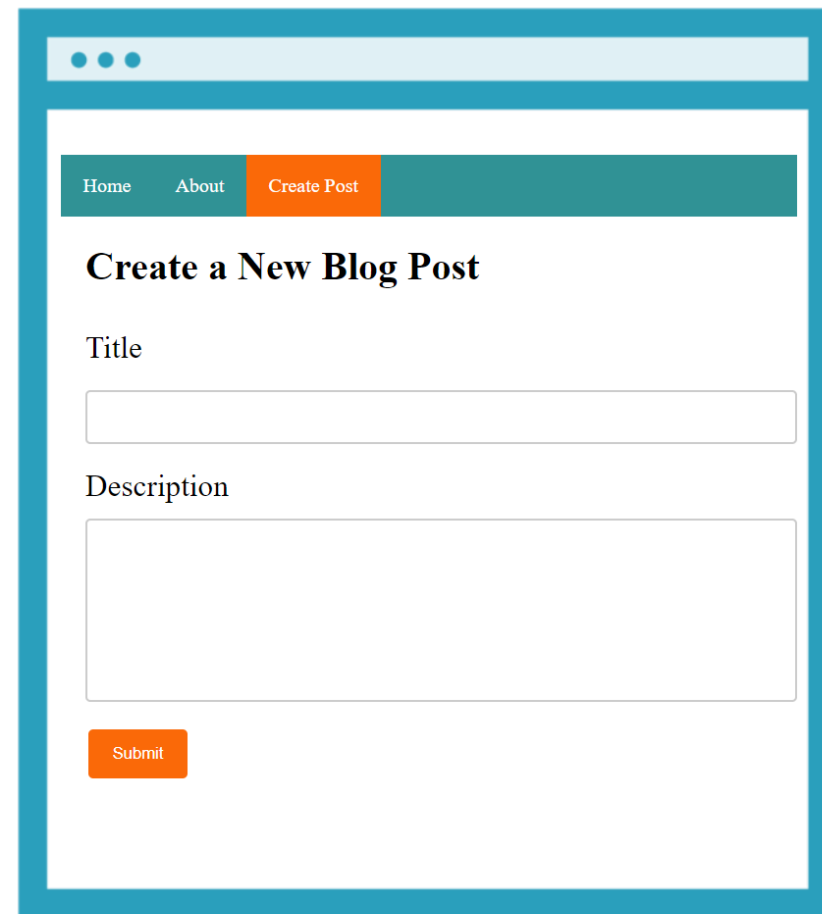


Cross Site Request Forgery (CSRF)

A type of attack that tricks the site visitor into submitting a malicious unwanted request to another website in which they are currently authenticated.



Cross Site Request Forgery (CSRF)



