Currently the sample app exposes the entire TodoItem object.

Production apps typically limit the data that's input and returned using a subset of the model.

There are multiple reasons behind this, and security is a major one.

The subset of a model is usually referred to as a Data Transfer Object (DTO), input model, or view model.

A DTO may be used to:

- Prevent over-posting.
- Hide properties that clients are not supposed to view.
- Omit some properties in order to reduce payload size.
- Flatten object graphs that contain nested objects. Flattened object graphs can be more convenient for clients.

```
namespace ToDoApi.Models
{
    6 references
    public class TodoItem
    {
        3 references
        public long Id { get; set; }
        0 references
        public string? Name { get; set; }
        0 references
        public bool IsComplete { get; set; }
        0 references
        public string? Secret { get; set; }
    }
}
```

```
namespace ToDoApi.Models
{
    0 references
    public class TodoItemDTO
    {
        0 references
        public long Id { get; set; }
        0 references
        public string? Name { get; set; }
        0 references
        public bool IsComplete { get; set; }
    }
}
```

```
// GET: api/TodoItems
[HttpGet]
0 references
public async Task<ActionResult<IEnumerable<TodoItemDTO>>> GetTodoItems()
{
  if (_context.TodoItems == null)
  {
      return NotFound();
  }
    return await _context.TodoItems.Select(x=> ItemToDTO(x)).ToListAsync();
}

2 references
private static TodoItemDTO ItemToDTO(TodoItem x)
{
    return new TodoItemDTO
    {
        Id = x.Id,
        Name = x.Name,
        IsComplete = x.IsComplete
    };
}
```

```
// GET: api/TodoItems/5
[HttpGet("{id}")]
1 reference
public async Task<ActionResult<TodoItemDTO>> GetTodoItem(long id)
{
  if (_context.TodoItems == null)
  {
      return NotFound();
  }
    var todoItem = await _context.TodoItems.FindAsync(id);

    if (todoItem == null)
    {
        return NotFound();
    }

    return ItemToDTO(todoItem);
}
```

```
// PUT: api/TodoItems/5
[HttpPut("{id}")]
0 references
public async Task<IActionResult> PutTodoItem(long id, TodoItemDTO todoDTO)
{
    if (id != todoDTO.Id)  {
        return BadRequest();
    }

    var todoItem = await _context.TodoItems.FindAsync(id);

    if (todoItem == null)
    {
        return NotFound();
    }

    todoItem.Name = todoDTO.Name;
    todoItem.IsComplete = todoDTO.IsComplete;

    try {
        await _context.SaveChangesAsync();
    }
    catch (DbUpdateConcurrencyException) when (!TodoItemExists(id))
    {
        return NotFound();
    }
    return NoContent();
}
```

```
// POST: api/TodoItems
[HttpPost]
0 references
public async Task<ActionResult<TodoItem>> PostTodoItem(TodoItemDTO todoDTO)
{
    var todoItem = new TodoItem()
    {
        Id = todoDTO.Id,
        Name = todoDTO.Name,
        IsComplete = todoDTO.IsComplete
    };
    _context.TodoItems.Add(todoItem);
    await _context.SaveChangesAsync();

    return CreatedAtAction(nameof(GetTodoItem), new { id = todoItem.Id }, ItemToDTO(todoItem));
}
```

POST /api/TodoItems

Parameters

Cancel Reset

No parameters

Request body application/json

```
{
  "id": 1,
  "name": "Prepare Notes",
  "isComplete": false
}
```

Server response

Code	Details

201
Undocumented
	Response body
	`{ "id": 1, "name": "Prepare Notes", "isComplete": false }`
	Download
	Response headers
	`content-type: application/json; charset=utf-8`
	`date: Sun,22 Jan 2023 22:41:01 GMT`
	`location: https://localhost:7051/api/TodoItems/1`
	`server: Kestrel`
	`x-firefox-spdy: h2`

