1、写在app.component.html的路由及其它内容的样式在app.component.css中去处理

2、写注册或登录的页面（各自组件的html中）

3、写各自的样式（各自组件的css中）

4、全局样式：写在src下的styles.css文件中

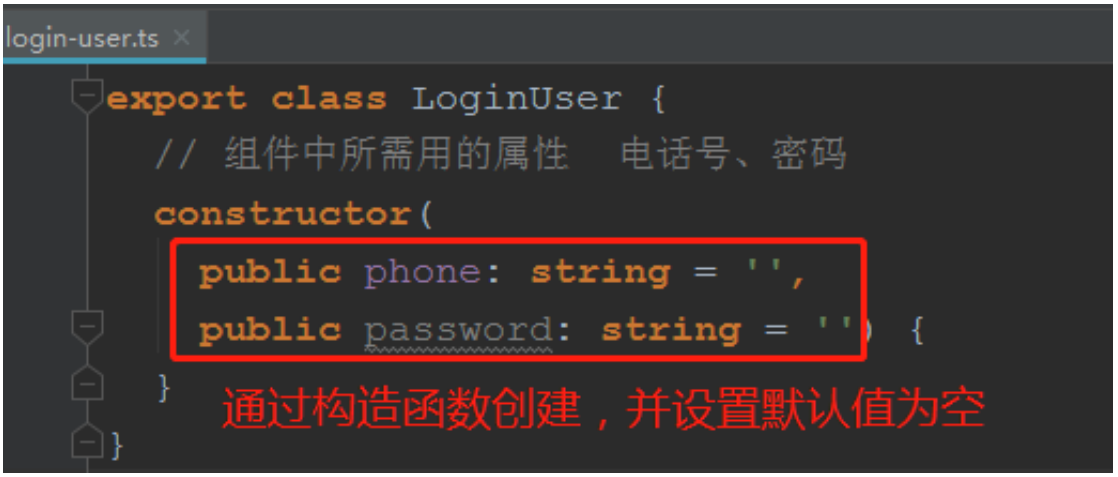
5、创建组件要的数据模型类

5.1、通过命令创建： ng g class 模板名 不指定模板位置，默认就在app目录下

5.2、通过可视化视图创建

app右键new------>Angular Schematic------->选择组件 class-------->回车-------->输处组件名------->回车

5.3、在模板中声明成员属性




6、组件具体的业务处理，在组件的ts文件中，创建模板对象及相应的方法

```ts
ser.ts × login.component.ts ×
@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent {
  // 声明模板对象名
  m: LoginUser;

  constructor() {
    // 构造中创建模板对象
    this.m = new LoginUser();
  }
```

7、组件html中有涉及form提交的事件，在app.module.ts的imports中 导入表单模版驱动

模型驱动表单需要引入模块： ReactiveFormsModule 。

模版驱动表单需要引入 FormsModule

组件的html中有用到ngModel、ngForm、ngSubmit等等

```ts
app.module.ts ×
  imports: [
    BrowserModule,
    FormsModule, // 导入表单模板驱动
```

8、在组件的html中做数据的绑定及逻辑验证

login.component.html

```
<!--(ngSubmit)是模板提交输出-->
<form (ngSubmit)="login()">
  <div class="form-group">
    <!--#phone是模板引用变量-->
    <!--required 空验证 、pattern格式验证 -->
    <!--[(ngModel)] 是模板变量的输入输出绑定-->
    <!--m.phone 是模板变量的对象引用-->
    <input type="text" class="form-control"
           name="phone" placeholder="请输入手机号"
           [(ngModel)]="m.phone" #phone="ngModel"
           required pattern="1[3,5,7,8]\d{9}" >
      <!--#phone是模板引用变量    显示验证的错误信息-->
      <!-- phone.valid || phone.untouched    没有输入过或没有错误隐藏该错误标签-->
      <div class="error" [hidden]="phone.valid || phone.untouched">
        <span *ngIf="phone?.errors?.required">请输入手机号</span>
        <span *ngIf="phone?.errors?.pattern">手机号格式不正确</span>
      </div>
  </div>
```

