

**PET SHOP WEB APPLICATION**

**USING ANGULAR 4 AND TYPESCRIPT**

Submitted by:

**SRIRAM KRISHNAN**

(skrishn5@binghamton.edu)

Under the Guidance of:

**Prof. Leslie Lander**

Department of Computer Science

Thomas J. Watson School of Engineering and Applied Science

**Binghamton University**

**1 |** P a g e

**Table Contents:**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No |  | Page Title | Page No |
| 1 | Introduction |  | 3 |
| 2 | Project Overview |  | 4 |
| 3 | Analysis and Design |  | 5 |
| 4 | Technologies Used |  | 6 |
| 5 | Implementation |  | 7 |
| 6 | Future Scope |  | 12 |
| 7 | Project Summary |  | 13 |
| 8 | Bibliography |  | 14 |

**2 |** P a g e

**1. INTRODUCTION:**

A pet shop web application is a place where user can add pets, edit the details of the

pets and select the pet food appropriate for the pet. It allows the user to send the pet food to the shopping list by clicking on the to shopping list from the drop down. There is a shopping list where the user can add pet food to the page, delete them and clear the text entered. It is a single page web application which is achieved through angular taking care of the routing and component aspects and typescript facilitating business logic.

Bootstrap CSS has been used for styling of the web page and angular CLI to ease out the creation of the web application.

**3 |** P a g e

**2. PROJECT OVERVIEW:**

The main purpose of the application is to add pets to the list, edit and manage the pets.

Main Features:

1. Add Pets.
2. Edit Pets.
3. Add Images.
4. edit pet food.
5. Add pet food to the shopping list.
6. Edit shopping list.
7. Delete Pet food from shopping list.
8. Clear the text in shopping list.
9. Save and Fetch Data.

**4 |** P a g e

**3. ANALYSIS AND DESIGN:**

I had done this project using the new version of Angular and typescript. The reason I chose this JavaScript framework was it's very recently introduced into the market and it’s an updated version of angular 2 thereby providing certain cool features as compared to angular 2. The project has been done using the webstorm IDE. It can also be done using sublime and other editors. I had used angular cli to ease out the coding as it helps in creating templates for components and directives. The code has also been written in such a way that user understands it and if any bugs in the future he can easily debug it.

3.1 Process Model

Process models are processes of the same nature that are classified together into a model. Thus, a process model is a description of a process at the type level. Since the process model is at the type level, a process is an instantiation of it. The same process model is used repeatedly for the development of many applications and thus, has many instantiations.

I have used The **waterfall model** which is a sequential design process, often used in software development processes, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of Conception, Initiation, Analysis, Design, Construction, Testing, Production/Implementation, and Maintenance.

**5 |** P a g e

**4. TECHNOLOGIES USED:**

1. **Hardware Requirements**:
   1. Computer with processor (at least 1.5 Ghz)
   2. Ram – 512 MB (at least)
2. **Software Requirements:**
   1. operating system (Windows/Macos) any version
   2. Angular CLI
   3. NPM
   4. Bootstrap
   5. Webstorm
   6. Firebase account

**6 |** P a g e

**5. IMPLEMENTATION:**

For implementation I used Angular and Typescript.

The Angular project comprises of the following:

1. Components
2. Directives
3. Routing
4. Services
5. Modules
6. Model

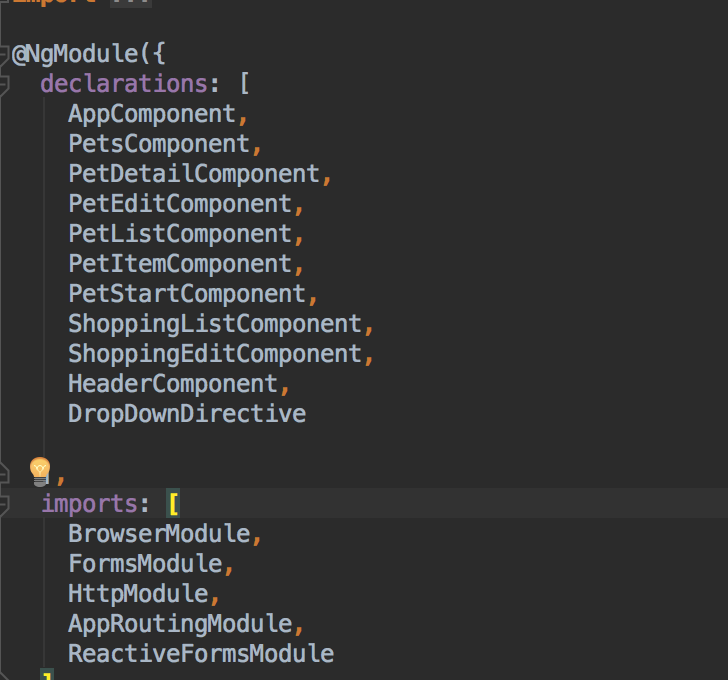
Angular startup:

1. When angular application loads main.ts is the first one to start in which the root module is bootstrapped which is similar to ng-app in angular 1.x.
2. Next it goes to the root module typescript and we have a bootstrap array which tells the root component we are looking for.
3. Angular analyses the root component typescript file and checks the setup. It knows the selector and template url through the component decorator through which it knows which html file should be displayed.
4. Now with content of the root component angular finds index.html and replaces the <app-root> section with the root html file.

