

---

# Laravel - Perusteet

Laravelin käytön perusteet esitetään keskittyen ydintoiminnallisiin ytimekkäiden esimerkkien avulla. Kaikkia mahdollisia käyttötapoja ja yksityiskohtia ei käsitellä.

Oletuksena on, että olet asentanut edellisessä luvussa esitetyllä tavalla Laravel-sovelluksen joko Windows/XAMPP- tai Ubuntu-ympäristöön `projekti01`-kansioon.

Tässä esityksessä käytetään esimerkkinä Ubuntuun `testi`-käyttäjänä kansioon

`/home/testi/public_html/projekti01` asennettua Laravel-sovellusta.

Esimerkkipalvelimen IP-numero on 192.168.1.126, joten sovelluksen päänäkömä löytyy osoitteesta <http://192.168.1.126/~testi/projekti01/public/>

---

## Laravelin juurikansio eli projektikansio projekti01

Laravelin juurikansiona toimii projektikansio

`/home/testi/public_html/projekti01`. Alikansio `public` voi sijaita muuallakin ja asiaa käsiteltiin edellisessä luvussa.

Laravelin kansiorakennetta ja eri kansioden tarkoitettuja sisältöjä on selitetty esim. dokumenteissa <https://laravel.com/docs/5.7/structure> ja

<https://www.w3schools.in/laravel-tutorial/application-directory-structure/>. Kaikki viittaukset tässä materiaalissa kohdistuvat kehityksen kohteena olevaan Laravel-sovelluksen juurikansioon eli projektikansioon

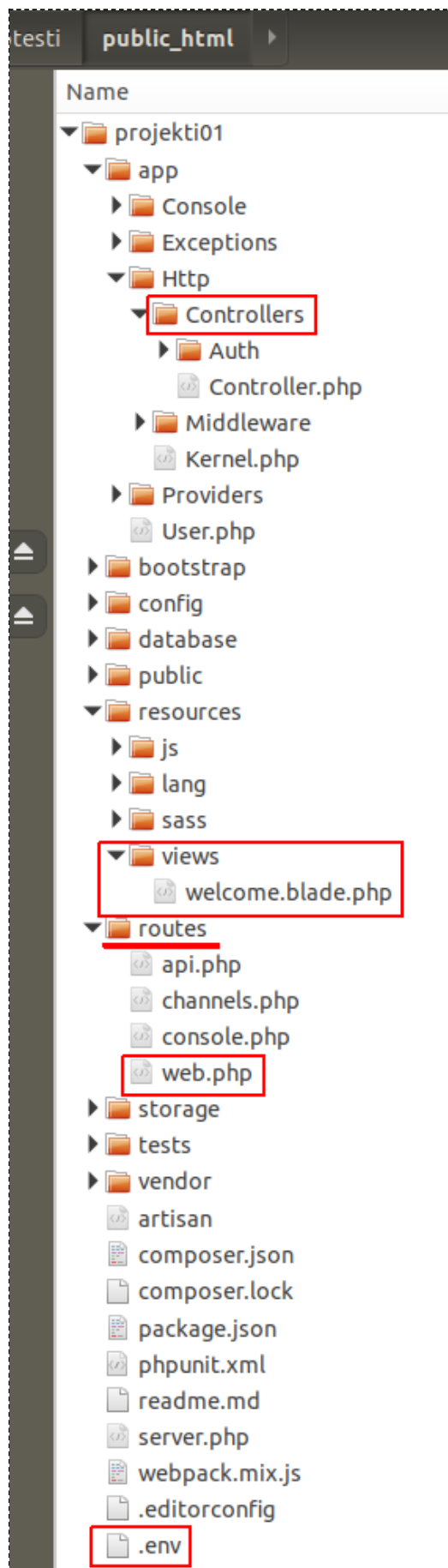
`/home/testi/public_html/projekti01`, jos erikseen ei muuta mainita. Ilmaisuu

`routes`-kansio tarkoittaa esimerkiksi kansiota

`/home/testi/public_html/projekti01/routes`

Oheisessa kuvassa on merkitty aloittamisen kannalta tärkeimmät sijainnit

- `routes`-kansion `web.php`-tiedosto sisältää kaikki Laravel-sovelluksen tarjoamat tavalliset reitit
- `resources/views`-kansio sisältää näkymät
- `app/Http/Controllers`-kansio sisältää kontrollerit
- `.env`-tiedosto sisältää tärkeitä sovelluksen asetuksia mm. tietokanta-asetukset



