.NET Core -ACCENTURE- 26MAY21

Avisos

ACTIVIDAD 8 - IMPLEMENTAR EL POSICIONAR BARCOS

Su progreso

Video actividad 8

email task8

web-mostrar-juego-3

Enviar (Posting) datos complejos con JSON

Las aplicaciones basadas en JSON, como las aplicaciones javascript comunes, pueden usar AJAX para publicar un objeto JSON que se parezca al objeto C# que desea enviar, y el código del framework en el back-end construirá el objeto C# basado en esa representación.

Por ejemplo, un sitio web de seguimiento de mascotas puede tener un formulario en el que un usuario que ha iniciado sesión puede agregar una nueva mascota para realizar un seguimiento. La parte clave del código para enviar los datos se vería así:

```
$.post({
  url: "api/owners/23/pets",
  data: JSON.stringify({ name: petName, type: petType, age: petAge, allergies: [{name:AllergiName1},
{name:AllergiName2}, {name:AllergiName3}] }),
  dataType: "text",
  contentType: "application/json"
})
.done(function (response, status, jqXHR) {
  alert( "Pet added: " + response );
})
.fail(function (jqXHR, status, httpError) {
  alert("Failed to add pet: " + textStatus + " " + httpError);
})
```

El controlador para manejar esta solicitud podría verse así, dejando de lado la verificación de errores y la creación del código de estado:

```
// GET: api/owners/{ownerId}/pets
[HttpPost("{ownerId}/pets")]
public IActionResult Post(int ownerId, [FromBody] PetDTO pet)
{
    Person person = _personRepository.FindById(ownerId);
    pet.OwnerId = person.Id;
    _petRepository.Save(pet);
    ...
}
```

Este ejemplo obtiene la data de dos lugares diferentes:

- ***HttpPost("{ownerId}/pets")***, le indica al framework que debe obtener el parámetro ownerId de la URL.
- ***[FromBody] PetDTO pet***, le indica al framework que transforme el cuerpo de la petición en un JSON y que utilize ese objeto para crear una instancia de PetDTO, por lo que PetDTO debe tener los campos:

name, type, age y allergies; éste último un listado de tipo AllergieDTO con el campo: Name.

Implementar posicionar barcos

Implementar un método en el controlador GamePlayersController que pueda recibir una lista de objetos de barcos, con ubicaciones, guardarlos en la base de datos y devolver el estado 201 Created si no hay problemas.

- La url deberá ser /api/gamePlayers/{id}/ships
- El cuerpo de la solicitud debe poder transformarse en una lista de Ship
- Se debe responder con código de estado 403 Forbidden si:
  - No existe el gamePlayer con el id indicado, "No existe el juego"
  - El usurio autenticado no se encuentra en el juego al que se quieren agregar los barcos, "El usuario no se encuentra en el juego"
  - Si ya se han posicionado los barcos, "Ya se han posicionado los barcos"
- Si no existen problemas insertar los barcos en el objeto gamePlayer del id indicado y actualizar el objeto gamePlayer.

Importante: ya que el método Save del repositorio GamePlayerRepository intenta guardar una nueva instancia se deberá modificar para que verifique si la instancia se debe actualizar:

```
public void Save(GamePlayer gamePlayer)
{
   if (gamePlayer.Id == 0)
      Create(gamePlayer);
   else
      Update(gamePlayer);
   SaveChanges();
}
```

Además, se debe tener en cuenta que se debe crear el método findByID en GamePlayersRepository incluyendo player y ships ya que se necesita parr obtener el GamePlayer indicado en la petición.

Nota: recuerda que para recibir y enviar los objetos se utilizan los DTO ya que son Objetos de Tansferencia de Datos.

Para probar el controller se puede usar:

```
axios.post("https://localhost:5001/api/gamePlayers/23/ships",
[{"id":0,"type":"PatroalBoat","locations":[{"id":0,"location":"A1"},{"id":0,"location":"A2"}]},
{"id":0,"type":"Destroyer","locations":[{"id":0,"location":"B1"},{"id":0,"location":"B2"},{"id":0,"location":"B3"}]},
{"id":0,"type":"Submarine","locations":[{"id":0,"location":"C1"},{"id":0,"location":"C2"},
{"id":0,"location":"C3"}]},{"id":0,"type":"BattleShip","locations":[{"id":0,"location":"D1"},
{"id":0,"location":"D2"},{"id":0,"location":"D3"},{"id":0,"location":"D4"}]},{"id":0,"type":"Carrier","locations":
[{"id":0,"location":"E1"},{"id":0,"location":"E2"},{"id":0,"location":"E3"},{"id":0,"location":"E4"},
{"id":0,"location":"E5"}]}]) .then(response=>{console.log("fichas posicionadas!!")}) .catch(error =>
{console.log("error, código de estatus: " + error.response.status + " mensaje: " +
error.response.data)});
```

- Si el usuario no se encuentra en el juego mostrará: "error, código de estatus: 403 mensaje: El usuario no se encuentra en el juego"
- Si ya se han posicionado los barcos mostrará: "error, código de estatus: 403 mensaje: Ya se han posicionado los barcos"
- Si no existe un gamePlayer con el id indicado mostrará: "error, código de estatus: 403 mensaje: No existe el juego"
- Si se guardaron los barcso indicará: barcos posicionados!!

Teniendo en cuenta que 23 será reemplazado por un id de un gamePlayer en el sistema.

Al terminar descargar el paquete web-mostrar-juego-3 y copiar su contenido en la carpeta wwwroot. Una vez copiado el contenido en la carpeta ejecutar el proyecto y colocar en el navegador la url *https://localhost:5001/index.html,* una vez allí iniciar sesión ir a un juego y se podrán posicionar los barcos, probar el cambiar de posición arrastándolos y haciendo click sobre los mismos para rotarlos.

Implementar posicionar barcos - Envía el proyecto en un .zip

Mantente en contacto

MindHub

Resumen de retención de datos

einiciar tour para usuario en esta página