POST /api/TodoItems

Parameters

Cancel Reset

No parameters

Request body application/json

```
{
  "id": 1,
  "name": "Prepare Notes",
  "isComplete": false.
  "secret": "my secret"
}
```

Server response

Code	Details
400 *Undocumented*	Error: Bad Request

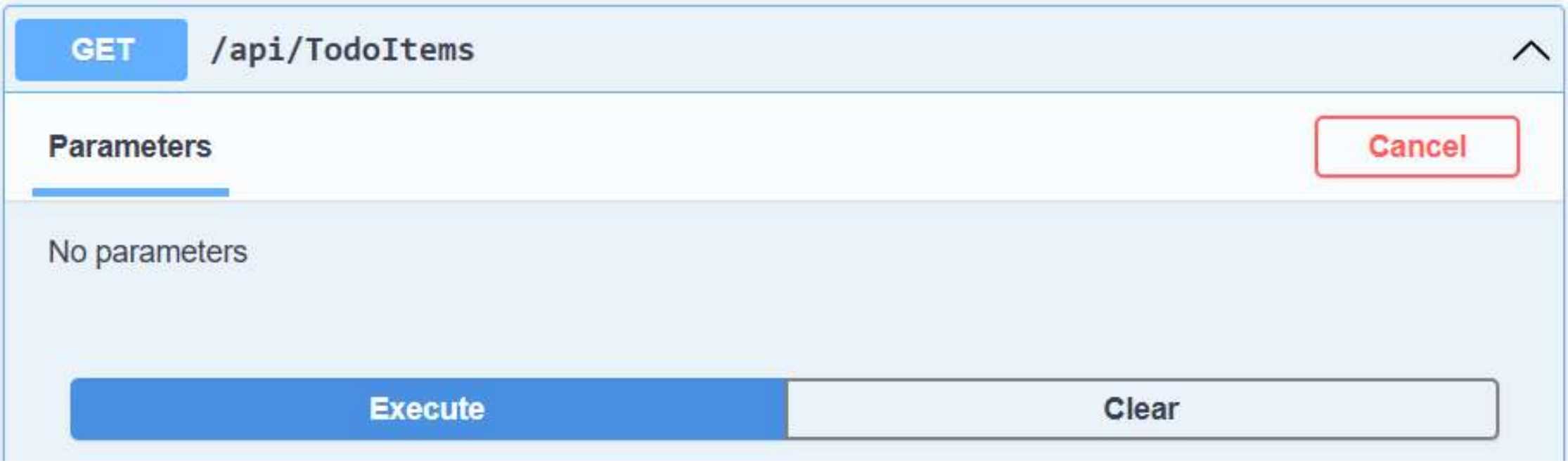
Response body


```
{
  "type": "https://tools.ietf.org/html/rfc7231#section-6.5.1",
  "title": "One or more validation errors occurred.",
  "status": 400,
  "traceId": "00-effe374fce1186383016233a28178f21-bf65fb6eda453186-00",
  "errors": {
    "$": [
      "'.' is invalid after a value. Expected either ',', '}', or ']'. Path: $ | LineNumbe
r: 3 | BytePositionInLine: 21."
    ],
    "todoDTO": [
      "The todoDTO field is required."
    ]
  }
}
```


Download

Response headers

```
content-type: application/problem+json; charset=utf-8
date: Sun,22 Jan 2023 22:43:23 GMT
server: Kestrel
x-firefox-spdy: h2
```


GET
/api/TodoItems
Parameters
Cancel
No parameters
Execute
Clear

Server response

Code	Details
200	Response body

```
[
  {
    "id": 1,
    "name": "Prepare Notes",
    "isComplete": false
  }
]
```

Download

Response headers

```
content-type: application/json; charset=utf-8
date: Sun,22 Jan 2023 22:46:24 GMT
server: Kestrel
x-firefox-spdy: h2
```