Home About Create Post

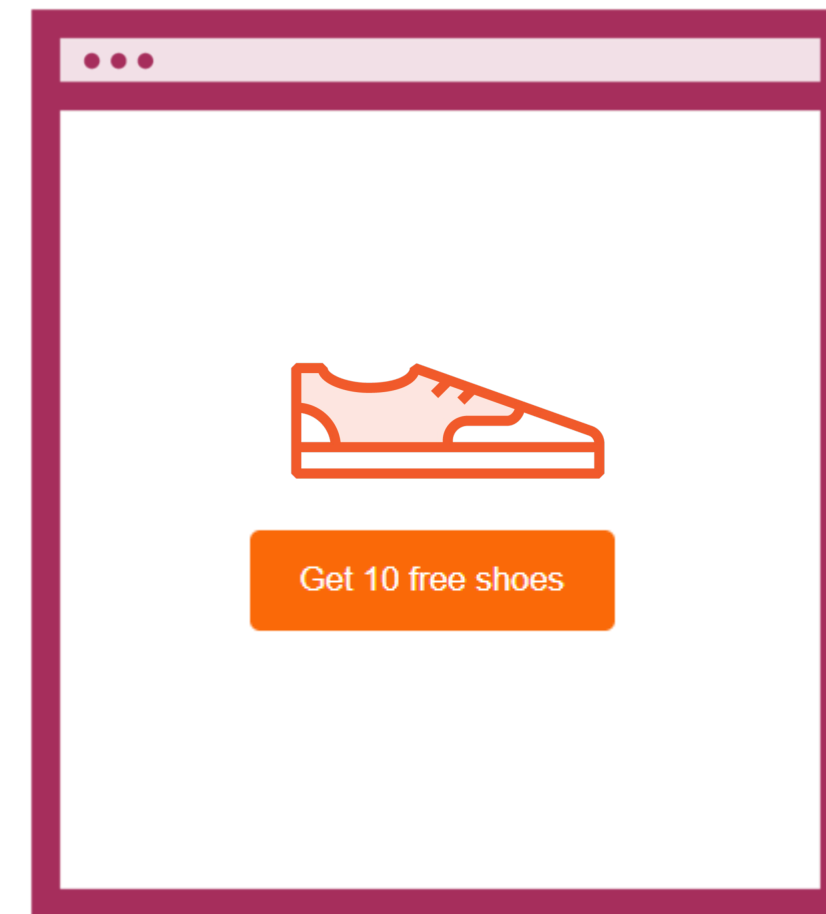
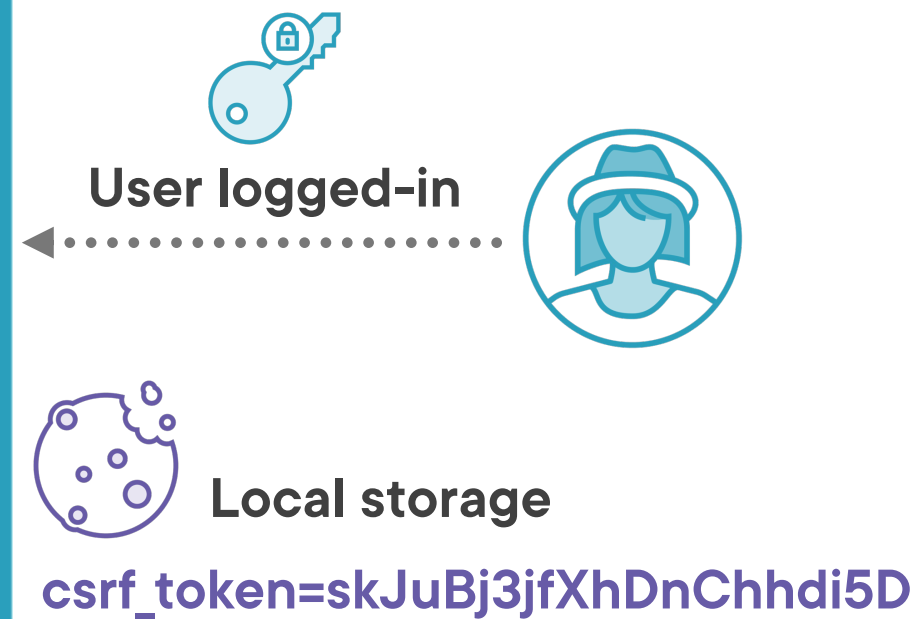
Create a New Blog Post

Title

Description

Submit

<https://our-blog.com/posts/create>



Get 10 free shoes

<https://definitely-legit-site.com>

```
...  
<form method="POST" action="/posts">  
  <input type="hidden" name="_token"  
    value="skJuBj3jfXhDnChhdi5D">  
  <input type="text" name="title">  
  <textarea name="description"></textarea>  
  <button type="submit">Submit</button>  
</form> ...
```

```
...  
<form method="POST"  
  action="https://our-blog.com/posts/">  
  <button type="submit">Get 10 free shoes</button>  
  <input type="hidden" value="Get 10 free shoes"  
    name="title">  
  <input type="hidden" value="Get free shoes at  
    definitely-legit-site.com!!!" name="description">
```



