



SAKARYA
ÜNİVERSİTESİ

BIG DATA

TOO BIG TO IGNORE

SÜMEYYE KAYNAK

OUTLINE



NoSQL

MongoDB

NOSQL

- The increase in the volume and diversity of data makes the relational database concept inadequate.
- In relational database systems, data is kept in tables and columns.
- In NoSQL database systems, data is kept in Json format.
- Read and write operations in NoSQL databases are faster than relational databases.
- NoSQL systems can scale horizontally.

MONGODB

- MongoDB is an open-source NoSQL database developed in 2009.
- There are a lot of NoSQL database today like Cassandra, BigTable, Dynamo.
- In MongoDB, each record is expressed as a document. The document is stored in Json format.
- MongoDB is one of the most preferred NoSQL databases. Because it has driver support for many programming languages.

SHARDING

- Sharding is a method for distributing data across multiple machines.
- MongoDB uses sharding to support deployments with very large data sets and high throughput operations.
- There are two methods for addressing system growth: vertical and horizontal scaling.

SHARDING

- Vertical Scaling involves increasing the capacity of a single server, such as using a more powerful CPU, adding more RAM, or increasing the amount of storage space.
- Limitations in available technology may restrict a single machine from being sufficiently powerful for a given workload.

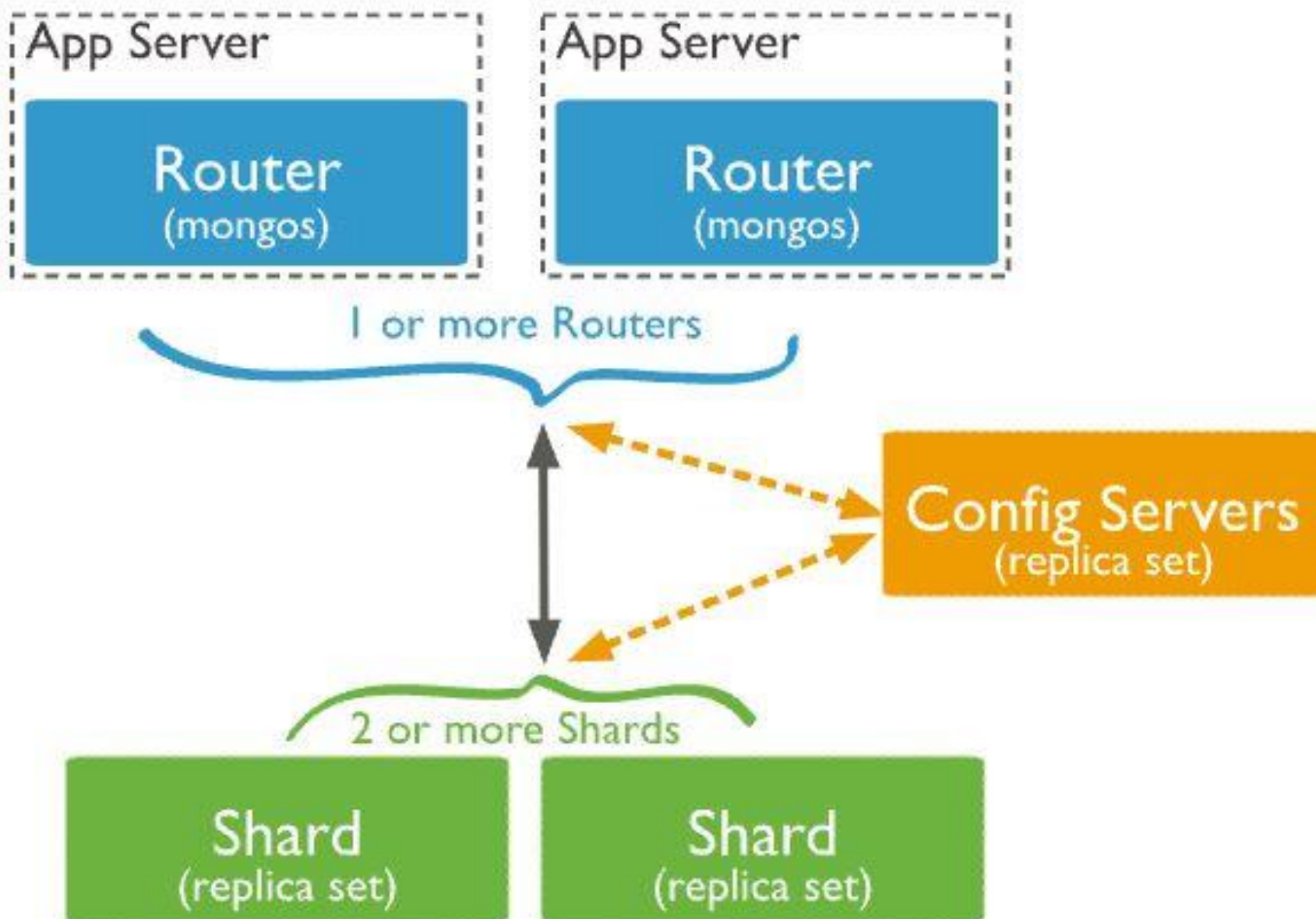
SHARDING

- Horizontal Scaling involves dividing the system dataset and load over multiple servers, adding additional servers to increase capacity as required.
- MongoDB supports horizontal scaling through sharding.

SHARDED CLUSTER

A MongoDB sharded cluster consists of the following components:

- **Shard:** Each shard contains a subset of the sharded data. Each shard can be deployed as a replica set.
- **Mongos:** The mongos acts as a query router, providing an interface between client applications and the sharded cluster.
- **Config servers:** Config servers store metadata and configuration settings for the cluster.



ADVANTAGES OF SHARDING

- **Reads/Writes:** MongoDB distributes the read and write workload across the shards in the sharded cluster, allowing each shard to process a subset of cluster operations. Both read and write workloads can be scaled horizontally across the cluster by adding more shards.
- **Storage Capacity:** Sharding distributes data across the shards in the cluster, allowing each shard to contain a subset of the total cluster data. As the data set grows, additional shards increase the storage capacity of the cluster.

ADVANTAGES OF SHARDING

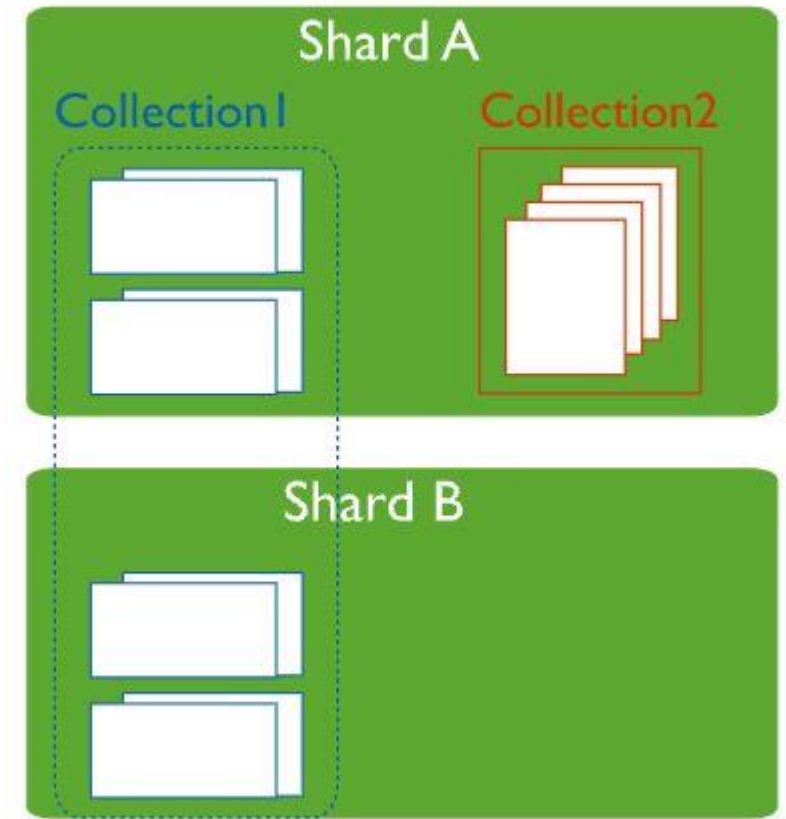
- **High Availability:** The deployment of config servers and shards as replica sets provide increased availability.
- Even if one or more shard replica sets become completely unavailable, the sharded cluster can continue to perform partial reads and writes. That is, while data on the unavailable shard(s) cannot be accessed, reads or writes directed at the available shards can still succeed.

