Flutter 2 Start

il più completo videocorso in italiano su Flutter

Fudeo

Stateless e Stateful

02

StatelessWidget

E' il classico componente che abbiamo usato fino ad ora.

03

StatelessWidget

E' un componente senza stato: **non abbiamo variabili che cambiano** all'interno della classe del nostro componente.

04

StatefulWidget

Se vogliamo avere delle **variabili che cambiano**: StatefulWidget.

StatefulWidget

- Click di un pulsante: incrementare variabile.
- Scaricare dati da internet: assegnare valore scaricato a variabile.

06

Esempi di codice

07

StatelessWidget

```
class HomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Text("Valore contatore: 0");
  }
}
```

08

StatefulWidget

```
class HomePage extends StatefulWidget {
  @override
  State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  int counter = 0;

  @override
  Widget build(BuildContext context) {
    return Text("Valore contatore: $counter");
  }
}
```

StatelessWidget

```
class HomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Text("Valore contatore: 0");
  }
}
```

StatefulWidget

```
class HomePage extends StatefulWidget {
  @override
  State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  int counter = 0;

  @override
  Widget build(BuildContext context) {
    return Text("Valore contatore: $counter");
  }
}
```

StatelessWidget

```
class HomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Text("Valore contatore: 0");
  }
}
```

StatefulWidget

```
class HomePage extends StatefulWidget {
  @override
  State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  int counter = 0;

  @override
  Widget build(BuildContext context) {
    return Text("Valore contatore: $counter");
  }
}
```

StatelessWidget

```
class HomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Text("Valore contatore: 0");
  }
}
```

StatefulWidget

```
class HomePage extends StatefulWidget {
  @override
  State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  int counter = 0;

  @override
  Widget build(BuildContext context) {
    return Text("Valore contatore: $counter");
  }
}
```

StatelessWidget

```
class HomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Text("Valore contatore: 0");
  }
}
```

StatefulWidget

```
class HomePage extends StatefulWidget {
  @override
  State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  int counter = 0;

  @override
  Widget build(BuildContext context) {
    return Text("Valore contatore: $counter");
  }
}
```

StatefulWidget

Come funziona.

setState

Un metodo ereditato dalla superclasse (extends StatefulWidget) per notificare il cambiamento di stato.

setState

```
int counter = 0;

void increment() {
    counter = counter + 1;
}
```

setState

```
int counter = 0;

void increment() {
    setState(() {
        counter = counter + 1;
    });
}
```

setState

```
int counter = 0;

void increment() {
    counter = counter + 1;
    setState(() {});
}
```

Virtual DOM

Concetto presente nel mondo Web con ReactJS e AngularJS.

Virtual DOM

- Mantiene in memoria una copia di tutta la grafica.
- Esegue un operazione detta: diffing.
- Aggiorna solamente le parti della grafica che sono effettivamente cambiate.
- Permette di incrementare notevolmente le performance.