Routes

La nostra applicazione web riceve i comandi dalla barra degli indirizzi del browser. L'url scritta conterrà oltre all'indirizzo del sito anche una serie di stringhe di testo che verranno interpretate da CI4 ricavandone il nome di un controller e di un metodo da invocare con l'eventuale passaggio di alcuni parametri. Il controller e il metodo possono essere passati direttamente nell'url oppure se si vuole evitare di mostrare i nomi dei controller e dei metodi della nostra applicazione si possono usare le routes, che creano una associazione tra un nome fittizio (nome della route) e la coppia Controller - metodo. Questo schema permette di sfruttare delle funzioni aggiuntive e rende più agevole la navigazione interna del nostro sito.

Vediamo nel dettaglio come funzionano questi due schemi:

- Controller/metodo
- Placeholder

Schema Controller/metodo

Nella barra degli indirizzi inseriremo dopo la semplice url del sito, i nostri comandi e passeremo anche dei parametri quando necessario separando il tutto con il carattere slash / (detto anche forward slash). Lo schema sarà questo:

> Schema Controller::metodo
>
> [**url sito internet**] [**comando**] [**eventuale parametro**]

CI4 sfrutta il carattere / (slash) per identificare le parti l'url. Nel dettaglio vediamo le parti contenute nella barra degli indirizzi del nostro browser:

url sito internet

La prima parte rappresenta l'indirizzo internet del sito su cui ci troviamo e viene riconosciuta da CI4 perché questo indirizzo é impostato nella variabile di configurazione *$baseURL* che troviamo in `app\Config\App.php`.

> A seconda delle configurazioni si potrà presentare anche *index.php* all'interno dell'url.

- on line avrò qualcosa di simile a questo:

```
https://mio.sito.com
```

- in locale posso trovare questo

```
http://localhost:8080
oppure
http://localhost:8080/index.php
```

comando

Qui siamo al punto principale: con il comando indichiamo quale azione eseguire alla nostra applicazione.

Il comando si compone di due parti separate del carattere /

```
controller / metodo
```

In tal modo scrivendo un url come questa:

```
mio.sito.com/prodotti/elenco
```

CI4 cercherà un controller con il nome ***Prodotti*** e al suo interno il metodo ***elenco***.

Ci sono a questo punto due considerazioni da fare:

1. Se la mia url si ferma al controller

   ```
   mio-sito.com/prodotti
   ```

   CI4 cercherà automaticamente il metodo ***index***

2. Se scrivo qualsiasi cosa dopo il controller ***prodotti /*** CI4 lo considererà come nome del metodo da cercare. Questo significa che come impostazione standard non protrò scrivere

   ```
   mio-sito.com/prodotti/12
   ```

   poichè 12 non sarà interpretato come parametro del metodo index, ma verrà visto come nome di un metodo da cercare nel controller Prodotti.php

parametro

Se l'url contiene il controller e il metodo allora posso inserire uno o più parametri sempre separati dal carattere slash(/). Se il metodo riceve meno parametri di quelli richiesti, verrà segnalato un errore.

Quindi se il metodo ***calcola*** che richiete tre parametri avrà questa struttura

```
class Esercizi extends BaseController
    {
        public function calcola($somma1, $somma2, $somma3)
            { ...
```

la mia url dovrà essere qualcosa di simile a questo:

```
www.mio.sito.com/esercizi/calcolatotale/20/30/10
```

Se passiamo meno dei tre parametri richiesti, la funzione/metodo richiamata emetterà un messaggio di errore.

Per non ricevere questo messaggio di errore nel caso l'url contenga meno parametri di quelli richiesti, basterà impostare dei velori di default:

```
public function calcola($somma1 = 10, $somma2 = 300, $somma3 = null)
    { ...
```

Schema Placeholder

Questo comportamento standard

[**url sito internet**] [**comando**] [**eventuale parametro**]

si può semplificare: il ***comando*** scritto nell'url si può sostituire con un ***nome***, passando da questa situazione

```
mio-sito/prodotti/modifica/12
```

a questa

```
mio-sito/elabora/12
```

Per ottenere questo risultato si vanno a creare le ***routes*** nel file `app\Config\Routes.php`

```
$routes->add('/elabora/(:num)', 'Prodotti::modifica/$1')
```

In questa nuova situazione fanno la loro comparsa i ***placeholders***: sono impiegati all'interno del nome della route e indicano, appunto, il posto occupato da un parametro. Oltre a questo impongono una regola: il parametro dovrà rispettare le indicazioni date dal particolare tipo di placeholders impiegato. Così ad esempio a (:num) dovrà corrispondere un dato contenente un valore numerico

Tipi di placeholder

Una premessa: se il segmento dedicato al parametro è previsto nel nome della route

```
... $routes->get('/modifica/(:num)', ...
```

allora dovrà essere presente altrimenti CI4 restituirà il messaggio di pagina non trovata.