Hyvä tietää aluksi

Laravel-kehiksen käytössä useita tehtäviä on suoraviivaisinta suorittaa komentokehoteesta Laravel-sovelluksen juurikansioista. Windows-ympäristössä komentokehoteen voi käynnistää `cmd`-komennolla tai käyttämällä PowerShellia. Linux-ympäristöön löytyy monenlaisia ohjelmia komentokehoteeseen pääsemiseksi. Windows-ympäristöstä Linuxin komentokehoteeseen (~shell, ~komentorivi) pääsee yleensä järkevimmin ssh-päätelytydellä käyttämällä Putty-ohjelmaa

## Artisan Console: `php artisan`

[Artisan Console](#) on komentorivikäyttöliittymä monien Laravel-tehtävien suoraviivaiseen suorittamiseen. **Kaikki artisan-komennot kohdistuvat siihen Laravel-sovellukseen, jonka projektikansiossa komento annetaan!**

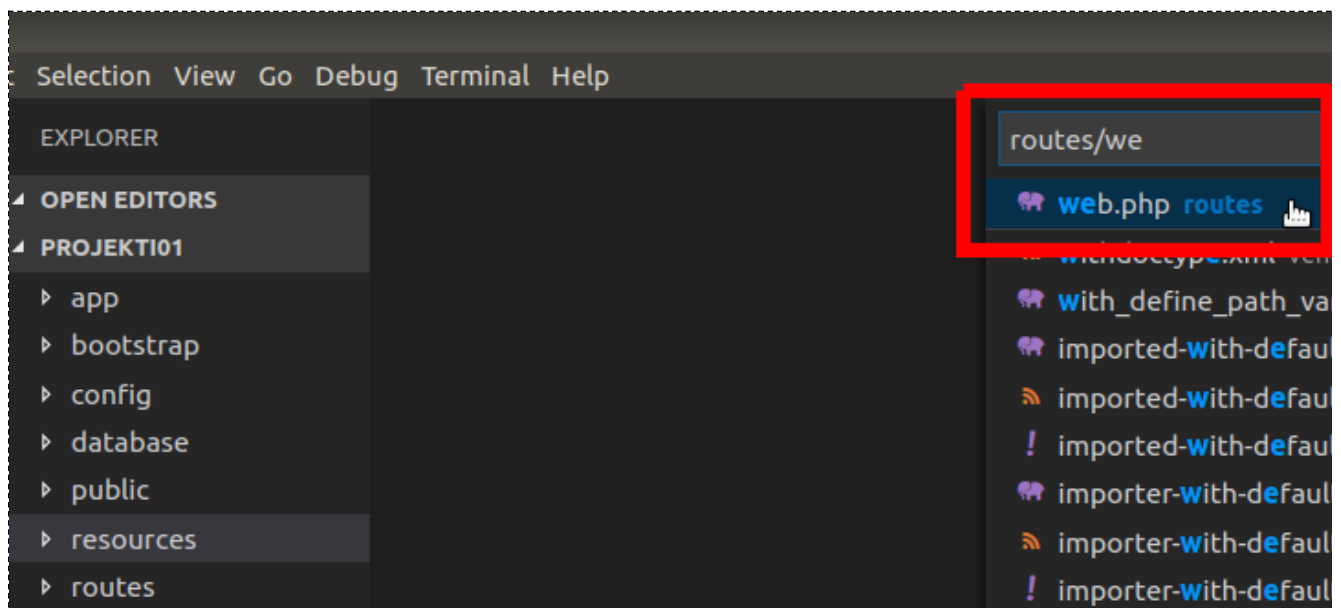
Komento `php artisan` listaa kaiken tarjolla olevan toiminnallisuuden

```
php artisan
```

## Visual Studio Code Laravel-projektin kanssa

Jos koodieditorivalintasi on Visual Studio Code (VSC), Laravel-projektin voi avata käsiteltäväksi `File/Open Folder` -toiminnolla. Tällöin halutun tiedoston löytäminen isosta tiedostomassasta on nopeampaa `Go To File` -toiminnolla (`CTRL+P`). Kaikki VSC:n tarjoamat komennot voit listata `SHIFT+CTRL+P`-komennolla.

Komentamalla `CTRL+P` ja kirjoittamalla avautuvaan tekstikenttään halutun polun alkua esim "`routes/we`" kuvan mukaan VSC ehdottaa avattavaksi hakuun sopivia tiedostoja esim. usein tarvittavaa `routes/web.php`-tiedostoa.



## Haloo Laravel! -teksti etusivulle

Reittitiedostossa on asennuksen jälkeen määritelty yksi reitti

### routes/web.php

```
1 Route::get('/', function () {  
2     return view('welcome');  
3 });
```

reitti viittaa näkymään, joka löytyy tiedostostosta

resources/views/welcome.blade.php. Muuta tiedostone sisältöä oheisella tavalla

### resources/views/welcome.blade.php

```
1 ...  
2         <div class="content">  
3             <div class="title m-b-md">  
4                 Haloo Laravel  
5             </div>  
6 ...
```

Selaimeen URL: <http://192.168.1.126/~testi/projekti01/public/>

Sivulle tulostuu teksti "Haloo Laravel!"

