# How do I declare a model class in my Angular 2 component using TypeScript?

Asked 3 years, 1 month ago Active 3 months ago Viewed 193k times

70

15

I am new to Angular 2 and TypeScript and I'm trying to follow best practices.

Instead of using a simple JavaScript model ({ }), I'm attempting to create a TypeScript class.

However, Angular 2 doesn't seem to like it.

My code is:

```
import { Component, Input } from "@angular/core";

@Component({
    selector: "testWidget",
    template: "<div>This is a test and {{model.param1}} is my param.</div>"
})

export class testWidget {
    constructor(private model: Model) {}
}

class Model {
    param1: string;
}
```

and I'm using it as:

```
import { testWidget} from "lib/testWidget";

@Component({
    selector: "myComponent",
    template: "<testWidget></testWidget>",
    directives: [testWidget]
})
```

I'm getting an error from Angular:

So I thought, Model isn't defined yet... I'll move it to the top!

Except now I get the exception:

> ORIGINAL EXCEPTION: No provider for Model!

How do I accomplish this??

Edit: Thanks to all for the answer. It led me to the right path.

In order to inject this into the constructor, I need to add it to the providers on the component.

This appears to work:

```
import { Component, Input } from "@angular/core";

class Model {
    param1: string;
}

@Component({
    selector: "testWidget",
    template: "<div>This is a test and {{model.param1}} is my param.</div>",
    providers: [Model]
})

export class testWidget {
    constructor(private model: Model) {}
}
```

class typescript angular

edited Jul 15 '16 at 15:52

asked Jul 15 '16 at 14:38

Scottie

**6,074** 12 53 96

## 7 Answers

I'd try this:

132

Split your Model into a separate file called `model.ts` :

```
export class Model {
    param1: string;
}
```

Import it into your component. This will give you the added benefit of being able to use it in other components:

```
Import { Model } from './model';
```

Initialize in the component:

```
export class testWidget {
   public model: Model;
   constructor(){
       this.model = new Model();
       this.model.param1 = "your string value here";
   }
}
```

Access it appropriately in the html:

```
@Component({
      selector: "testWidget",
      template: "<div>This is a test and {{model.param1}} is my param.</div>"
})
```

I want to add to the answer a comment made by @PatMigliaccio because it's important to adapt to the latest tools and technologies:

> If you are using `angular-cli` you can call `ng g class model` and it will generate it for you. model being replaced with whatever naming you desire.

edited Feb 7 '18 at 18:52

answered Jul 15 '16 at 14:55


Brendon Colburn

1 looks prettier, thanks reto – Brendon Colburn Jul 15 '16 at 15:02

1 Interesting... If I try and do the shorthand of constructor(private model: Model), I get the error saying No Provider. However, if I define it as private model:Model = new Model(), it works. Why is this? – Scottie Jul 15 '16 at 15:15

7 I'm not an Angular 2 architect, but based on my experience with Angular when you bring something in through the constructor you are implying it is injected. Injecting requires that you add it to the @Component as a provider like such: `providers: [Model]` . Also as per the Angular 2 Tour of Hero's demo you should just go with it as a property instead of an injectable as that functionality is typically reserved for more complex classes such as services. – Brendon Colburn Jul 15 '16 at 15:19

1 Problem with your way of instantiating the model (not using `new` ). Is that the contructor of the model won't get called. And that `instanceOf Model` will be false – PierreDuc Jul 15 '16 at 15:31

5 If you are using `angular-cli` you can call `ng g class model` and it will generate it for you. `model` being replaced with whatever naming you desire. – Pat Migliaccio Aug 11 '17 at 15:23

16

The problem lies that you haven't added `Model` to either the `bootstrap` (which will make it a singleton), or to the `providers` array of your component definition:

```
@Component({
    selector: "testWidget",
    template: "<div>This is a test and {{param1}} is my param.</div>",
    providers : [
       Model
    ]
})

export class testWidget {
    constructor(private model: Model) {}
}
```

And yes, you should define `Model` above the `Component` . But better would be to put it in his own file.

But if you want it to be just a class from which you can create multiple instances, you better just use `new` .

```
@Component({
    selector: "testWidget",
    template: "<div>This is a test and {{param1}} is my param.</div>"
})
```

```
    constructor() {}
}
```

edited Jul 15 '16 at 15:01

answered Jul 15 '16 at 14:52

PierreDuc

**36k** 5 71 88

1 Model is just a class, import should work ideally. why we need it in `providers` array? – Pankaj Parkar Jul 15 '16 at 14:57

Yeah but his templating wouldn't work here, it would be model.param1. Also, he hasn't given it an initial value? – Brendon Colburn Jul 15 '16 at 14:57

@PankajParkar Because i still get a `No Provider for Model` if I don't add it to the providers array – PierreDuc Jul 15 '16 at 14:59

@PierreDuc do you have `export` before `Model` class? – Pankaj Parkar Jul 15 '16 at 15:00

@PankajParkar **look here** – PierreDuc Jul 15 '16 at 15:01

6

In your case you are having model on same page, but you have it declared after your Component class, so that's you need to use `forwardRef` to refer to `Class` . **Don't prefer to do this, always have `model` object in separate file.**

```
export class testWidget {
    constructor(@Inject(forwardRef(() => Model)) private service: Model) {}
}
```

Additionally you have to change you view interpolation to refer to correct object

```
{{model?.param1}}
```