CONSIDERING BEFORE SHARDING

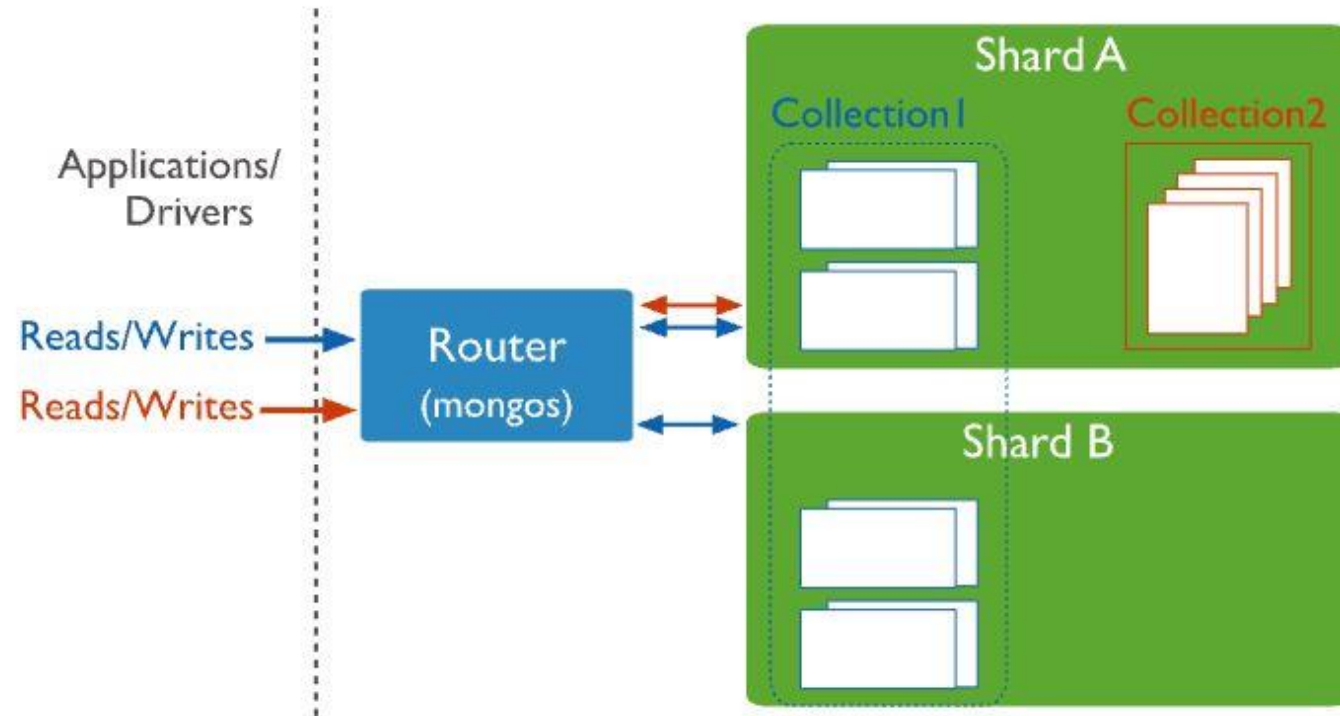
- Once a collection has been sharded, MongoDB provides no method to unshard a sharded collection.

SHARDED AND NON-SHARDED COLLECTIONS

- A database can have a mixture of sharded and unsharded collections.
- Sharded collections are partitioned and distributed across the shards in the cluster.
- Unsharded collections are stored on a primary shard. Each database has its own primary shard.



CONNECTING TO A SHARDED CLUSTER



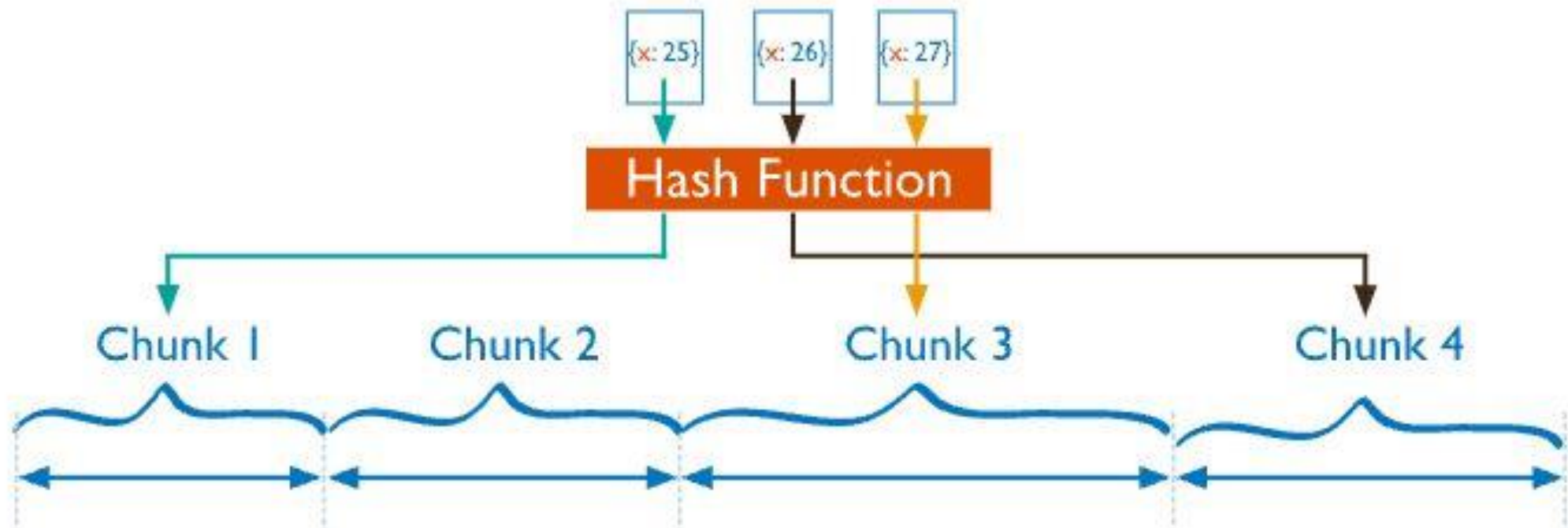
CHUNKS

- MongoDB partitions sharded data into chunks.
- A contiguous range of shard key values within a particular shard.
- Chunk ranges are inclusive of the lower boundary and exclusive of the upper boundary.
- MongoDB splits chunks when they grow beyond the configured chunk size, which by default is 64 megabytes.
- MongoDB migrates chunks when a shard contains too many chunks of a collection relative to other shards.