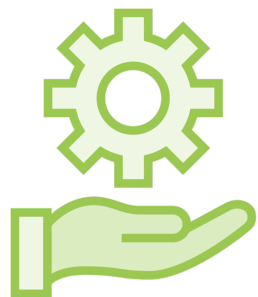
Architecture Concepts in Laravel



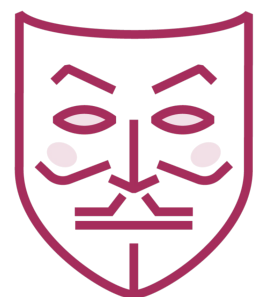
Request Lifecycle



Service Container



Service Providers



Facades



Service Container

Service Container / Application Instance

```
$bindings = [];
```



*** This is a simplified presentation of the service container functionality**



Service Container

Service Container / Application Instance

```
$bindings = [  
    'abstract' => function(){  
        // Callable factory function  
        return new Class();  
    },  
];
```



* This is a simplified presentation of the service container functionality



Service Container

Service Container / Application Instance

```
use Illuminate\Http\Request;

$bindings = [
    'request' => function(){
        return new Request(
            new DependencyClass(new HelperClass),
            $request_config);
    },
];

$aliases = [
    'Illuminate\Http\Request' => 'request'
];
```



```
// app() helper returns the service container
instance
$request = app()->make('request');

$request->input('title');
```

* This is a simplified presentation of the service container functionality



Service Container

Service Container / Application Instance

```
use Illuminate\Http\Request;

$bindings = [
    'request' => function(){
        return new Request(
            new DependencyClass(new HelperClass),
            $request_config);
    },
];

$aliases = [
    'Illuminate\Http\Request' => 'request'
];
```



```
// app() helper returns the service container
instance
// $request = app()->make('request');
$request = app()->make(Request::class);

$request->input('title');
```

* This is a simplified presentation of the service container functionality



SOLID Principles of Object Oriented Design

SOLID



Single Responsibility Principle

**A class should ideally have only one
responsibility**



Architecture Concepts in Laravel



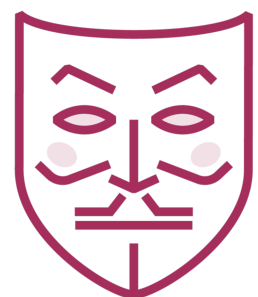
Request Lifecycle



Service Container



Service Providers



Facades

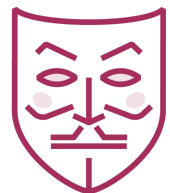


Facades

routes/web.php

```
use Illuminate\Support\Facades\Request;
...
Request::input('title');
```

Does this facade contain the `input()` method?



Request Facade

```
class Request extends Facade
{
    protected static function getFacadeAccessor()
    {
        return 'request';
    }
}
```

* This is a simplified presentation of the facade functionality

.....▶ **Request** facade does not implement the `input()` method.

Extended **Facade** class will use the `__callStatic()` magic method to pass the `input()` call to the `Illuminate\Http\Request` object.

Object will be instantiated from the service container. The abstract key `'request'` is provided from the `getFacadeAccessor()` method.

The result should be the returned value of:
`app()->make('request')->input('title');`



Facade Class

```
public static function __callStatic($method_name,
    $args)
{
    return app()
        ->make(static::getFacadeAccessor())
        ->$method_name();
}
```



Summary



Rendering Web Forms

Define the view

```
Route::get('/posts/create', function () {  
    return view('create');  
})->name('posts.create');
```

routes/web.php

Create the form

```
<form method="POST"  
action="{{ route('posts.store') }}">  
  
@csrf  
<input type="text" name="title">  
<textarea name="description"></textarea>  
<button type="submit">Submit</button>  
  
</form>
```

resources/views/create.blade.php



Processing Form Input

Process the form input with the **`request()`** helper

```
Route::post('/posts', function () {  
    request()->input('title');  
  
})->name('posts.store');
```

routes/web.php

Create the form

```
<form method="POST"  
action="{{ route('posts.store') }}">  
  
@csrf  
<input type="text" name="title">  
<textarea name="description"></textarea>  
<button type="submit">Submit</button>  
  
</form>
```

resources/views/create.blade.php



Processing Form Input

Process the form input with the
Request facade

```
use Illuminate\Support\Facades\Request;

Route::post('/posts', function () {
    Request::input('title');

})->name('posts.store');
```

routes/web.php

Create the form

```
<form method="POST"
action="{{ route('posts.store') }}">

@csrf
<input type="text" name="title">
<textarea name="description"></textarea>
<button type="submit">Submit</button>

</form>
```

resources/views/create.blade.php

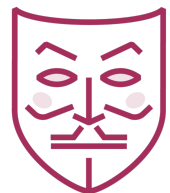


Facades

routes/web.php

```
use Illuminate\Support\Facades\Request;
...
Request::input('title');
```

Does this facade contain the `input()` method?



