When we make applications

Application have User Interface

1. Console based (CUI) (Standalone , can be run on one machine)
2. Graphical Based (GUI)
3. Windows Application (Standalone)
4. Web Applications (They are stored on server & can be accessesd from a client machine)

Scalability /Reachability is more

Every application could have many components

FE (Front End) , Console, Windows , Web

BE (Back End) , Database (Sql Server , MySql , Oracle , XML file)

Middlelayer (It is in between FE and BE)

Applications cud be of different types depending upon where you put these three components

1. Single Tier (Every component, FE , BE , Middlelayer on one machine)
2. 2-Tier (Every component, FE , BE , Middlelayer on one machine)

FE / /Presentation will be on client

BE / Data Layer on server

1. 3-Tier One machine ,we put Presenation (FE) , Backend on 2nd machine , Middlelayer \_ 3rd Machine
2. N-Tier

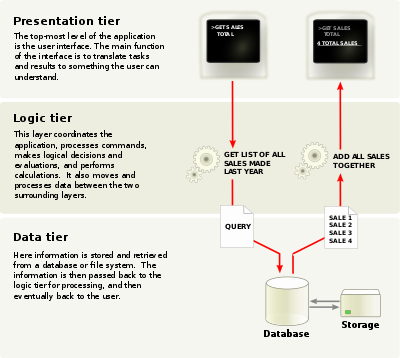
Advantage of putting layers on different machines??