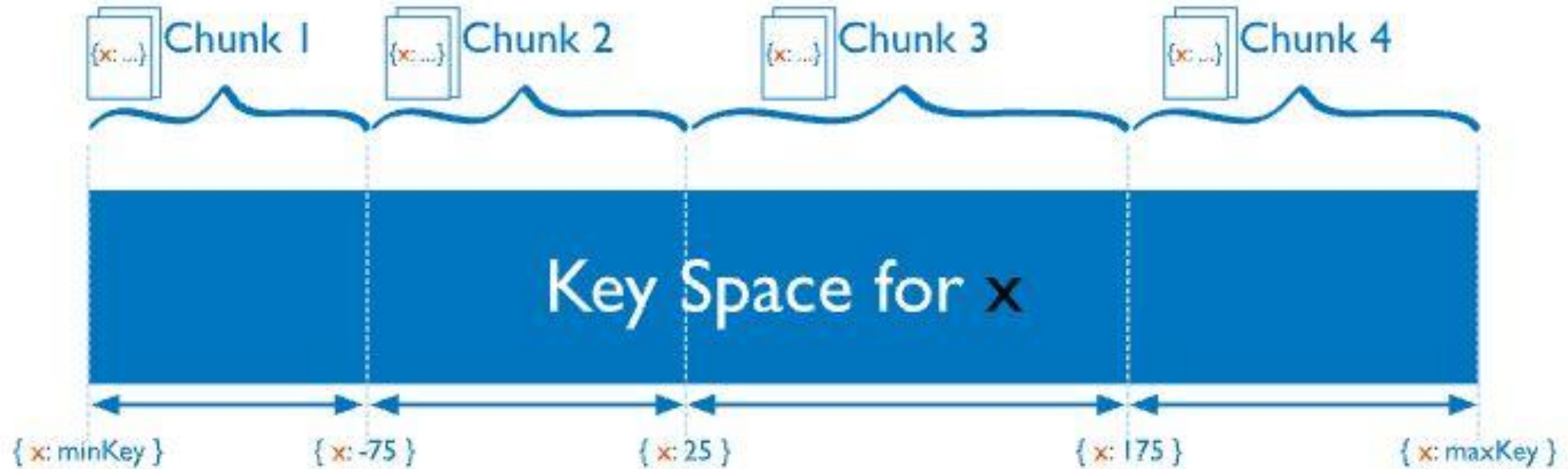
SHARDING STRATEGY

- Hashed Sharding
- Ranged Sharding

HASHED SHARDING

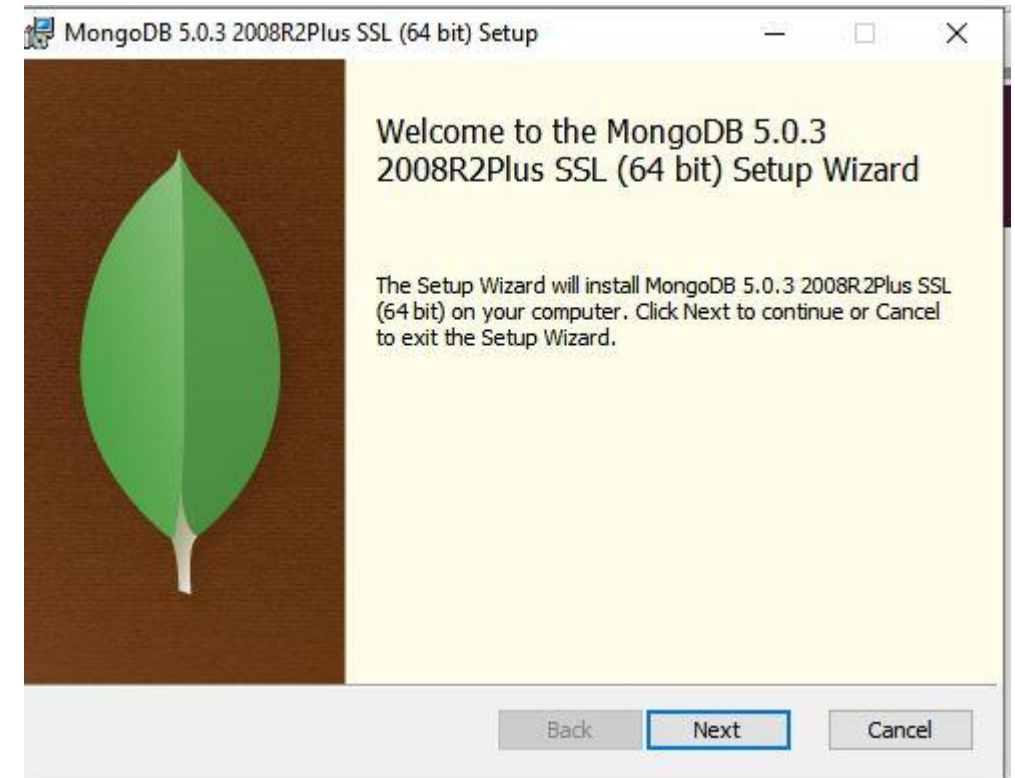
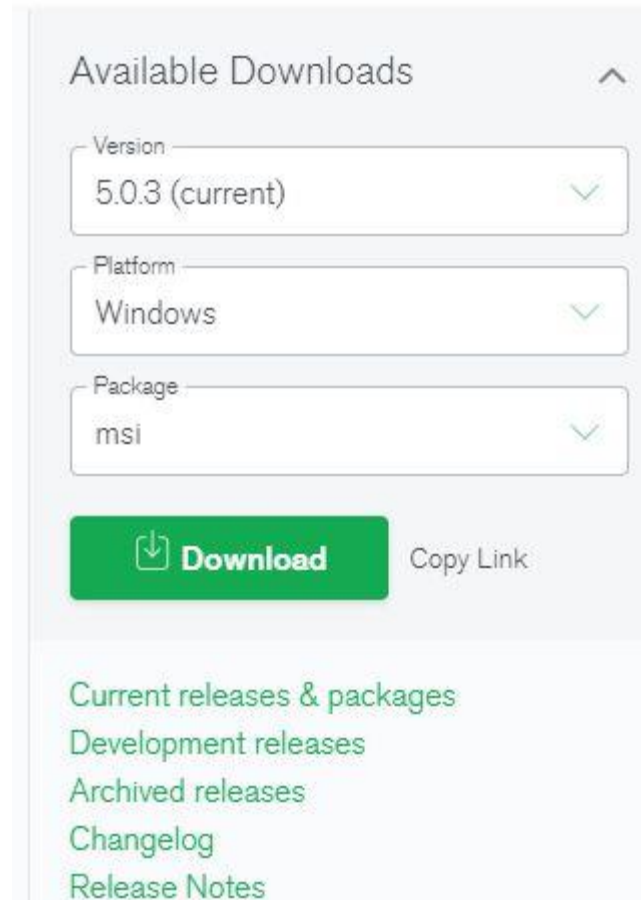


RANGED SHARDING

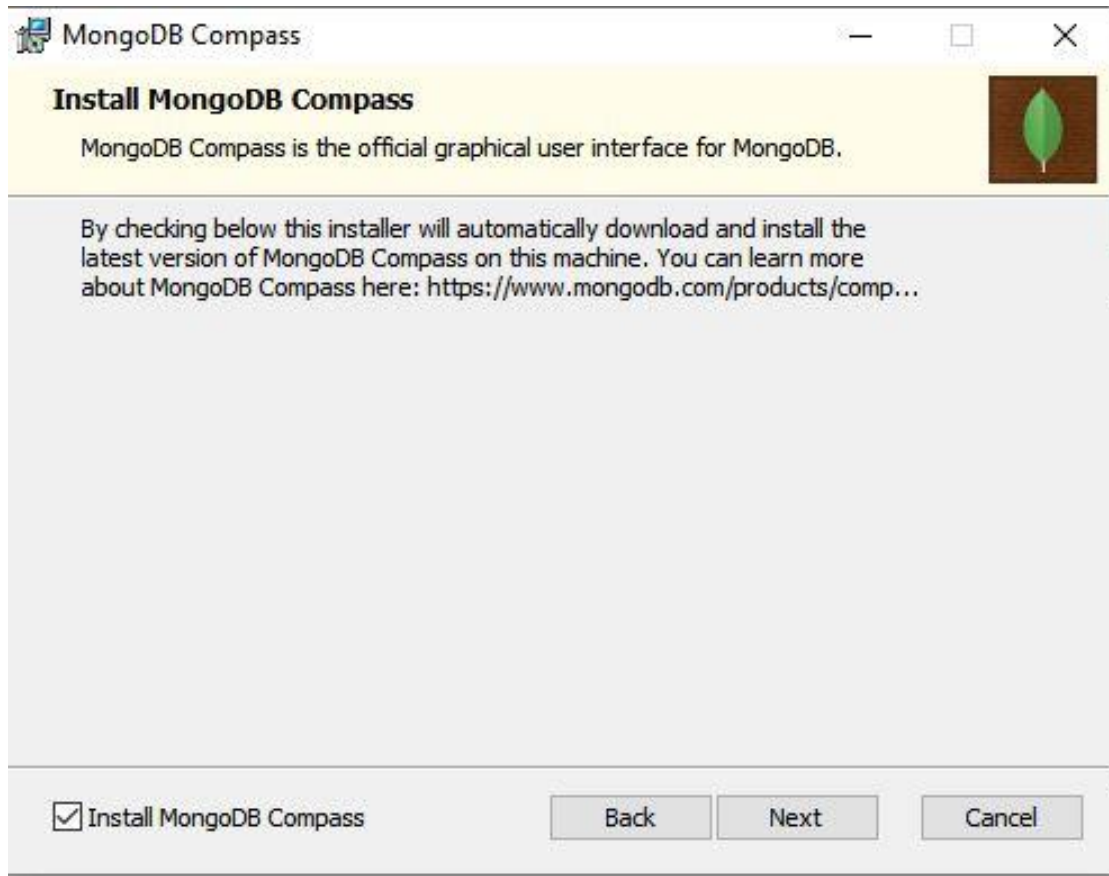


MONGODB INSTALLATION

- MongoDB Download



MONGODB INSTALLATION



MONGODB INSTALLATION

- Let's create a folder called mongodb under C: and create two files called mongo.config and mongo.log inside.
- Let's create a folder named “data” in the mongodb folder under C:
- Let's open the mongo.config file and inside the file

dbpath = C:\mongodb\data

logpath = C:\mongodb\mongo.log

Let's add the lines.

MONGODB INSTALLATION

To run the MongoDB server;

- Let's open cmd
- After coming to the mongo installation path on the command line

```
mongod.exe --config C:\mongodb\mongo.config
```

Type the line and press enter.