Better thing you should do is, you can have your `Model` Class define in different file & then import it as an when you require it by doing. Also have `export` before you class name, so that you can import it.

```
import { Model } from './model';
```

**118k** 16 178 247

You're missing that his template looks wrong. – Brendon Colburn Jul 15 '16 at 15:01

@BrendonColburn: You're right. Fixed. – Scottie Jul 15 '16 at 15:02

@BrendonColburn thanks man, I saw that in your answer. so I thought there is no meaning to edit my answer aftewards, Thanks man for heads up though, cheers :) – Pankaj Parkar Jul 15 '16 at 15:03

1 np, cheers Pankaj :) – Brendon Colburn Jul 15 '16 at 15:08

4

my code is

```
    import { Component } from '@angular/core';

class model {
  username : string;
  password : string;
}

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})



export class AppComponent {

 username : string;
 password : string;
  usermodel = new model();

  login(){
  if(this.usermodel.username == "admin"){
    alert("hi");
  }else{
    alert("bye");
    this.usermodel.username = "";
  }
```

and the html goes like this :

```
<div class="login">
  Usernmae : <input type="text" [(ngModel)]="usermodel.username"/>
  Password : <input type="text" [(ngModel)]="usermodel.password"/>
  <input type="button" value="Click Me" (click)="login()" />
</div>
```

answered Jan 24 '17 at 2:54

Ahmed Basha
**41** 2

---

1

You can use the angular-cli as the comments in @brendon's answer suggest.

You might also want to try:

```
ng g class modelsDirectoy/modelName --type=model

/* will create
 src/app/modelsDirectoy
  ├── modelName.model.ts
  ├── ...
 ...
*/
```

**Bear in mind:** `ng g class` **!==** `ng g c`
However, you can use `ng g cl` as shortcut depending on your angular-cli version.

answered Apr 19 '18 at 17:22

user9869932
**3,061** 2 30 35

---

I realize this is a somewhat older question, but I just wanted to point out that you've add the model variable to your test widget class incorrectly. If you need a Model variable, you shouldn't be trying to pass it in through the component constructor. You are only intended

As an example, your code should really look something like this:

```
import { Component, Input, OnInit } from "@angular/core";
import { YourModelLoadingService } from "../yourModuleRootFolderPath/index"

class Model {
    param1: string;
}

@Component({
    selector: "testWidget",
    template: "<div>This is a test and {{model.param1}} is my param.</div>",
    providers: [ YourModelLoadingService ]
})

export class testWidget implements OnInit {
    @Input() model: Model; //Use this if you want the parent component instantiating
this
        //one to be able to directly set the model's value
    private _model: Model; //Use this if you only want the model to be private within
        //the component along with a service to load the model's value
    constructor(
        private _yourModelLoadingService: YourModelLoadingService //This service should
        //usually be provided at the module level, not the component level
    ) {}

    ngOnInit() {
        this.load();
    }

    private load() {
        //add some code to make your component read only,
        //possibly add a busy spinner on top of your view
        //This is to avoid bugs as well as communicate to the user what's
        //actually going on

        //If using the Input model so the parent scope can set the contents of model,
        //add code an event call back for when model gets set via the parent
        //On event: now that loading is done, disable read only mode and your spinner
        //if you added one

        //If using the service to set the contents of model, add code that calls your
        //service's functions that return the value of model
        //After setting the value of model, disable read only mode and your spinner
        //if you added one. Depending on if you leverage Observables, or other methods
```

A class which is essentially just a struct/model should not be injected, because it means you can only have a single shared instanced of that class within the scope it was provided. In this case, that means a single instance of Model is created by the dependency injector every time testWidget is instantiated. If it were provided at the module level, you would only have a single instance shared among all components and services within that module.

Instead, you should be following standard Object Oriented practices and creating a private model variable as part of the class, and if you need to pass information into that model when you instantiate the instance, that should be handled by a service (injectable) provided by the parent module. This is how both dependency injection and communication is intended to be performed in angular.

Also, as some of the other mentioned, you should be declaring your model classes in a separate file and importing the class.

I would strongly recommend going back to the angular documentation reference and reviewing the basics pages on the various annotations and class types: https://angular.io/guide/architecture

You should pay particular attention to the sections on Modules, Components and Services/Dependency Injection as these are essential to understanding how to use Angular on an architectural level. Angular is a very architecture heavy language because it is so high level. Separation of concerns, dependency injection factories and javascript versioning for browser comparability are mainly handled for you, but you have to use their application architecture correctly or you'll find things don't work as you expect.

answered Apr 23 '18 at 16:46

user2904660
**39** 5

---

0

```
export class Car {
  id: number;
  make: string;
  model: string;
  color: string;
  year: Date;

  constructor(car) {
      {
        this.id = car.id;
        this.make = car.make || '';
        this.model = car.model || '';
        this.color = car.color || '';
        this.year = new Date(car.year).getYear();
```

The || can become super useful for very complex data objects to default data that doesn't exist.

. .

In your component.ts or service.ts file you can deserialize response data into the model:

```
// Import the car model
import { Car } from './car.model.ts';

// If single object
car = new Car(someObject);

// If array of cars
cars = someDataToDeserialize.map(c => new Car(c));
```

edited May 8 at 17:19

answered May 8 at 17:11

Mike Hawes

**257** 2 5