CI4 permette di restringere il tipo di dati che può essere accettato. Partiamo dalla categoria più ampia scendendo man mano nelle categorie che impongono maggiori restrizioni ai tipi di caratteri consentiti. In tutte le tipologie di placeholder elencate il simbolo slash / viene riconosciuto solo come termine di un segmento

- **(:any)** accetta ogni tipo di carattere (alfanumerici e simboli) se compaiono più parametri (separati dallo slash / questi verranno tutti inviati alla funzione indicata)
- **(:segment)** accetta ogni tipo di carattere (alfanumerici e simboli) se compaiono più parametri (separati dallo slash /) verranno inviati tanti parametri quante volte compare (:segment).

  ```
  ...
  $routes->get('/modifica/(:segment)/(:segment)', ...
  ...
  invia i primi 2 parametri incontrati
  ```

- **(:hash)** equivalente a (:segment) usato per identificare le route con hash
- **(:alphanum)** caratteri alfanumerici senza simboli

- **(:alpha)** solo caratteri alfabetici senza simboli
- **(:num)** solo numeri senza simboli

Ordinamento delle routes

Se abbiamo un url come questa:

```
mio-sito.com/modifica/12
```

CI4 trova un valore numerico nel parametro (12) e scorrerà nell'ordine in cui sono elencate tutte le routes alla ricerca della prima route che sia compatibile. Se tra le nostre routes abbiamo solo un placeholder (:alpha) CI4 non troverà una route compatibile con ciò che legge nell'url e restituirà il messaggio di 'Pagina non trovata'.

Se scriviamo le route in questo ordine:

```
$routes->get('/modifica/(:alphanum)', ...
$routes->get('/modifica/(:alpha)', ...
$routes->get('/modifica/(:num)', ...
```

Non riusciremo ad intercettare nè (:alpha) e nè (:num) perchè con un parametro solo numerico o solo con caratteri alfabetici la prima route ad essere soddisfatta seguendo l'ordine con cui sono state scritte sarebbe (:alphanum) e qui si fermerebbe CI4.

Quindi va posta attenzione anche all'ordine in cui sono scritte le routes.

Placeholder e parametri multipli

Poniamo che il metodo creato richieda più parametri. Abbiamo due strategie per questa situazione.

1. Singolo placeholder **(:any)**:

   CI4 si aspetta almeno un parametro. Ma poichè (:any accetta anche il carattere slash (/) posso sfruttare questa caratteristica e passare con un singolo (:any) tutti i parametri che voglio

```
$routes->get('/addiziona/(:any)', 'Calc::sum($p1,$p2,$p3)');

//url
mio-sito.com/addiziona     // errore
mio-sito.com/addiziona/10 // ok la funzione riceve un parametro
mio-sito.com/addiziona/10/20/30 // ok la funzione riceve tre parametri
```

2. Multipli placeholder **(:any)**:

   CI4 si aspetta di trovare almeno tanti parametri quanti sono i placeholder (:any)

```
$routes->get('/addiziona/(:any)/(:any)', 'Calc::sum($p1,$p2,$p3)');

//url
mio-sito.com/addiziona        // errore
mio-sito.com/addiziona/10     // errore
mio-sito.com/addiziona/10/20  // errore
mio-sito.com/addiziona/10/20/30 // ok la f. riceve tre parametri
```

3. Atri placeholder:

   CI4 si aspetta di trovare tanti parametri quanti sono i placeholders e ognuno dovrà soddisfare le regole del corrispondente placeholdr.

```
$routes->add('/addiziona/(:num)/(:num)', 'Calc::sum($p1,$p2,$p3)');

//url
mio-sito.com/addiziona          // errore
mio-sito.com/addiziona/10       // errore
mio-sito.com/addiziona/10/20    // ok la f. riceve due parametri
mio-sito.com/addiziona/10/20/30 // errore
```

$routes->add('/festa/(:alpha)/(:num)','Diario::scrivi(p1,p2,p3)');

//url mio-sito.com/festa // errore mio-sito.com/festa/marco // errore mio-sito.com/festa/marco/20 // ok la f. riceve due parametri mio-sito.com/festa/10/marco // errore mio-sito.com/festa/marco/20/10 // errore

```
Placeholder vs  Controller::metodo
```