MONGODB INSTALLATION

Command Prompt

```
Microsoft Windows [Version 10.0.18363.1556]  
(c) 2019 Microsoft Corporation. All rights reserved.
```

```
C:\Users\Sumeyye>cd..
```

```
C:\Users>cd..
```

```
C:\>cd "Program Files"
```

```
C:\Program Files>cd MongoDB
```

```
C:\Program Files\MongoDB>cd Server
```

```
C:\Program Files\MongoDB\Server>cd 5.0
```

```
C:\Program Files\MongoDB\Server\5.0>cd bin
```

```
C:\Program Files\MongoDB\Server\5.0\bin>mongod.exe --config C:\mongodb\mongo.config
```

CRUD OPERATION

Create databases

```
Microsoft Windows [Version 10.0.18363.1556]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Sumeyye>cd..
C:\Users>cd..
C:\>cd "Program Files"
C:\Program Files>cd MongoDB
C:\Program Files\MongoDB>cd Server
C:\Program Files\MongoDB\Server>cd 5.0
C:\Program Files\MongoDB\Server\5.0>cd bin
C:\Program Files\MongoDB\Server\5.0\bin>mongo
MongoDB shell version v5.0.3
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("4350277a-77af-4805-8956-22e4e0042752") }
MongoDB server version: 5.0.3
=====
Warning: the "mongo" shell has been superseded by "mongosh",
which delivers improved usability and compatibility.The "mongo" shell has been deprecated and will be removed in
```


CRUD OPERATION

Create databases

```
Improvements and to suggest MongoDB products and deployment options to you.  
  
To enable free monitoring, run the following command: db.enableFreeMonitoring()  
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()  
---  
> use firstexample ←  
switched to db firstexample  
>
```

```
> use firstexample  
switched to db firstexample  
> show dbs  
admin    0.000GB  
config   0.000GB  
local    0.000GB  
>
```

```
> db  
firstexample  
>
```

CRUD OPERATION

Create collection

```
> db.createCollection("person");
{ "ok" : 1 }
> show collections
person
>
```

Insert data

```
> db.person.insert({adi: "Stephen Hawking"});
WriteResult({ "nInserted" : 1 })
>
```


Filter data

```
> db.person.insert({adi: "Stephen Hawking"});
WriteResult({ "nInserted" : 1 })
> db.person.insert({adi: "Albert Einstein"});
WriteResult({ "nInserted" : 1 })
> db.person.insert({adi: "Isaac Newton"});
WriteResult({ "nInserted" : 1 })
> db.person.find();
{ "_id" : ObjectId("614c5726a2fb35a04712baf2"), "adi" : "Stephen Hawking" }
{ "_id" : ObjectId("614c577aa2fb35a04712baf3"), "adi" : "Albert Einstein" }
{ "_id" : ObjectId("614c5792a2fb35a04712baf4"), "adi" : "Isaac Newton" }
>
```

Filter data

```
> db.person.find( {adi:"Isaac Newton"})
{ "_id" : ObjectId("614c5792a2fb35a04712baf4"), "adi" : "Isaac Newton" }
>
```

ROBOMONGO-ROBO 3T



Robo 3T

Simplicity Meets Power

Robo 3T: the hobbyist GUI

Robo 3T 1.4 brings support for MongoDB 4.0 to 4.2, with the ability to manually

Download Robo 3T

Robo 3T 1.4.4

Download for Windows

Email*

Please complete this required field.

First name* Last name*

Country code Phone number

Please select

By clicking on the download button, I agree to the 3T Software Labs Privacy Policy.

Download for Windows

Close

Simplicity Meets Power

Robo 3T 1.4.4


Download for Windows

[robo3t-1.4.4-windows-x86_64-e6ac9ec5.exe](#)

[robo3t-1.4.4-windows-x86_64-e6ac9ec5.zip](#)

Close

ROBOMONGO-ROBO 3T

 EULA

Thank you for choosing Robo 3T!

Share your email address with us and we'll keep you up-to-date with updates from us and new features as they come out.

First Name:

Last Name:

Email:

Phone:

Company:

By submitting this form I agree to 3T Software Labs [Privacy Policy](#).

Back

Next


Cancel

Finish

MongoDB Connections

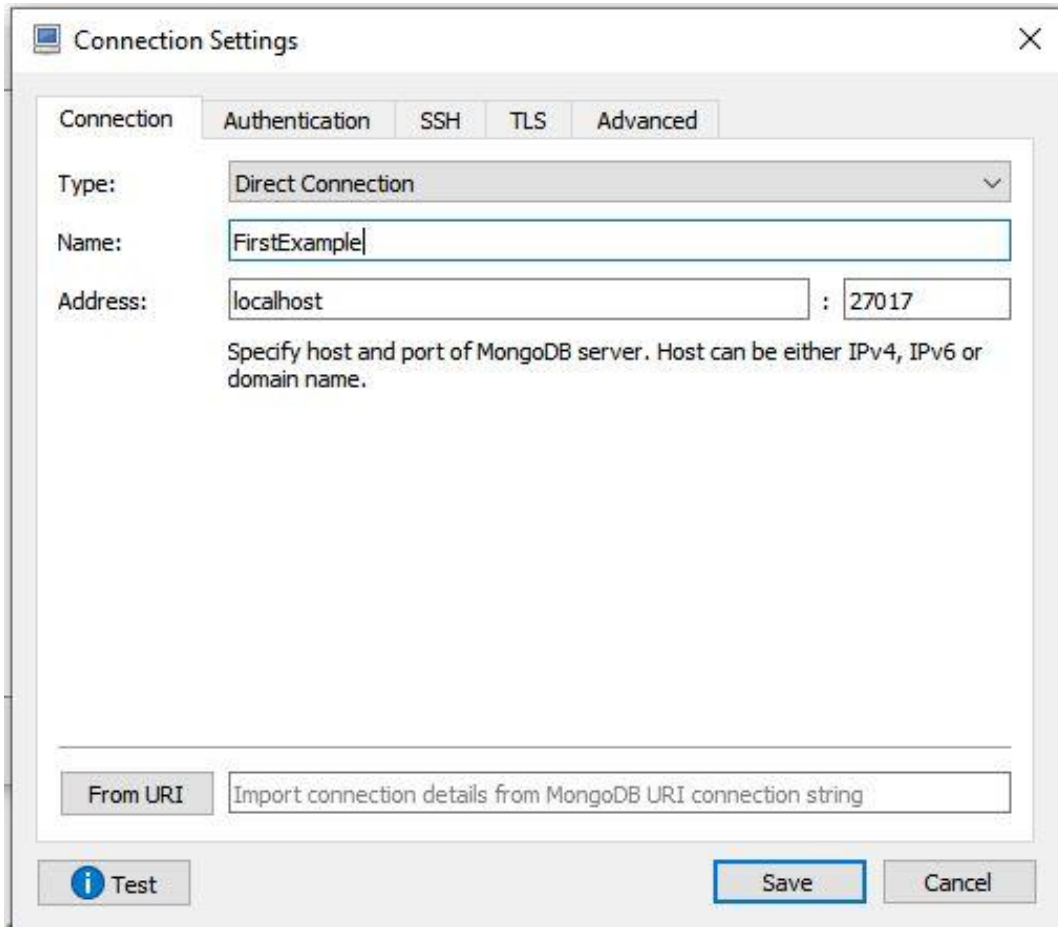
[Create](#), [edit](#), [remove](#), [done](#) or reorder connections via drag'n'drop.

Name	Address	Attributes	Auth. Database / User
------	---------	------------	-----------------------

