PEC 3

TypeScript

Universitat Oberta de Catalunya

Información relevante:

- Fecha límite de entrega: 14 de abril.
- Peso en la nota de FC: 15%.

Contenido

Información docente

Presentación

Esta práctica se centra en desarrollar aplicaciones usando TypeScript sin la necesidad de utilizar un framework. Se crearán códigos en TypeScript que serán transpilados a JavaScript.

Objetivos

Los objetivos que se desean lograr con el desarrollo de esta PEC son:

- **Desarrollar** código TypeScript.
- Comprender los **tipos de datos de TypeScript.**
- **Utilizar el patrón de arquitectura MVC** en el desarrollo de una aplicación Web.
- Comprender la complejidad del **transpilado** de una aplicación **TypeScript hacia JavaScript.**

Enunciado

Esta PEC contiene 4 ejercicios evaluables. Debes entregar vuestra solución de los 4 ejercicios evaluables (ver el último apartado).

Debido a que las actividades están encadenadas (i.e. para hacer una se debe haber comprendido la anterior), **es altamente recomendable hacer las tareas y ejercicios en el orden en que aparecen en este enunciado**.

Antes de continuar debes:

Haber leído el recurso teórico `T01.PEC3_Teoria_2020.pdf` disponible en el apartado "Contenidos y recursos" del aula de esta PEC.

Ejercicio 1 – Configuración de TypeScript (0.5 puntos)

Mira los 3 últimos vídeos del apartado "Setting up environment" del curso "Introduction to TypeScript" de Tamas Piros.

Te pasamos el enlace al primer vídeo que recomendamos ver:

https://learning.oreilly.com/videos/introduction-to-typescript/10000DIHV201907/10000DIHV201907-piros_u02m02

En este ejercicio debes entregar los ficheros resultado de las tareas realizadas en el video (first.ts, first.js, y tsconfig.json), es posible que alguna de las opciones del tsconfig.json que se indican en el video ya no funcionen en tu versión de TypeScript (te lo indicara el Visual Studio Code o tu Editor) , averigua cual es el problema y corrígelo.

Ejercicio 2 – Primeros códigos en TypeScript (3 puntos)

Antes de continuar debes:

Descargar fichero **`PEC3_Ej2_Primeros_codigos_TS.zip`** adjunto en la PEC

1. (0.5 puntos) Programa sobre el fichero `code1.ts` realizando las siguientes tareas, deja comentarios en el código fuente sobre tus acciones:
   - Sitúate sobre las variables `a, b, c y d` y observa cómo TypeScript infiere el tipo de todas las variables por ti, de tal modo que a es un número, b es un número, c es un objeto con una específica forma, y d es también un número.
   - Modifica el código para conseguir que aparezca una línea roja de error en el IDE avisándote de que se está disparando un `TypeError`. Toma una captura de pantalla de tu resultado y entrégala dentro de un fichero texto (`Word, Pdf, …`) que tenga como nombre `PEC3_Ej2_respuestas_teoria`. Dentro de este mismo documento explica por qué se ha producido esto y qué ventajas tiene.

En el mismo documento PEC3_Ej2_respuestas_teoria contesta las siguientes cuestiones:

2. (1 punto) Para cada uno de los valores del fichero `code2.ts`, ¿Qué tipo de datos inferirá TypeScript? Explica por qué se ha inferido este tipo de datos.
3. (1 punto) ¿Por qué se dispara cada uno de los errores del fichero `code3.ts`?
4. (0.5 puntos) ¿Cuál es la diferencia entre una clase y una interface en TypeScript?

Ejercicio 3 – Modelo de clases (2.5 puntos)

Antes de continuar debéis:

Descargar el fichero **PEC2_Ej3_ModelUML.jpeg** adjunto en la PEC

Este fichero es un diagrama de clases UML que representa un software de gestión deportiva siguiendo el paradigma de programación orientado a objetos. En el diagrama verás clases con herencia, agregación, composición y tipos enumerados.

En este ejercicio te pedimos que crees los ficheros necesarios para modelar las clases están definidas en el diagrama de clases. **NO SE DEBE PROGRAMAR EL CÓDIGO DE LOS MÉTODOS/FUNCIONES**, sólo se deben definir las clases con sus atributos y métodos/constructores (pero sin código), los enum, etc, en TypeScript.

