TypeScript Generics y su funcionamiento

27 abril, 2018 por Cecilio Álvarez Caules — Deja un comentario

Crear **TypeScript Generics** poco a poco se convertirá en algo **muy común**. El lenguaje cada día tiene mayor tracción **y el uso de Genéricos en cualquier lenguage compilado es muy importante**. Vamos a ver un ejemplo sencillo de como manejar clases genéricas utilizando Typescript. Para ello partiremos de dos clases muy sencillas **Galleta y Golosina** :

```
export class Galleta{

  sabor:string;
  constructor(sabor:string) {

    this.sabor=sabor;

  }

}
```

```
export class Golosina {
```

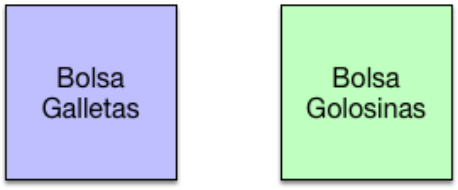
```
  }

}
```

Como podemos ver ambas clases son muy parecidas. Vamos a crear ahora un par de clases que se denominen **BolsaGalleta y BolsaGolosina** que nos permitan almacenar Galletas y Golosinas.

```
import {Galleta} from "./Galleta";
export class BolsaGalletas {

  lista:Array<Galleta>= new Array<Galleta>();


  add(galleta:Galleta) {

    this.lista.push(galleta);
  }


  remove(galleta:Galleta) {

    var index = this.lista.indexOf(galleta, 0);
    if (index > -1) {
     this.lista.splice(index, 1);
  }
}

  forEach(fn:(value: Galleta, index: number, array: Galleta[])=>void):void {

    this.lista.forEach(fn);
  }

}
```

```
  add(golosina:Golosina) {

    this.lista.push(golosina);
  }

  remove(golosina:Golosina) {

    var index = this.lista.indexOf(golosina, 0);
    if (index > -1) {
     this.lista.splice(index, 1);
  }
}

  forEach(fn:(value: Golosina, index: number, array: Golosina[])=>void):void {

    this.lista.forEach(fn);
  }
}
```

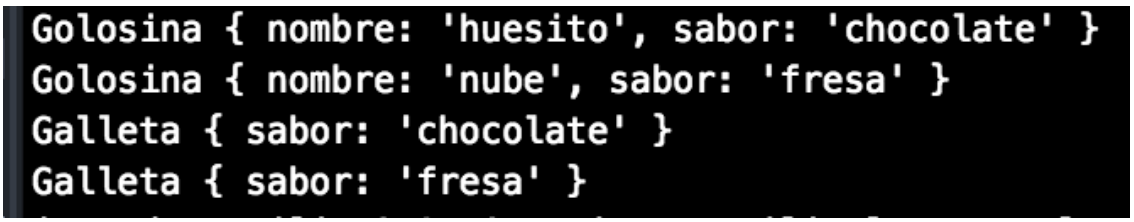
Es más que evidente que las bolsas son muy parecidas y que **únicamente cambia el tipo de la bolsa.**



Vamos a ver el código del programa main que se encargaría de crear estas bolsas y añadir elementos a ellas.

```
let g1= new Golosina("huesito","chocolate");
let g2= new Golosina("nube","fresa");

bolsaGolosinas.add(g1);
bolsaGolosinas.add(g2);


bolsaGolosinas.forEach(function(elemento) {

  console.log(elemento);
});

let bolsaGalletas= new BolsaGalletas();
let galleta1= new Galleta("chocolate");
let galleta2= new Galleta("fresa");

bolsaGalletas.add(galleta1);
bolsaGalletas.add(galleta2);


bolsaGalletas.forEach(function(elemento) {

  console.log(elemento);
});
```

Ejecutamos con node:



TypeScript Generics

```
  lista:Array<T>= new Array<T>();


  add(elemento:T) {

    this.lista.push(elemento);
  }

  remove(elemento:T) {

    var index = this.lista.indexOf(elemento, 0);
    if (index > -1) {
     this.lista.splice(index, 1);
  }
  }


  forEach(fn:(value: T, index: number, array: T[])=>void):void {

    this.lista.forEach(fn);
  }

}
```

TypeScript Generics y Main

Es momento de utilizar esta clase genérica en nuestro programa Main para que veamos como utilizarla:

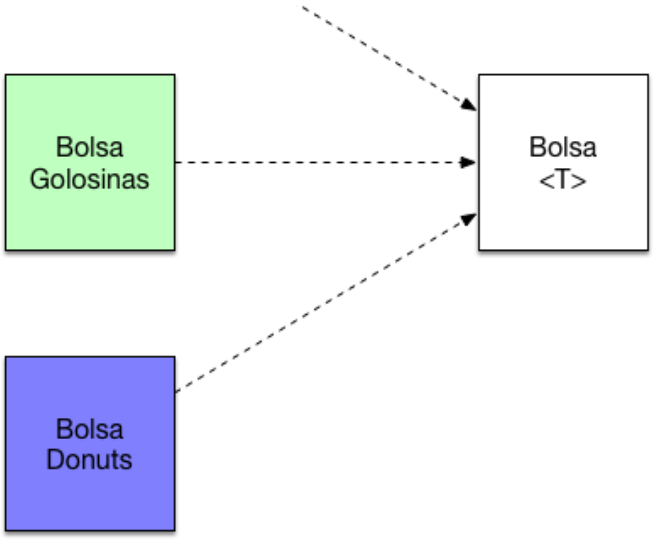
```
import {Bolsa} from "./Bolsa";
import {Golosina} from "./Golosina";
import {Galleta} from "./Galleta";
let bolsa= new Bolsa<Golosina>();

let g1= new Golosina("huesito","chocolate");
```

```
  console.log(elemento.nombre);
  console.log(elemento.sabor);
});


let bolsa2= new Bolsa<Galleta>();

let galleta1= new Galleta("chocolate");
let galleta2= new Galleta("fresa");
bolsa2.add(galleta1);
bolsa2.add(galleta2);

bolsa.forEach(function(elemento) {


  console.log(elemento.sabor);
});
```

El resultado será idéntico :

```
Golosina { nombre: 'huesito', sabor: 'chocolate' }
Golosina { nombre: 'nube', sabor: 'fresa' }
Galleta { sabor: 'chocolate' }
Galleta { sabor: 'fresa' }
```

Acabamos de simplificar nuestro código utilizando **TypeScript Generics** .



Otros artículos relacionados:

1. 10 Características que me gustan de TypeScript
2. 10 Atom plugins imprescindibles
3. Angular 5 Hello World y su funcionamiento
4. Angular async pipe y observables

Externos:

1. TypeScript

PDF

Archivada en: JavaScript Conceptos Avanzados

Subscribe

BUSCAR

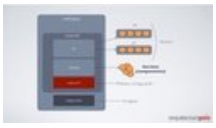
Buscar en este sitio ...

Mis Cursos de Java Gratuitos

Java Herencia

Java JDBC

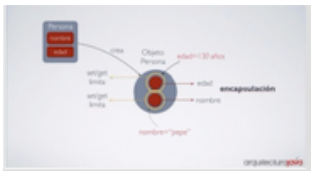
Servlets



Introduccion JPA

Mis Cursos de Java

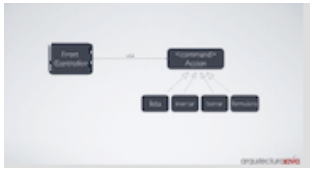
Programación Orientada a Objeto en Java



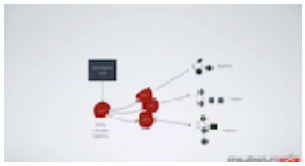
Java APIS Core

Java Web

Pack Java Core



Arquitectura Java Solida con Spring



POPULAR

Spring REST Client con RestTemplates

Nuevo Curso:Arquitectura Java Sólida con Spring 4.3 y Anotaciones

Angular 5 Hello World y su funcionamiento

Java 8 Lambda Syntax ,simplificando nuestro código

Spring REST Test utilizando Rest Assured

Java 9 Collections y sus novedades

Spring @Qualifier utilizando @Autowired

CONTACTO

contacto@arquitecturajava.com

LO MAS LEIDO

¿Qué es Spring Boot?

Java Constructores this() y super()

Usando Java Session en aplicaciones web

Java Iterator vs ForEach

Introducción a Servicios REST

¿Cuales son las certificaciones Java?

Java Override y encapsulación

Ejemplo de Java Singleton (Patrones y ClassLoaders)

¿Qué es Gradle?

REST JSON y Java

Usando el patron factory

Ejemplo de JPA , Introducción (I)

Uso de Java Generics (I)

¿Qué es un Microservicio?

Comparando java == vs equals

Mis Libros

Arquitecturas REST y sus niveles

Spring MVC Configuración (I)