 Connect

Cancel

ROBOMONGO-ROBO 3T



Connection Settings

Connection Authentication SSH TLS Advanced

Type: Direct Connection

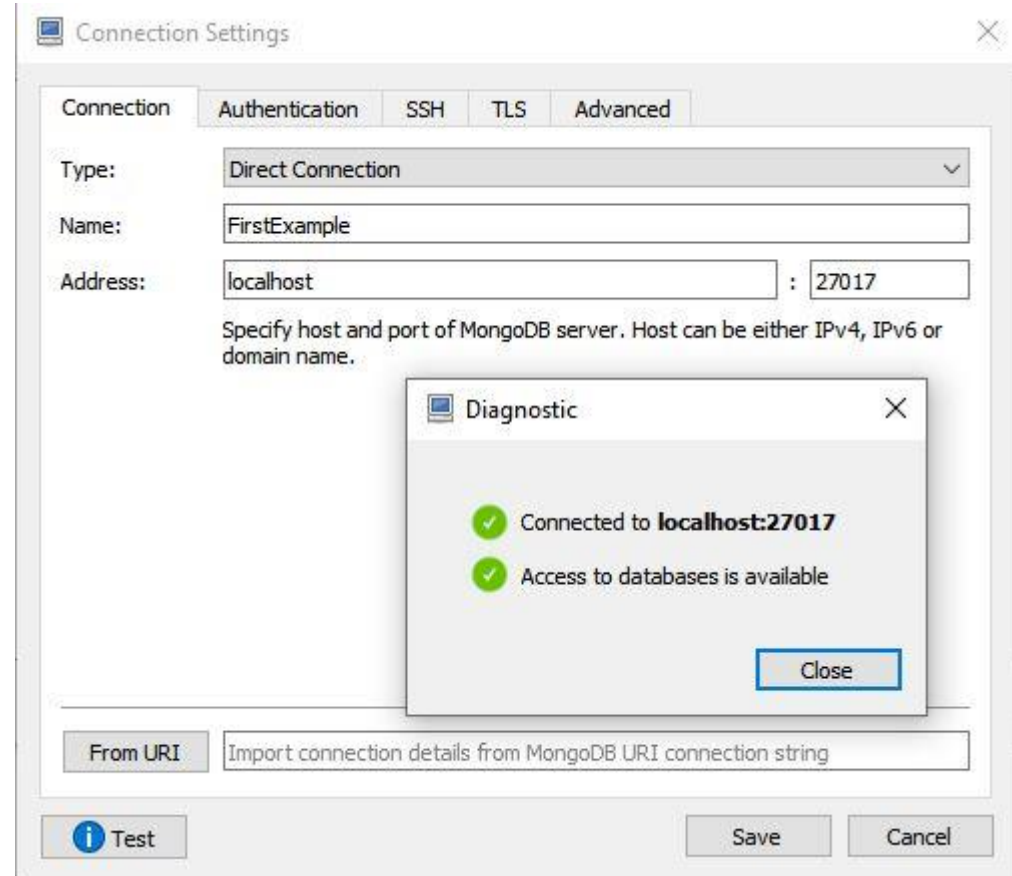
Name: FirstExample

Address: localhost : 27017

Specify host and port of MongoDB server. Host can be either IPv4, IPv6 or domain name.

From URI Import connection details from MongoDB URI connection string

Test Save Cancel



Connection Settings

Connection Authentication SSH TLS Advanced

Type: Direct Connection

Name: FirstExample

Address: localhost : 27017

Specify host and port of MongoDB server. Host can be either IPv4, IPv6 or domain name.

Diagnostic

- ✓ Connected to **localhost:27017**
- ✓ Access to databases is available

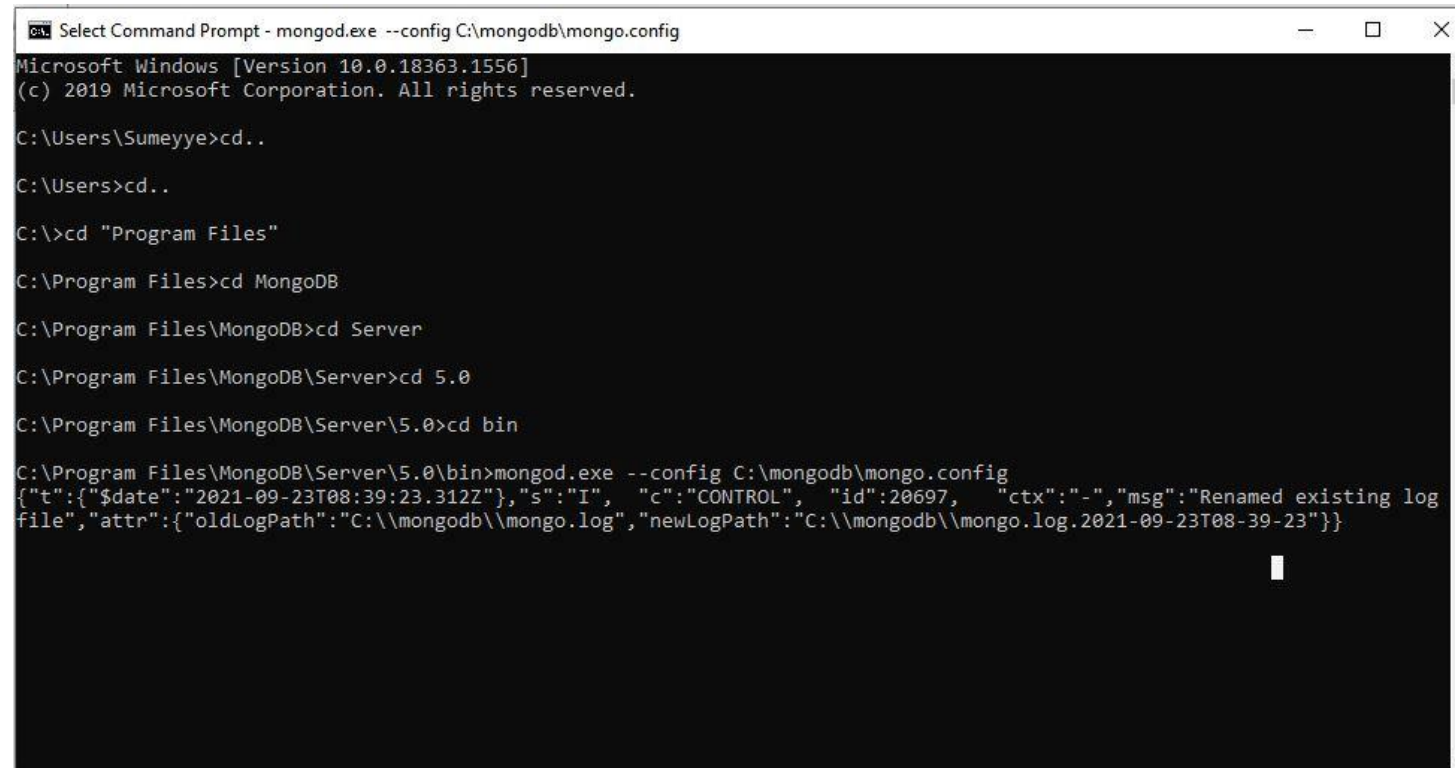
Close

From URI Import connection details from MongoDB URI connection string

Test Save Cancel

ROBOMONGO-ROBO 3T

MongoDB server



```

Select Command Prompt - mongod.exe --config C:\mongodb\mongo.config
Microsoft Windows [Version 10.0.18363.1556]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Sumeyye>cd..

C:\Users>cd..

C:\>cd "Program Files"

C:\Program Files>cd MongoDB

C:\Program Files\MongoDB>cd Server

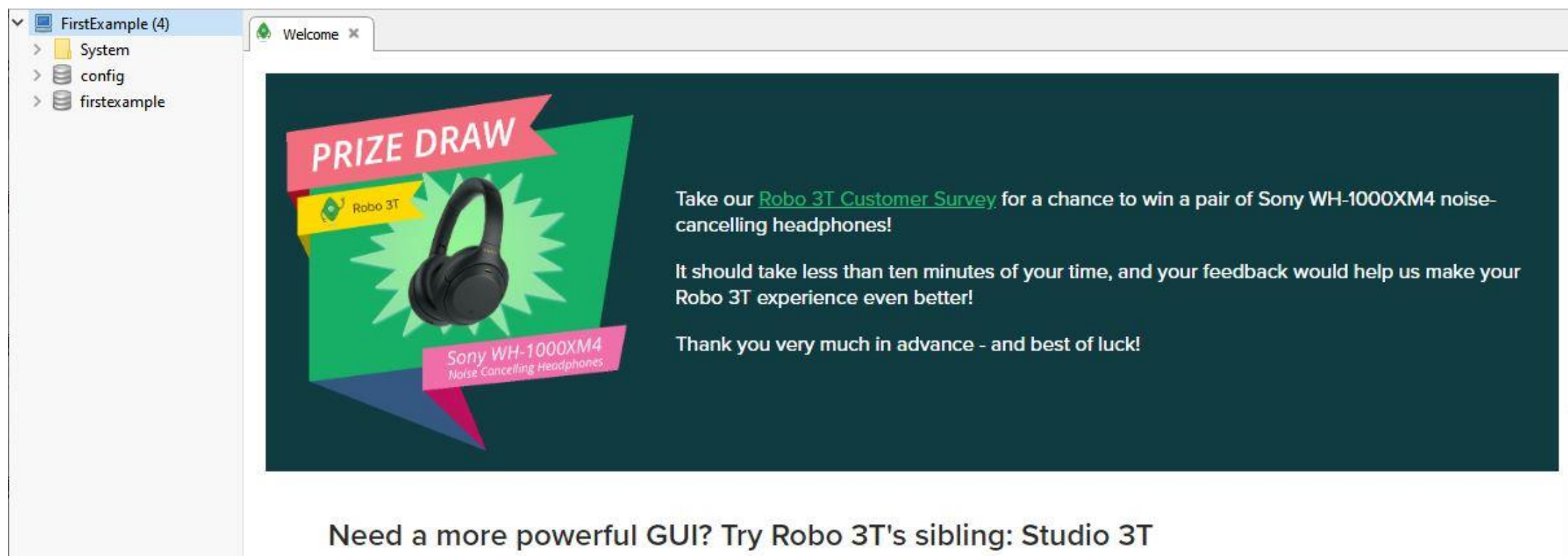
C:\Program Files\MongoDB\Server>cd 5.0

C:\Program Files\MongoDB\Server\5.0>cd bin

C:\Program Files\MongoDB\Server\5.0\bin>mongod.exe --config C:\mongodb\mongo.config
{"t":{"$date":"2021-09-23T08:39:23.312Z"},"s":"I", "c":"CONTROL", "id":20697, "ctx":"-", "msg":"Renamed existing log
file", "attr":{"oldLogPath":"C:\\mongodb\\mongo.log", "newLogPath":"C:\\mongodb\\mongo.log.2021-09-23T08-39-23"}}

```

ROBOMONGO-ROBO 3T



The screenshot displays the Robo 3T application window. On the left is a sidebar with a tree view containing 'FirstExample (4)', 'System', 'config', and 'firstexample'. The main area has a 'Welcome' tab and a large dark green banner for a 'PRIZE DRAW'. The banner features a pair of Sony WH-1000XM4 headphones and text encouraging users to take a survey for a chance to win the headphones. Below the banner, a text prompt suggests trying Studio 3T for a more powerful GUI.