- Reittitiedoston kauttaviiva on reitti, joka tulee ip-numeron jatkeeksi
- Reitin päässä on näkymä nimeltä `welcome`. Laravel etsii näkymäksi tiedoston joka on nimetty `resources/views/welcome.blade.php`.
- Ei käytä kontrolleria. Route->View (nk. "closure")

## Oma /about-reitti

Luodaan reitti `/about` **lisäämällä** seuraava määrittely reittitiedostoon

### routes/web.php

```
1 Route::get('/about', function () {  
2     return view('siteinfo');  
3 });
```

Luodaan tiedosto

### resources/views/siteinfo.blade.php

```
1 <h3>Hey! Olen /about-reitin päässä näkyvä resources/views/siteinfo.blade.php
```

Selaimeen URL: <http://192.168.1.126/~testi/projekti01/public/about>

Sivulle tulostuu teksti "Hey! Olen /about-reitin päässä näkyvä resources/views/siteinfo.blade.php-tiedosto"

## Oma /aboutfolder-reitti

Luodaan reitti /aboutfolder **lisäämällä** seuraava määrittely reittitiedostoon

routes/web.php

```
1 Route::get('/aboutfolder', function () {  
2     return view('kansio/siteinfo');  
3 });
```

Luodaan tiedosto

resources/views/kansio/siteinfo.blade.php

```
1 <h3>Hey! Olen /aboutfolder-reitin päässä näkyvä resources/views/kans
```

Selaimeen URL: <http://192.168.1.126/~testi/projekti01/public/aboutfolder>

Sivulle tulostuu teksti "Hey! Olen /aboutfolder-reitin päässä näkyvä resources/views/kansio/siteinfo.blade.php-tiedosto"

- Huomaa, että **luodun** kansion resources/views/kansio/ näkymään siteinfo.blade.php viitataan kauttaviiva-notaatiolla kansio/siteinfo. Kauttaviivan tilalla voidaan käyttää myös pistettä.

## Kontrollerit

### Oma Contact-kontrolleri

Luodaan aluksi tarvittavat reitit

routes/web.php

```
1 Route::get('/contact1', 'ContactController@showinfo');  
2 Route::get('/contact2', 'ContactController@returninfo');
```

Luodaan uusi kontrolleri-tiedosto app/Http/Controllers/ContactController.php artisan-komennolla:

```
php artisan make:controller ContactController
```

=> Controller created successfully.

Editoidaan luodun tiedoston `app/Http/Controllers/ContactController.php` sisällöksi

#### app/Http/Controllers/ContactController.php

```
1  <?php
2
3  namespace App\Http\Controllers;
4  use Illuminate\Http\Request;
5
6  class ContactController extends Controller
7  {
8      public function showinfo()
9      {
10         return view('contact');
11     }
12
13     public function returninfo()
14     {
15         return 'Olen ContactControllerin return-lause';
16     }
17 }
```

Luodaan näkymä

#### resources/views/contact.blade.php

```
1  <h3>Olen ContactControllerin showinfo()-metodin contact-näkymä</p><
```

#### Lopputulokset seaimessa

<http://192.168.1.126/~testi/projekti01/public/contact1>

---> Olen ContactControllerin showinfo()-metodin contact-näkymä

<http://192.168.1.126/~testi/projekti01/public/contact2>

--->Olen ContactControllerin return-lause

#### Huomioita ContactControllerista

- `php artisan`-komento pitää antaa laravel-projektikansiossa (~työhakemistona)
- `php artisan`-komendo luo lähes tyhjän kontrolleritiedoston. Hyötynä on se, että namespace- ja use-viittaukset ovat oikein käytetylle Laravelin versiolle ja ympäristölle.
- Kontrolleritiedoston voi luoda myös tyhjästä tekstieditorilla

- Kontrolleritiedoston sijainnin ja nimen tulee noudattaa esitettyä muotoa **NimiController.php** eli kontrollerin nimi isolla alkukirjaimella ja kirjoitetaan yhteen merkkijonon Controller.php kanssa. Luokan määrittelyn kontrolleritiedoston sisällä tulee olla myös nimen mukainen.