The root content for this application is named app.

**Components:**

To add a component the following must be done:

1. Create a typescript class.
2. Import component decorators and add selector, template url and styling url to the component decorator.
3. Then provide the component in the NgModule decorator of the appcomponent.ts file which is the root component. 

The hierarchy of the components is as follows:

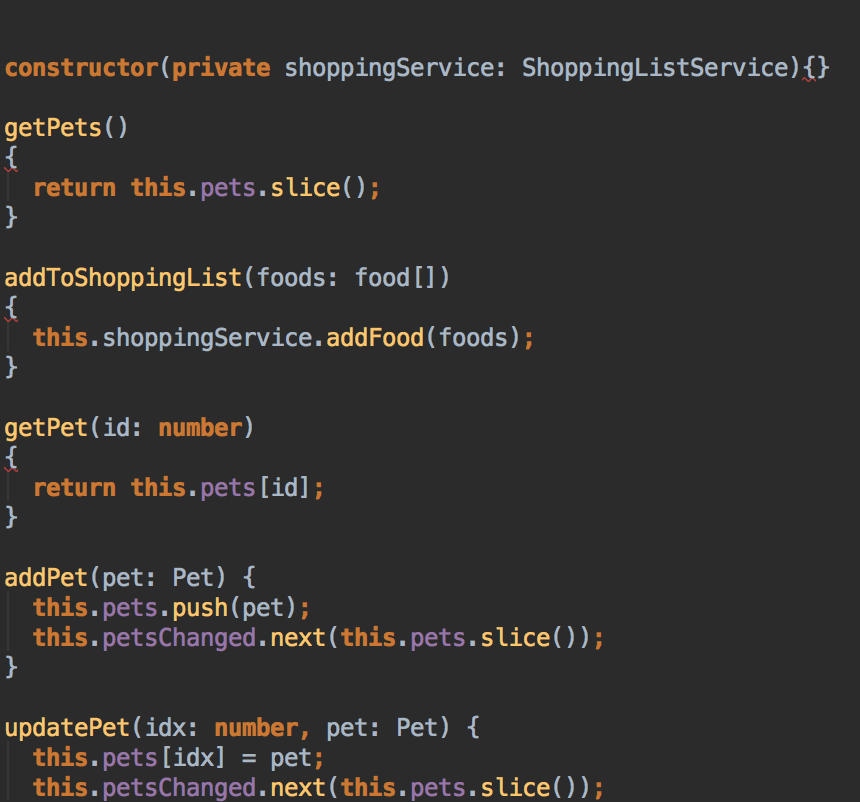
1. App component is the root component.
2. It has three children header, Pet, Shopping List.
3. Pet has 3 children Pet detail, pet edit, pet list.
4. Pet list has a component within it called pet item.
5. Similarly, for shopping list it has one child shopping edit.

Services:

There are two services which takes care of the communication between different components.

petServices.ts:

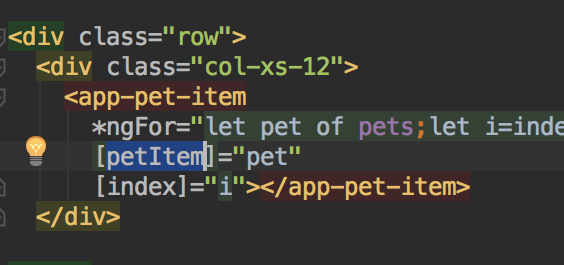




This service class is called through the component typecript class in case of any events for eg click events.We have a subject observable which helps in subscribing in case of any changes in the application content asynchronously.

Another service that had been created is for shopping list which serves almost the same purpose.

We can create event emitters in case we have to propogate the data from a child to parent and provide an @Output() decorator with variable and list to the event emitted in the parent html files and add the business logic to the parents typescript file.Similary we can also provide input from the parent to the child by using @input() in the child which is the receiving end and providing the values in the parent and binding it to the property of the child.



In the above code you can see the attribute [pet-item] which is the input to the app-pet item component.

You can find the input properties in the below typescript code of the pet-item component.

Pet-item.component.ts



Model:

There were 2 model classes created for loading the data this can be extended when there are more functionalites introduced and large amount of data has to be used so instead of providing the data in the same ts file we can create a model class which segregates the data from the business logic.