FirstExample (4)

- > System
- > config
- > firstexample

Welcome

PRIZE DRAW

Robo 3T

Sony WH-1000XM4
Noise Cancelling Headphones

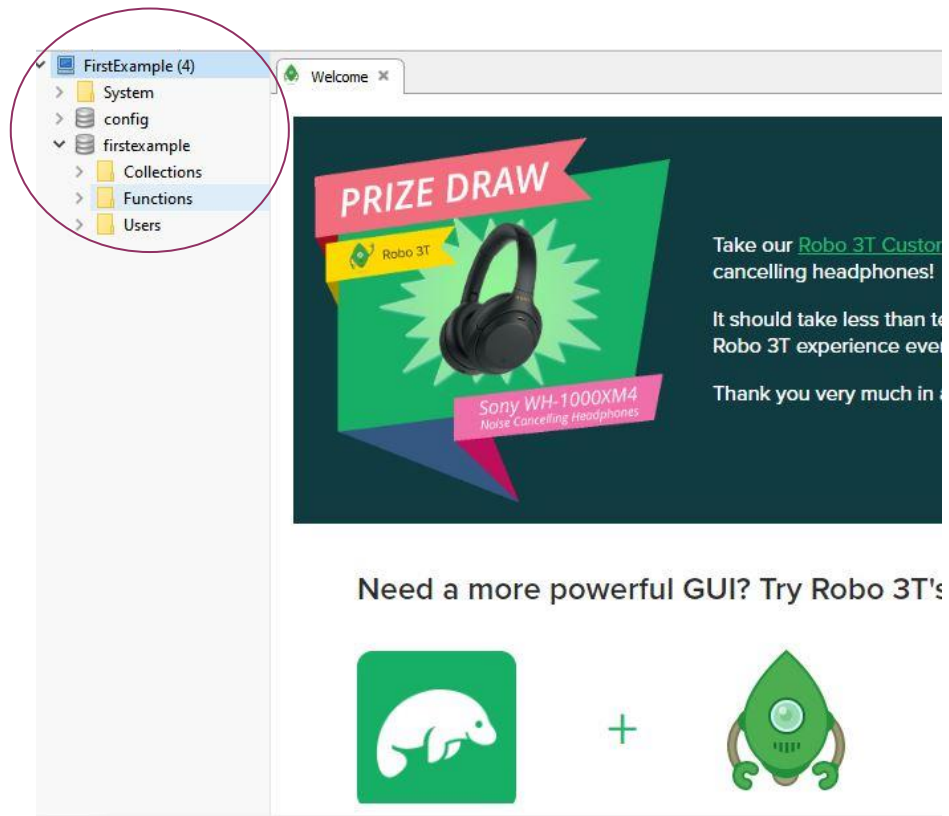
Take our [Robo 3T Customer Survey](#) for a chance to win a pair of Sony WH-1000XM4 noise-cancelling headphones!

It should take less than ten minutes of your time, and your feedback would help us make your Robo 3T experience even better!

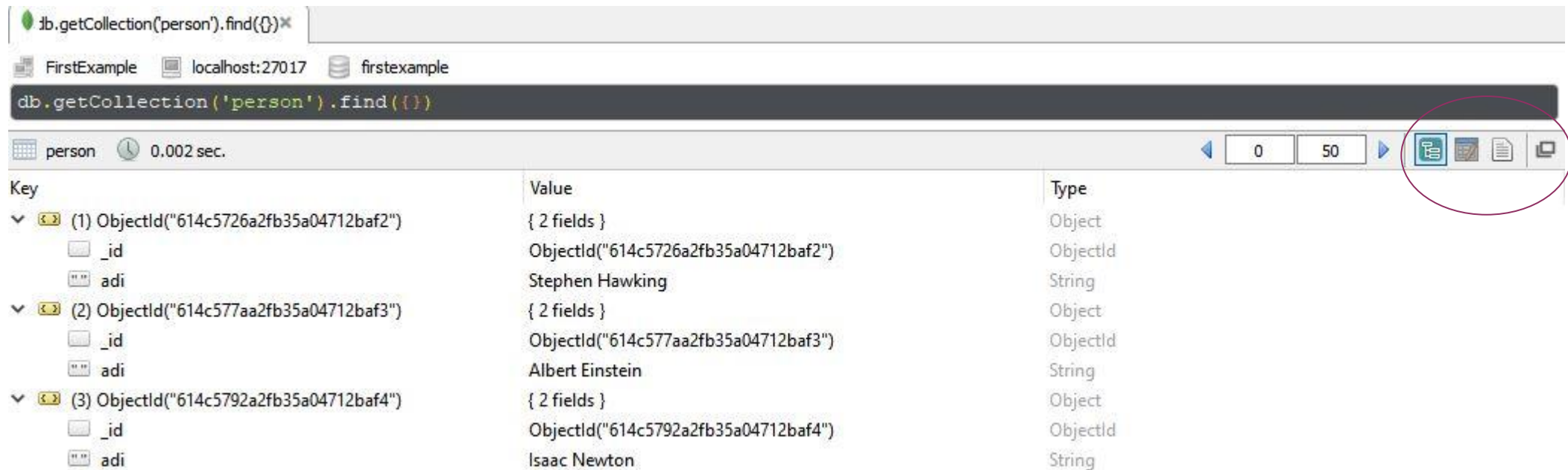
Thank you very much in advance - and best of luck!

Need a more powerful GUI? Try Robo 3T's sibling: Studio 3T

ROBOMONGO-ROBO 3T



ROBOMONGO-ROBO 3T

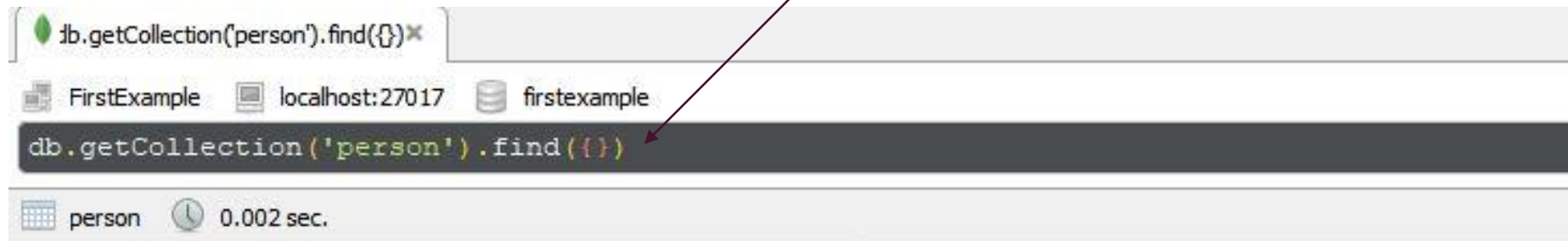


The screenshot displays the Robo 3T application interface. At the top, a command bar contains the query `db.getCollection('person').find({})`. Below this, the status bar shows the database 'firstexample' and the collection 'person', along with a query execution time of 0.002 seconds. The main area displays the results of the query in a table format with three columns: Key, Value, and Type. The results are expanded to show three documents, each with an '_id' and an 'adi' field. The 'adi' field values are 'Stephen Hawking', 'Albert Einstein', and 'Isaac Newton'. A red circle highlights the toolbar on the right side of the results table, which includes icons for copy, paste, and other actions.

