LOGICA DEI FORM

Per gestire l'acquisizione di dati da parte di un utente faremo una richiesta di tipo ***Post***.
La richiesta viene gestita *attraverso una rotta* che creiamo.
Ad essa *associamo un controller* dedicato che creiamo (*php artisan make:controller*), e ricordiamoci di collegare la ***use***.

```
Route::post('/contacts/submit' , [ContactController::class,'submit'])->name('contacts.submit')
```

Nel controller dedicato attiviamo la funzione *submit()* e come parametro facciamo una *dependency injection* ad una classe predefinita di Laravel che gestisce l'afflusso di dati in entrata.

```
class ContactController extends Controller
{
	public function submit(ContactRequest $request){
		//dd($request->all()); verifichiamo se stiamo acquisendo dati
	}
}
```

Creiamo una nostra classe Request per gestire al meglio la ***validation*** dei dati in entrata.
- php artisan make:request ContactRequest crea il file php in *app/Http/Requests*

```
class ContactRequest extends FormRequest
{
	public function authorize(){
		return true;
	}

	public function rules(){
		return [
			'username'=>'required|min:4|max:15',
			'email'=>'required|email:rfc,dns',
			'message'=>'required|min:10|max:150'
			];
	 }
}
```

authorize . Funzione che determina se siamo autorizzati ad effettuare richieste, *true.*
rules . Qui inseriamo tutta la logica della validazione. La lista completa dei comandi si trova nella documentazione sotto *available validation rules.*
required . Rende obbligatorio l'inserimento di quel parametro.
min|max|email . Validazioni presenti in *available validation rules* nella documentazione.

```
<form class="form-signin" action="{{route('contacts.submit')}}" method="POST">
@csrf
@if ($errors->any())
	<div class="alert alert-danger">
		<ul>
			@foreach ($errors->all() as $error)
				<li>{{ $error }}</li>
			@endforeach
		</ul>
	</div>
@endif
	<div class="form-label-group">
		<input type="text" class="form-control" placeholder="Username"
				name="username" value="{{old('username')}}">
		<label for="inputUsername">Username</label>
	</div>

	<div class="form-label-group">
		<input type="email" class="form-control" placeholder="Email
			address" name="email" value="{{old('email')}}">
		<label for="inputEmail">Aggiungi la tua Email</label>
	</div>

	<div class+"form-label-group">
		<textarea name="message" cols="30" rows="10">{{old('message')}}
</textarea>
	</div>

<button type="submit">Registrati</button>
```

action . Diamo una rotta ai dati quando verrà cliccato il *button type submit*
name . Indichiamo la key dell'array chiave=>valore che creiamo quando facciamo *submit*
method . Specifichiamo il tipo di rotta che utilizziamo.
@csrf . Genera un input con attributi *type hidden, name _token, value* con un codice che in risposta verrà riconosciuto dal Server. Ne genera uno ad ogni sessione per evitare il CSFR, ovvero la creazione di massa di user per bloccare il server. (***errore 419***)
@if / **@endif** . Per visualizzare gli *errori di validazione,* sulla documentazione (*Displaying the Validation Errors*) troviamo l'*if* da inserire nell'HTML per far scattare il messaggio di errore se non vengono rispettate le ***rules*** nella nostra ***request***
value . Diamo questo attributo di valore *{{old('nameValue')}}* agli input per visualizzare il testo scritto in precedenza in caso di una Error Validation. Nel *textarea* l'attributo non è previsto e si inserisce il valore direttamente tra i tag.