数据到模板对象的双向输入输出绑定

属性引用绑定

对输入数据的效验规则约定

9、表单提交后的具体业务处理

发送请求：

9.1、配置代理，防止跨域

在项目的根目录创建一个json文件，命名：proxy.conf.json

文件是定义：请求的地址、请求的协议

proxy.conf.json

```
{
  "/api": {
    "target":"http://localhost:3000",
    "secure":false
  }
}
```

地址

协议

9.2、改启动脚本，在package.json配置文件中改

```
"start": "ng serve 改 加上代理文件

"start": "ng serve --proxy-config proxy.conf.json",
```

package.json

```
{
  "name": "user-clis",
  "version": "0.0.0",
  "scripts": {
    "ng": "ng",
    "start": "ng serve --proxy-config proxy.conf.json",
    "build": "ng build",
    "test": "ng test",
```

9.3、导入HttpClient模块才能使用http请求 HttpClientModule

app.module.ts文件的imports里导入 HttpClientModule

```
imports: [
      BrowserModule,
      FormsModule, // 导入表单模板驱动
      HttpClientModule, // 导入HttpClient模块
      RouterModule.forRoot(routes)    // 导入路由配置模块
 ],
```

9.4、请求后台接口一般写入服务中，创建服务，也可以用可视化创建

在app目录创建服务文件，创建命令： ng g s 服务名

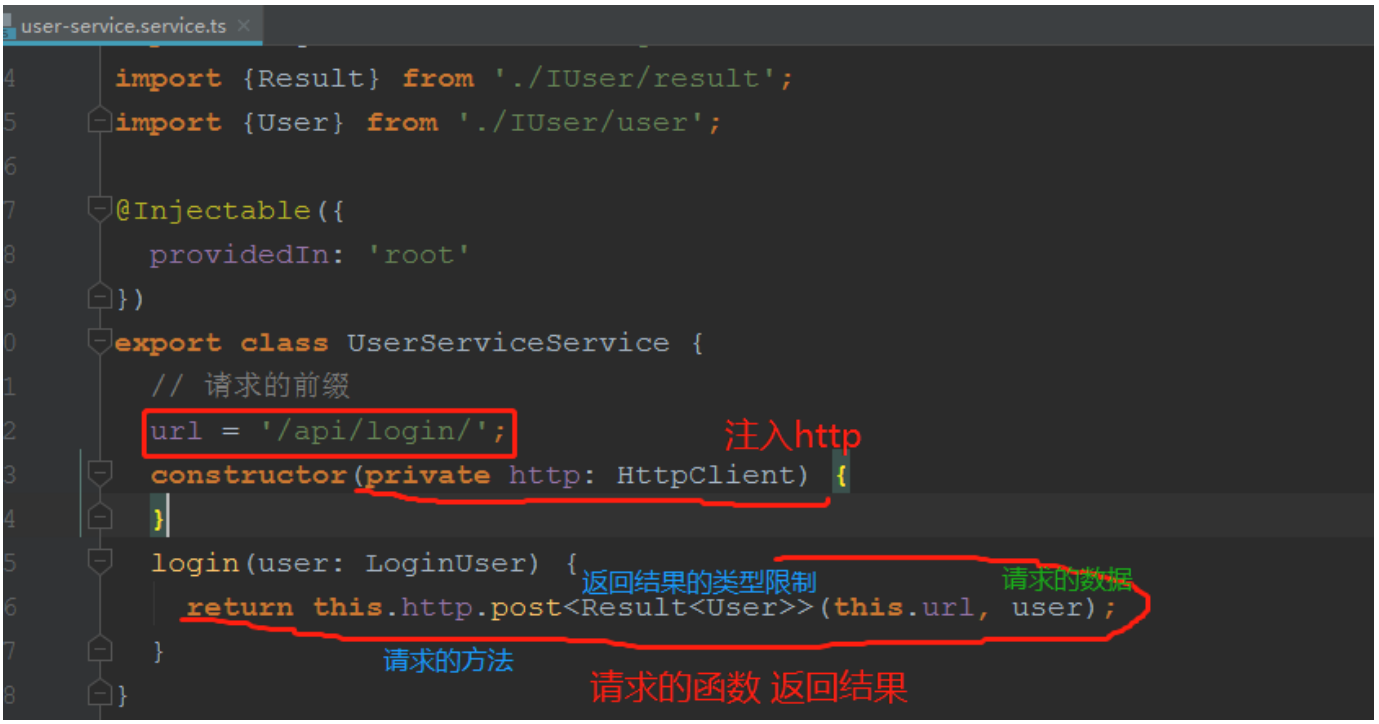
如：ng g s user　默认在app目录下生成服务文件

9.5、在服务中指定请求的前缀，在构造中注入HttpClient，约定请求函数、传入相应的数据及约定返回的数据类型

在服务类中指定请求的前缀（如果需要）：url = '/api/users/';