## Datan välittäminen näkymälle

Luodaan pari uutta reittiä

### routes/web.php

```
1 Route::get('/person', 'ContactController@showperson');
2 Route::get('/listpersons', 'ContactController@listpersons');
```

Lisätään ContactControlleriin seuraavat metodit

### app/Http/Controllers/ContactController.php

```
1 public function showperson()
2 {
3     $name = "<span style='background-color:#ffc;'>Raaka-Arska</span>";
4     return view('person')->with('name', $name);
5 }
6
7 public function listpersons()
8 {
9     $persons = ['Guru Ken', 'Ainen Sani', 'Tana Ruu'];
10    return view('persons')->with('persons', $persons);
11 }
```

Luodaan näkymät

### resources/views/person.blade.php

```
1 <h2>Muuttujat Laravelin Blade Template -tiedostossa</h2>
2
3 <hr>
4 <h4>Muuttuja &dollar;name PHP-syntaksilla</h4>
5
6 <table border=1>
7 <tr><th>Syntaksi</th><th>Tulos</th></tr>
8 <tr>
9 <td><code>&lt;&quest;&equals;&dollar;name&semi; &quest;&gt;</code></td>
10 <td><?=$name; ?></td>
11 </tr>
12 </table>
13
```

```

14 <hr>
15 <h4>Muuttuja &dollar;name Laravelin Blade Template -syntaksilla:</h4>
16
17 <table border=1>
18 <tr><th>Syntaksi</th><th>Tulos</th></tr>
19 <tr>
20 <td><code>&lbrace;&lbrace; name &rbrace;&rbrace;</code></td>
21 <td>{{ $name }}</td>
22 </tr>
23 </table>
24
25 <hr>
26 <h4>Muuttuja &dollar;name Laravelin Blade Template -syntaksilla:</h4>
27
28 <table border=1>
29 <tr><th>Syntaksi</th><th>Tulos</th></tr>
30 <tr>
31 <td><code>&lbrace;&excl;&excl; name &excl;&excl;&rbrace;</code></td>
32 <td>{!! $name !!}</td>
33 </tr>
34 </table>

```

ja

resources/views/persons.blade.php

```

1 @if (count($persons))
2 <h3>Hurja sakki:</h3>
3 <ul>
4     @foreach ($persons as $person)
5         <li>{{ $person }}</li>
6     @endforeach
7 </ul>
8 @endif

```

### Lopputulokset seaimessa

Tutki ja ymmärrä lopputulokset nähdyn ohjelmakoodin perusteella

<http://192.168.1.126/~testi/projekti01/public/person>



## Muuttujat Laravelin Blade Template -tiedostossa

Muuttuja \$name PHP-syntaksilla

Syntaksi	Tulos
<?=\$name; ?>	Raaka-Arska

Muuttuja \$name Laravelin Blade Template -syntaksilla:

Syntaksi	Tulos
{{ name }}	<span style='background-color:#ffc;'>Raaka-Arska</span>

Muuttuja \$name Laravelin Blade Template -syntaksilla:

Syntaksi	Tulos
{!! name !!}	Raaka-Arska

<http://192.168.1.126/~testi/projekti01/public/listpersons>

### Hurja sakki:

- Guru Ken
- Ainen Sani
- Tana Ruu

Lisää toiminnallisuutta voit lukea dokumentaatiosta

- [Blade Template - Laravel](#)

master-template

Luodaan sivustolle layouts-kansio ja master-template tiedosto

resources/views/layouts/app.blade.php seuraavasti

resources/views/layouts/app.blade.php

```
1 <html>
2   <head>
3     <title>InfoSite</title>
4   </head>
5   <body>
6     <h3>InfoSite</h3>
7     <div class="infobar">
8       @section('infobar')
9         <p>Ei infoa</p>
10      @show
```

```

11         </div>
12
13         <div class="container">
14             @yield('content')
15         </div>
16     </body>
17 </html>

```

- Huomaa infobar-niminen @section-merkintä, joka päättyy merkintään @show. Tämän kohdan *Ei infoa* -teksti korvataan seuraavana esitettävän lapsi-templatien infobar-sektiolla, **jos ja vain jos** sellainen on olemassa
- Huomaa @yield('content') -merkintä, johon voidaan tuoda sisältöä lapsi-templatien content-sektiosta

Luodaan yksittäisen sivun lapsi-template `resources/views/child.blade.php` seuraavasti

#### resources/views/child.blade.php

```

1  @extends('layouts.app')
2
3
4  @section('infobar')
5      <p><span style='background-color: #ffc'>Tämä info tulee tiedosto
6          resources/views/child.blade.php</span></p>
7  @endsection
8
9  @section('content')
10     <p>Lorem Ipsum, Lorem Ipsum...</p>
11 @endsection

```

- Huomaa merkintä @extends('layouts.app'), joka kertoo mihin leiskaan tässä esitettävät sektiot liitetään.
- Sektiot @section('infobar') ja @section('content') liitetään master-templateen sinne merkityille paikoille

Sivun layout näyttää seuraavalta:

## InfoSite

Tämä info tulee tiedostosta  
resources/views/child.blade.php

Lorem Ipsum, Lorem Ipsum...