Request Facade

```
class Request extends Facade
{
    protected static function getFacadeAccessor()
    {
        return 'request';
    }
}
```

* This is a simplified presentation of the facade functionality

`Request` facade does not implement the `input()` method.

Extended `Facade` class will use the `__callStatic()` magic method to pass the `input()` call to the `Illuminate\Http\Request` object.

`Object` will be instantiated from the service container. The abstract key `'request'` is provided from the `getFacadeAccessor()` method.

The result should be the returned value of:
`app()->make('request')->input('title');`



Facade Class

```
public static function __callStatic($method_name,
    $args)
{
    return app()
        ->make(static::getFacadeAccessor())
        ->$method_name();
}
```



Processing Form Input

Process the form input with the
Request facade

```
use Illuminate\Support\Facades\Request;

Route::post('/posts', function () {
    Request::input('title');

})->name('posts.store');
```

routes/web.php

Create the form

```
<form method="POST"
action="{{ route('posts.store') }}">

@csrf
<input type="text" name="title">
<textarea name="description"></textarea>
<button type="submit">Submit</button>

</form>
```

resources/views/create.blade.php



Processing Form Input

Process the form input with the dependency injection

```
use Illuminate\Http\Request;

Route::post('/posts', function () {

    })->name('posts.store');
```

routes/web.php

Create the form

```
<form method="POST"
action="{{ route('posts.store') }}">

@csrf
<input type="text" name="title">
<textarea name="description"></textarea>
<button type="submit">Submit</button>

</form>
```

resources/views/create.blade.php



Processing Form Input

Process the form input with the dependency injection

```
use Illuminate\Http\Request;

Route::post('/posts',
    function (Request $request)
    {
        $request->input('title');

    })->name('posts.store');
```

routes/web.php

Create the form

```
<form method="POST"
action="{{ route('posts.store') }}">

@csrf
<input type="text" name="title">
<textarea name="description"></textarea>
<button type="submit">Submit</button>

</form>
```

resources/views/create.blade.php



Service Container

Service Container / Application Instance

```
use Illuminate\Http\Request;

$bindings = [
    'request' => function(){
        return new Request(
            new DependencyClass(new HelperClass),
            $request_config);
    },
];

$aliases = [
    'Illuminate\Http\Request' => 'request'
];
```

```
// app() helper returns the service container
instance
// $request = app()->make('request');
$request = app()->make(Request::class);

$request->input('title');
```



* This is a simplified presentation of the service container functionality



Validating User Input

Validate user input from the request object with validation rules

```
$request->validate([  
    'title' => 'required',  
    'description' => ['required', 'min:10'],  
]);
```



Repopulate Old Input and Show Validation Errors

```
@if ($errors->any())
    @foreach ($errors->all() as $error)
        <p>{{ $error }}</p>
    @endforeach
@endif

...

<input class="@error('title') error-border @enderror" type="text" name="title"
value="{{ old('title') }}">
@error('title')
    {{ $message }}
@enderror
```



Display Messages from the Session

**Redirect the user
and flash a message to the session**

```
...  
  
Route::post('/posts', function () {  
    ...  
  
    return redirect()  
        ->route('posts.create')  
        ->with('success', 'Flash message');  
})->name('posts.store');
```

routes/web.php

Retrieve messages from the session

```
@if (session('success'))  
    <div>  
        {{ session('success') }}  
    </div>  
@endif
```

resources/views/layout.blade.php



Include Subviews (Partials)

Create a subview template

```
<!-- Some Blade template markup -->
```

resources/views/_errors.blade.php

Include a subview in another template

```
@include('_errors')
```

```
<!-- Include when condition is true -->
```

```
@includeWhen(condition, '_errors')
```

resources/views/layout.blade.php



Up Next:

Implementing CRUD Operations with the
Resource Controller