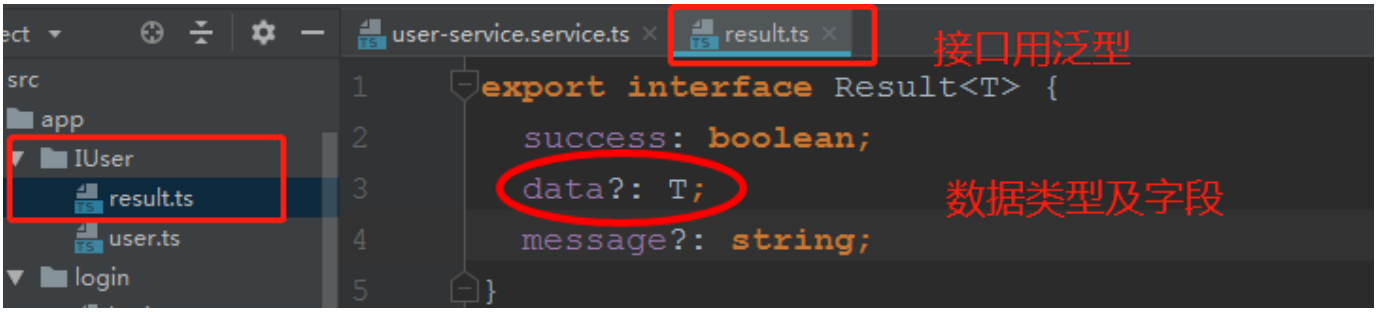
在服务的构造函数中注入HttpClient：private http: HttpClient

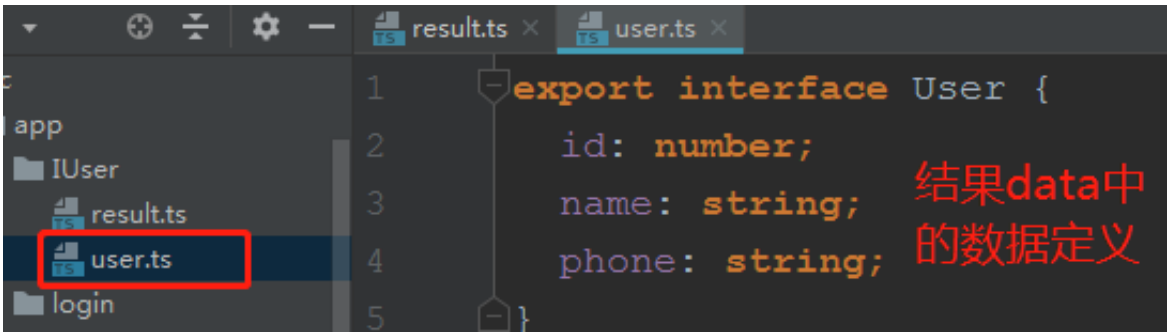
定义请求的方法及参数：return this.http.post<Result<User>>(this.url , user);




9.6、通过接口约定返回函数

定义返回结果的类型限制接口：




9.8、注入网络请求服务调用请求网络方法并处理响应结果

在组件的ts文件中有组件类 在类的构造中注入： `private uslogin: UserServiceService`

login.component.ts

```
13  export class LoginComponent {
14    // 声明模板对象名
15    m: LoginUser;
16    // 注入网络请求的服务
17    constructor(private uslogin: UserServiceService) {
18      // 构造中创建模板对象
19      this.m = new LoginUser();
20    }
21    login() {    // 提交form的方法
22      // 发起网络请求
23      this.uslogin.login(this.m).subscribe(
24        next: (result: Result<User>) => {
25          if (result.success) {
26            alert('登录成功！');
27          } else {
28            alert('登录失败！');
29          }
30        },
31        error: (error) => {
32          console.log(error);
33          alert('登录失败！');
```