Debido a que TypeScript avisará de que se produce error en caso de que no se retorne el tipo que se especifica en la firma de un método, deberás retornar un atributo de dicho valor en cada una de las funciones/métodos. Por ejemplo, el siguiente método retorna un objeto del tipo `Author`:

```
createAuthor(name: String): Author {
  return {} as Author;
}
```

Ejercicio 4 – Arquitectura MVC usando TypeScript (4 puntos)

En este ejercicio vamos a transformar la aplicación TODO que te facilitamos en JavaScript en la PEC2 a TypeScript. Por tanto, todos los ficheros que componen nuestra aplicación son:

- `todo.model.js` – Los atributos (el modelo) de una tarea.
- `todo.controller.js` – El encargado de unir al servicio y la vista.
- `todo.service.js` – Gestiona todas las operaciones sobre los TODOs.
- `todo.views.js` – Encargado de refrescar y cambiar la pantalla de visualización.

Así pues, los ficheros anteriores deben transformarse a sus versiones TypeScript .

Toda la aplicación se renderizará en un simple nodo de html denominado `root` el cual deberá cargar un único fichero JavaScript (`bundle.js`) que será el que se genere tras realizar la transpilación.

Para realizar esta transformación debes:
- Leer el recurso **`P02.PEC3_Arquitectura_MVC_TS.pdf`** que te guiara en cómo debes realizar este proceso.
- Descargar fichero **`PEC3_Ej4_Aplicacion_TODO.zip`** adjunto en la PEC

Una vez leído el documento y descargado el zip, realiza las siguientes acciones:

a) (0.50 puntos) Construye las clases relativas a modelos, controladores, servicios, vistas y lanzador de la aplicación desde donde irás desarrollando la aplicación. En este punto sólo debes crear la estructura de ficheros que modelan nuestro problema. Es decir, organizar las clases relativas a modelos (`todo.model.ts`), controladores (`todo.controller.ts`), servicios (`todo.service.ts`) y lanzadora (`app.ts`). (comprueba que se transpila correctamente todo y que todas las partes están conectadas).

b) (0.75 puntos) Codifica completamente la clase modelo (anémico) que sea necesaria para esta aplicación utilizando TypeScript, genera todos los interfaces y clases que requieras.

c) (0.75 puntos) Codifica completamente la clase servicio que es la encargada de realizar todas las operaciones sobre una estructura de datos.

d) (0.75 puntos) Codifica completamente la clase vista que controlará todas las operaciones relativas a la vista.

e) (0.75 puntos) Codifica completamente el controlador que es el encargado de poner en comunicación la vista con el servicio, en este proyecto.

g) (0.50 puntos) Configura webpack para que transpile la aplicación completa a un único fichero JavaScript (`bundle.js`). Para ello se ha facilitado dos ficheros webpack de ejemplos (se tienen que modificar y adaptar a nuestro proyecto).

Puedes utilizar la siguiente referencia para comprender el funcionamiento de la configuración de webpack:

https://learning.oreilly.com/videos/introduction-to-typescript/10000DIHV201907/10000DIHV201907-piros_u08m01

Formato y fecha de entrega

Tienes que entregar un fichero *.zip, cuyo nombre tiene que seguir este patrón: `loginUOC_PEC3.zip`. Por ejemplo: `dgarciaso_PEC3.zip`. Este fichero comprimido tiene que incluir los siguientes elementos:

- Una carpeta `PEC3_Ej1` Con los ficheros resultado de las tareas realizadas en el video del Ejercicio1 (first.ts, first.js, y tsconfig.json),
- Una carpeta `PEC3_Ej2` con:
    - El fichero `code1.ts` modificado siguiendo las peticiones y especificaciones del Ejercicio 2.
    - Un fichero texto (`Word, Pdf, ...`) con nombre `PEC3_Ej2_respuestas_teoria` que contenga la captura de pantalla y respuestas a todos los enunciados planteados en el Ejercicio2.
- Una carpeta `PEC3_Ej3` con todos los ficheros de las clases TypeScript que resulten de la implementación del diagrama UML.
- Una carpeta `PEC3_Ej4` Con los ficheros resultado de haber realizado las tareas del Ejercicio 4

El último día para entregar esta PEC es el **14 de abril de 2021** hasta las **23:59**. Cualquier PEC entregada más tarde será considerada como no presentada.