Key	Value	Type
(1) ObjectId("614c5726a2fb35a04712baf2")	{ 2 fields }	Object
_id	ObjectId("614c5726a2fb35a04712baf2")	ObjectId
adi	Stephen Hawking	String
(2) ObjectId("614c577aa2fb35a04712baf3")	{ 2 fields }	Object
_id	ObjectId("614c577aa2fb35a04712baf3")	ObjectId
adi	Albert Einstein	String
(3) ObjectId("614c5792a2fb35a04712baf4")	{ 2 fields }	Object
_id	ObjectId("614c5792a2fb35a04712baf4")	ObjectId
adi	Isaac Newton	String

ROBOMONGO-ROBO 3T

Commands are written here.







run



BUILD A WEB APP WITH ASP.NET CORE AND MONGODB

Car Gallery

[Add New Car](#)

Photo	Brand	Model	Year	Price(\$)	
	Porsche	Cayenne	2018	60,600	Edit Details Delete
	Maserati	Quattroporte	2019	107,680	Edit Details Delete
	Bentley	Bentayga	2019	165,000	Edit Details Delete
	Chrysler	300	2018	28,995	Edit Details Delete

CREATE A NEW ASP.NET CORE PROJECT

Create a new project

Recent project templates

	ASP.NET Core Web Application	C#
	Console App (.NET Core)	C#
	ASP.NET Web Application (.NET Framework)	C#
	ASP.NET Core Web Application	F#

Search for templates (Alt+S)



[Clear all](#)

C#

All platforms

All project types



Console App (.NET Core)

A project for creating a command-line application that can run on .NET Core on Windows, Linux and MacOS.

C# Linux macOS Windows Console



ASP.NET Core Web Application

Project templates for creating ASP.NET Core web apps and web APIs for Windows, Linux and macOS using .NET Core or .NET Framework. Create web apps with Razor Pages, MVC, or Single Page Apps (SPA) using Angular, React, or React + Redux.

C# Linux macOS Windows Cloud Service Web



Blazor App

Project templates for creating Blazor apps that run on the server in an ASP.NET Core app or in the browser on WebAssembly (wasm). These templates can be used to build web apps with rich dynamic user interfaces (UIs).

C# Linux macOS Windows Cloud Web

CREATE MVC PROJECT

Configure your new ASP.NET Core web application


ASP.NET Core Web Application

Project name

CarGalleryApp

Location

C:\Users\Sumeyye\source\repos

Solution name 

CarGalleryApp

☐ Place solution and project in the same directory

Create a new ASP.NET Core web application

.NET Core

ASP.NET Core 3.1



Empty

An empty project template for creating an ASP.NET Core application. This template does not have any content in it.



API

A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.



Web Application

A project template for creating an ASP.NET Core application with example ASP.NET Razor Pages content.



Web Application (Model-View-Controller)

A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.

ADD MONGODB DRIVER

The screenshot shows the NuGet Package Manager window for the 'CarGalleryApp' project. The 'Browse' tab is active, and the search bar contains 'mongodb'. The search results list three packages: 'MongoDB.Bson', 'MongoDB.Driver' (which is selected), and 'MongoDB.Driver.Core'. The right-hand pane displays details for 'MongoDB.Driver', including its version (2.13.1), author (MongoDB Inc.), and a description. The 'Install' button is visible next to the version dropdown.

NuGet: CarGalleryApp → X CarGalleryApp

Browse Installed Updates

mongodb x ↕ ☐ Include prerelease

Package source: nuget.org ⚙

Package Name	Author	Downloads	Version
MongoDB.Bson	MongoDB Inc.	60,2M	v2.13.1
MongoDB.Driver	MongoDB Inc.	55,4M	v2.13.1
MongoDB.Driver.Core	MongoDB Inc.	56,4M	v2.13.1

Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.

☐ Do not show this again

MongoDB.Driver nuget.org

Version: Latest stable 2.13.1

Options

Description
Official .NET driver for MongoDB.

Version: 2.13.1
Author(s): MongoDB Inc.
License: [View License](#)
Date published: Thursday, August 5, 2021 (8/5/2021)

LAYOUT EDITING

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>@ViewData["Title"] - Car Gallery App</title>
  <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
  <link rel="stylesheet" href="~/css/site.css" />
</head>
<body>
  <header>
    <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white border-bottom box-shadow mb-3">
      <div class="container">
        <a class="navbar-brand" asp-area="" asp-controller="Home" asp-action="Index">Car Gallery App</a>
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-target=".navbar-collapse" aria-expanded="false" aria-label="Toggle navigation">
          <span class="navbar-toggler-icon"></span>
        </button>
      </div>
    </nav>
  </header>
  <div class="main">
    @RenderSection("Main", required: false)
  </div>
  <div class="border-top footer text-muted">
    <div class="container">
      &copy; 2021 - Car Gallery App - <a asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>
    </div>
  </div>
  <script src="~/lib/jquery/dist/jquery.min.js"></script>
  <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
  <script src="~/js/site.js" asp-append-version="true"></script>
  @RenderSection("Scripts", required: false)
</body>
</html>
```

ADDING A MODEL

```
25 references
public class Cars
{
    [BsonId]
    [BsonRepresentation(MongoDB.Bson.BsonType.ObjectId)]
    14 references
    public string ID { get; set; }

    [Required]
    [BsonElement("Model")]
    12 references
    public string Model { get; set; }

    [Required]
    [BsonElement("Year")]
    12 references
    public int Year { get; set; }

    [BsonElement("Price")]
    12 references
    public decimal Price { get; set; }

    [BsonElement("ImageURL")]
    [Required]
    12 references
    public string ImageURL { get; set; }
}
```


ADDING A MODEL

```
0 references
public class Cars
{
    [BsonId]
    [BsonRepresentation(MongoDB.Bson.BsonType.ObjectId)]
    0 references
    public int ID { get; set; }

    [Required]
    [BsonElement("Model")]
    0 references
    public string Model { get; set; }

    [Required]
    [BsonElement("Year")]
    0 references
    public int Year { get; set; }

    [BsonElement("Price")]
    0 references
    public decimal Price { get; set; }

    [BsonElement("ImageURL")]
    [Required]
    0 references
    public string ImageURL { get; set; }
}
```

↑ fail

```
25 references
public class Cars
{
    [BsonId]
    [BsonRepresentation(MongoDB.Bson.BsonType.ObjectId)]
    14 references
    public string ID { get; set; }

    [Required]
    [BsonElement("Model")]
    12 references
    public string Model { get; set; }

    [Required]
    [BsonElement("Year")]
    12 references
    public int Year { get; set; }

    [BsonElement("Price")]
    12 references
    public decimal Price { get; set; }

    [BsonElement("ImageURL")]
    [Required]
    12 references
    public string ImageURL { get; set; }
}
```

ADDING A CRUD SERVICES CLASS

```
4 references
public class CarServices
{
    private readonly IMongoCollection<Cars> cars;
    0 references
    public CarServices(IConfiguration config)
    {
        MongoClient client = new MongoClient(config.GetConnectionString("CarGalleryDb"));
        IMongoDatabase database = client.GetDatabase("CarGalleryDb");
        cars = database.GetCollection<Cars>("Cars");
    }
    1 reference
    public List<Cars> Get()
    {
        return cars.Find(car => true).ToList();
    }
    4 references
    public Cars Get(string id)
    {
        return cars.Find(car => car.ID == id).FirstOrDefault();
    }
    1 reference
    public Cars Create(Cars car)
    {
        cars.InsertOne(car);
        return car;
    }
}
```

```
1 reference
public void Update(string id, Cars carIn)
{
    cars.ReplaceOne(car => car.ID == id, carIn);
}
0 references
public void Remove(Cars carIn)
{
    cars.DeleteOne(car => car.ID == carIn.ID);
}
1 reference
public void Remove(string id)
{
    cars.DeleteOne(car => car.ID == id);
}
```

ADDING A CRUD SERVICES CLASS

- The “CarServices” class uses the following MongoDB.Driver members to perform CRUD operations against the database.

```
private readonly IMongoCollection<Cars> cars;
0 references
public CarServices(IConfiguration config)
{
    MongoClient client = new MongoClient(config.GetConnectionString("CarGalleryDb"));
    IMongoDatabase database = client.GetDatabase("CarGalleryDb");
    cars = database.GetCollection<Cars>("Cars");
}
```

ADDING A CRUD SERVICES CLASS

- **MongoClient:** Reads the server instance for performing database operations. The constructor of this class is provided the MongoDB connection string.
- **IMongoDatabase:** Represents the Mongo database for performing operations.

“COLLECTION” DEFINITION

- `GetCollection<T>(collection)`
- “collection” represents the collection name in the database.
- “T” represents the CLR object type stored in the collection.

A CRUD SERVICES CLASS

```
public List<Cars> Get()
{
    return cars.Find(car => true).ToList();
}
```

```
4 references
public Cars Get(string id)
{
    return cars.Find(car => car.ID == id).FirstOrDefault();
}
```

```
1 reference
public Cars Create(Cars car)
{
    cars.InsertOne(car);
    return car;
}
```

```
1 reference
public void Update(string id, Cars carIn)
{
    cars.ReplaceOne(car => car.ID == id, carIn);
}
```

```
0 references
public void Remove(Cars carIn)
{
    cars.DeleteOne(car => car.ID == carIn.ID);
}
```

```
1 reference
public void Remove(string id)
{
    cars.DeleteOne(car => car.ID == id);
}
```

ADD THE MONGODB CONNECTION STRING

- appsettings.json

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "ConnectionStrings": {
    "CarGalleryDb": "mongodb://localhost:27017"
  },
  "AllowedHosts": "*"
}
```

REGISTERING SERVICE WITH THE DEPENDENCY INJECTION SYSTEM

- Startup.cs

```
// references  
public void ConfigureServices(IServiceCollection services)  
{  
    services.AddScoped<CarServices>();  
    services.AddControllersWithViews();  
}
```


ADDING A CONTROLLER

```
1 reference
public class CarsController : Controller
{
    private readonly CarServices carservice;

    0 references
    public CarsController(CarServices carservice)
    {
        this.carservice = carservice;
    }
    // GET: CarsController
}
```

INDEX METHOD AND VIEW

- Change the Index method in *CarsController.cs* as follows:

```
3 references  
public ActionResult Index()  
{  
    return View(carservice.Get());  
}
```

ADD INDEX VIEW

service;

Get

File

(s

st(

Add Razor View

View name:

Index

Template:

List

Model class:

Cars (CarGalleryApp.Models)

Options:

☐ Create as a partial view

☒ Reference script libraries

☒ Use a layout page:

...

(Leave empty if it is set in a Razor _viewstart file)

Add

Cancel

INDEX VIEW

Open the "Index.cshml" and make the following changes:

- Remove the Id fields
- Change the title Index to **Car Gallery**
- Change Create New to **Add New Car**
- Change the **ImageUrl** field to:

```
<td>  
    @if (item.ID != null)  
    {  
          
    }  
</td>
```

INDEX VIEW

- Update the action links as below:

```
<td>  
    @Html.ActionLink("Edit", "Edit", new { id=item.ID}) |  
    @Html.ActionLink("Details", "Details", new { id=item.ID}) |  
    @Html.ActionLink("Delete", "Delete", new { id=item.ID})  
</td>
```

CREATE VIEW

```
<h4>Cars</h4>
<hr />
<div class="row">
  <div class="col-md-4">
    <form asp-action="Create">
      <div asp-validation-summary="ModelOnly" class="text-danger"></div>
      <div class="form-group">
        <label asp-for="Model" class="control-label"></label>
        <input asp-for="Model" class="form-control" />
        <span asp-validation-for="Model" class="text-danger"></span>
      </div>
      <div class="form-group">
        <label asp-for="Year" class="control-label"></label>
        <input asp-for="Year" class="form-control" />
        <span asp-validation-for="Year" class="text-danger"></span>
      </div>
      <div class="form-group">
        <label asp-for="Price" class="control-label"></label>
        <input asp-for="Price" class="form-control" />
        <span asp-validation-for="Price" class="text-danger"></span>
      </div>
      <div class="form-group">
        <label asp-for="ImageURL" class="control-label"></label>
        <input asp-for="ImageURL" class="form-control" />
        <span asp-validation-for="ImageURL" class="text-danger"></span>
      </div>
      <div class="form-group">
        <input type="submit" value="Create" class="btn btn-primary" />
      </div>
    </form>
  </div>
</div>
```

CREATE POST METHOD

```
// POST: CarsController/Create
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Create(Cars car)
{
    if (ModelState.IsValid)
    {
        carservice.Create(car);
        return RedirectToAction(nameof(Index));
    }
    return View(car);
}
```

ADDING A NEW RECORD

Create Cars

Model

Year

Price

ImageURL

Create

Back to List

Index


[Add New Car](#)

Model	Year	Price	ImageURL	
Porsche	2020	1120000,00		Edit Details Delete

EDIT VIEW

```
<div class="col-md-4">
  <form asp-action="Edit">
    <div asp-validation-summary="ModelOnly" class="text-danger"></div>
    <div class="form-group">
      <label asp-for="Model" class="control-label"></label>
      <input asp-for="Model" class="form-control" />
      <span asp-validation-for="Model" class="text-danger"></span>
    </div>
    <div class="form-group">
      <label asp-for="Year" class="control-label"></label>
      <input asp-for="Year" class="form-control" />
      <span asp-validation-for="Year" class="text-danger"></span>
    </div>
    <div class="form-group">
      <label asp-for="Price" class="control-label"></label>
      <input asp-for="Price" class="form-control" />
      <span asp-validation-for="Price" class="text-danger"></span>
    </div>
    <div class="form-group">
      <label asp-for="ImageURL" class="control-label"></label>
      <input asp-for="ImageURL" class="form-control" />
      <span asp-validation-for="ImageURL" class="text-danger"></span>
    </div>
    <div class="form-group">
      <input type="submit" value="Save" class="btn btn-primary" />
    </div>
  </form>
```

EDIT GET AND POST METHODS

Model	Year	Price	ImageURL	
Porsche	2020	1120000,00		Edit Details Delete

Edit

Cars

Model

Year

Price

ImageURL

EDIT GET AND POST METHODS

// GET: CarsController/Edit/5

```
0 references  
public ActionResult Edit(string id)  
{  
    if (id == null)  
    {  
        return NotFound();  
    }  
  
    var car = carservice.Get(id);  
    if (car == null)  
    {  
        return NotFound();  
    }  
    return View(car);  
}
```

// POST: CarsController/Edit/5

```
[HttpPost]  
[ValidateAntiForgeryToken]  
0 references  
public ActionResult Edit(string id, Cars cars)  
{  
    if (id != cars.ID)  
    {  
        return NotFound();  
    }  
    if (ModelState.IsValid)  
    {  
        carservice.Update(id, cars);  
        return RedirectToAction(nameof(Index));  
    }  
    else  
    {  
        return View(cars);  
    }  
}
```

DETAILS VIEW

```
<dl class="row">
  <dt class = "col-sm-2">
    @Html.DisplayNameFor(model => model.Model)
  </dt>
  <dd class = "col-sm-10">
    @Html.DisplayFor(model => model.Model)
  </dd>
  <dt class = "col-sm-2">
    @Html.DisplayNameFor(model => model.Year)
  </dt>
  <dd class = "col-sm-10">
    @Html.DisplayFor(model => model.Year)
  </dd>
  <dt class = "col-sm-2">
    @Html.DisplayNameFor(model => model.Price)
  </dt>
  <dd class = "col-sm-10">
    @Html.DisplayFor(model => model.Price)
  </dd>
  <dt class = "col-sm-2">
    @Html.DisplayNameFor(model => model.ImageURL)
  </dt>
  <dd class = "col-sm-10">
    @Html.DisplayFor(model => model.ImageURL)
  </dd>
</dl>
```

DETAILS GET METHOD

```
// GET: CarsController/Details/5
0 references
public ActionResult Details(string id)
{
    if (id == null)
    {
        return NotFound();
    }

    var car = carservice.Get(id);
    if (car == null)
    {
        return NotFound();
    }
    return View(car);
}
```

DELETE VIEW

```
<dl class="row">
  <dt class = "col-sm-2">
    @Html.DisplayNameFor(model => model.Model)
  </dt>
  <dd class = "col-sm-10">
    @Html.DisplayFor(model => model.Model)
  </dd>
  <dt class = "col-sm-2">
    @Html.DisplayNameFor(model => model.Year)
  </dt>
  <dd class = "col-sm-10">
    @Html.DisplayFor(model => model.Year)
  </dd>
  <dt class = "col-sm-2">
    @Html.DisplayNameFor(model => model.Price)
  </dt>
  <dd class = "col-sm-10">
    @Html.DisplayFor(model => model.Price)
  </dd>
  <dt class = "col-sm-2">
    @Html.DisplayNameFor(model => model.ImageURL)
  </dt>
  <dd class = "col-sm-10">
    @Html.DisplayFor(model => model.ImageURL)
  </dd>
</dl>
```


DELETE GET AND POST METHODS

```
// GET: CarsController/Delete/5
0 references
public ActionResult Delete(string id)
{
    if (id == null)
    {
        return NotFound();
    }

    var car = carservice.Get(id);
    if (car == null)
    {
        return NotFound();
    }
    return View(car);
}
```

```
// POST: CarsController/Delete/5
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
0 references
public ActionResult DeleteConfirmed(string id)
{
    try
    {
        var car = carservice.Get(id);

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

        carservice.Remove(car.ID);

        return RedirectToAction(nameof(Index));
    }
    catch
    {
        return View();
    }
}
```