Easy to understand / We can differentiate

Easy to manange

Load is divided

In 3-Tier architecture, there is a dependency of FE on BR through Business Layer



MVC is a design Pattern

There is a separation of concerns from the beginning

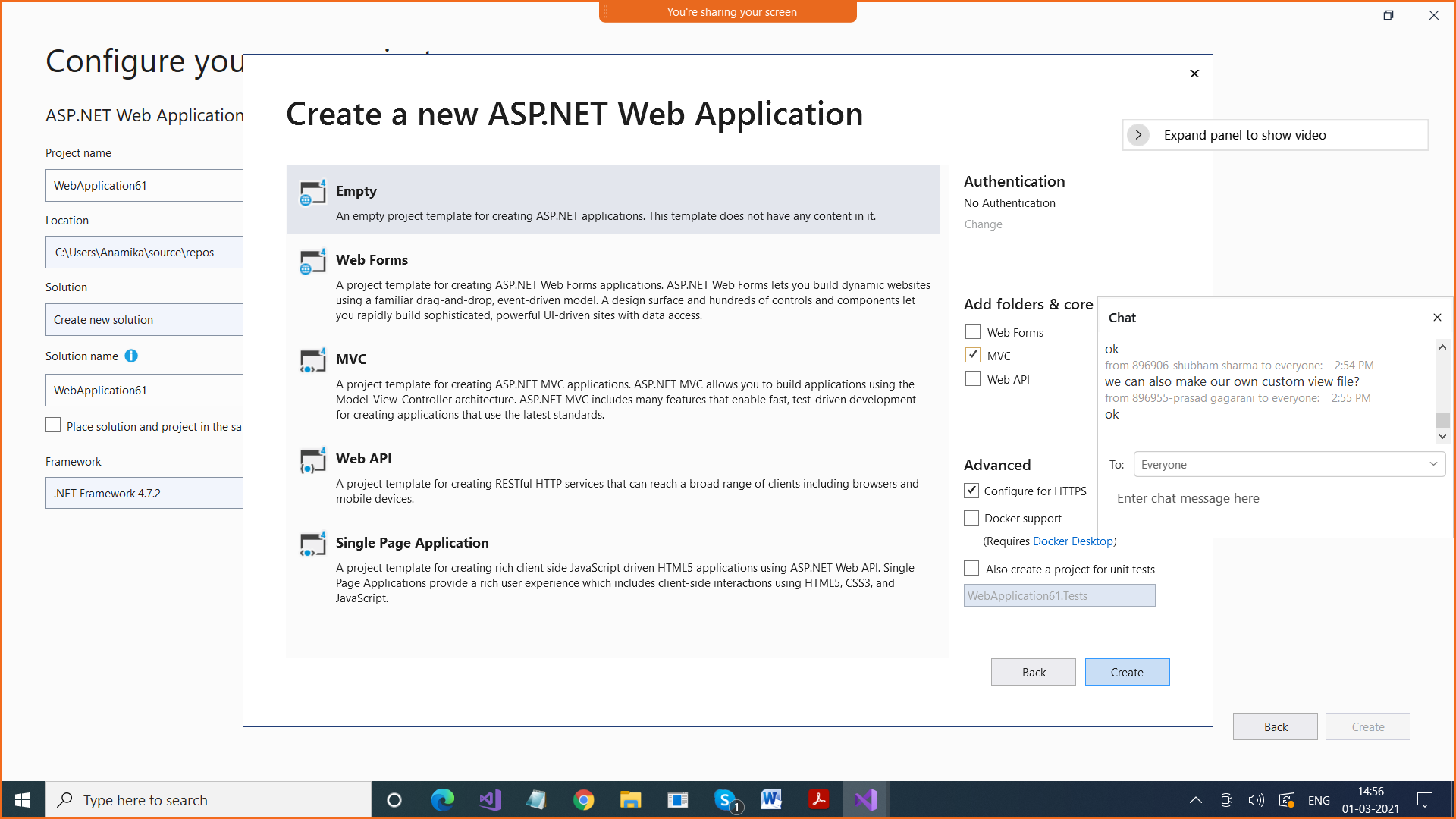
MVC ? Model , View , Controller

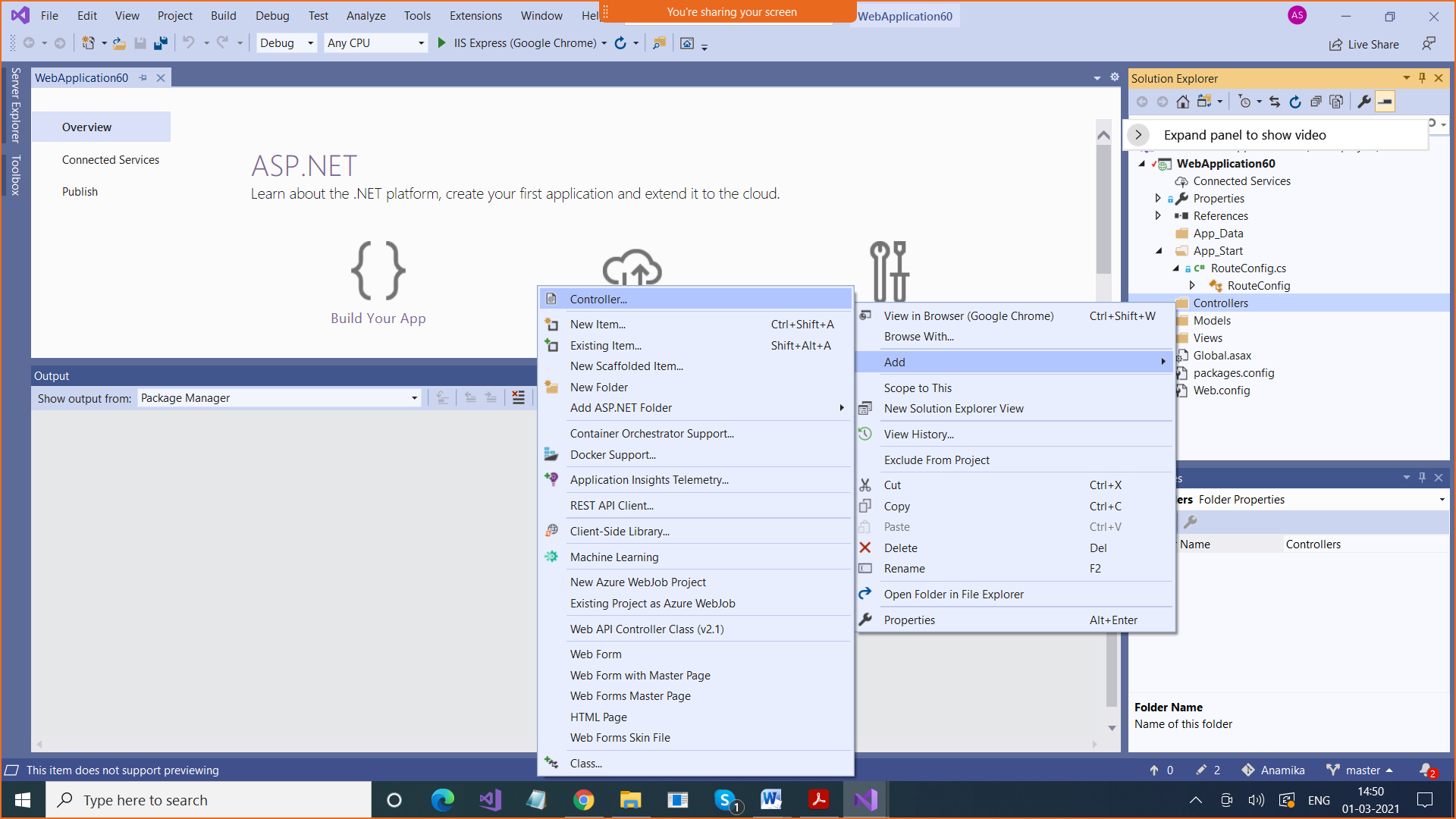
Model > Main domain , entities

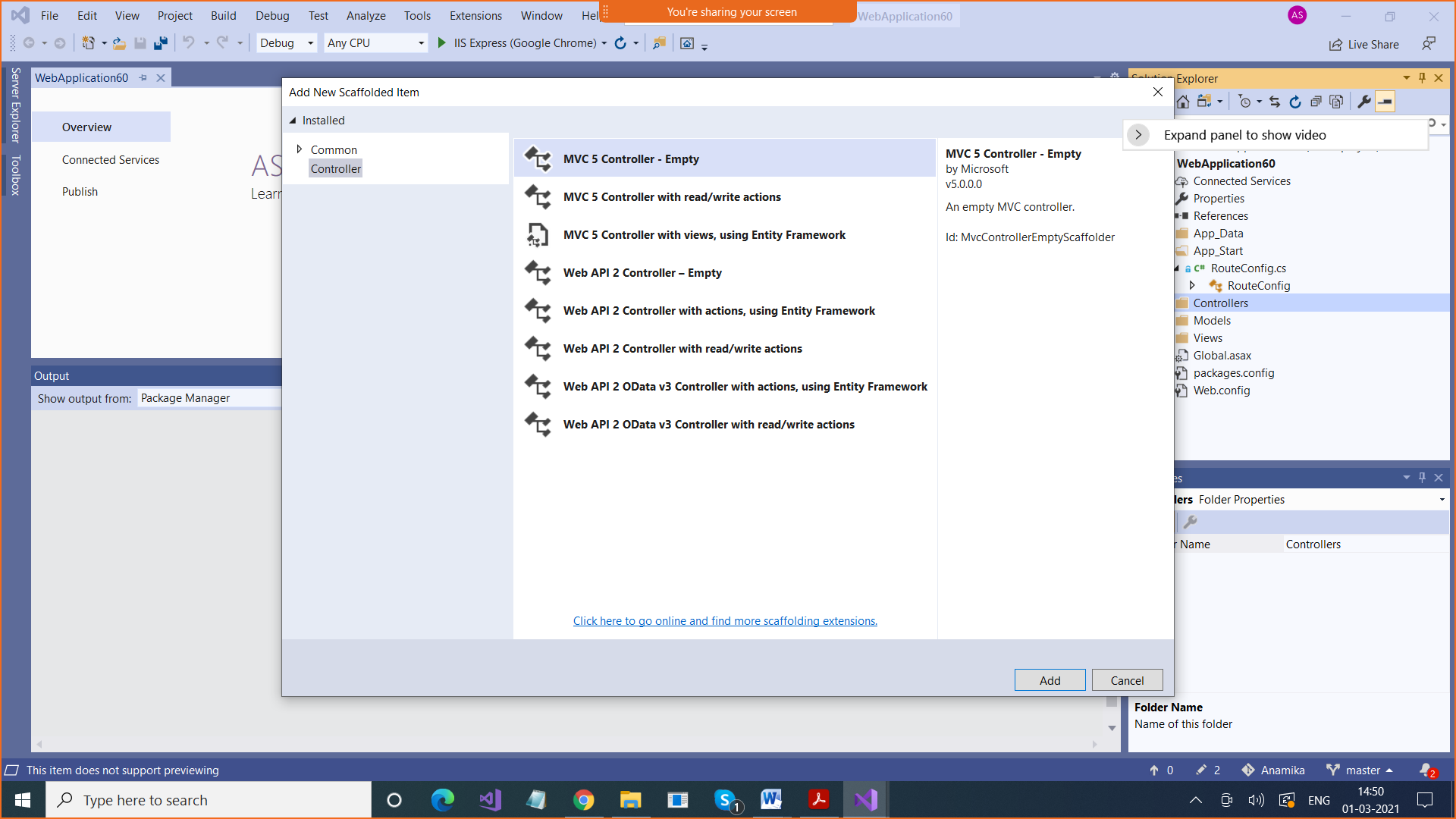
View > Presentation

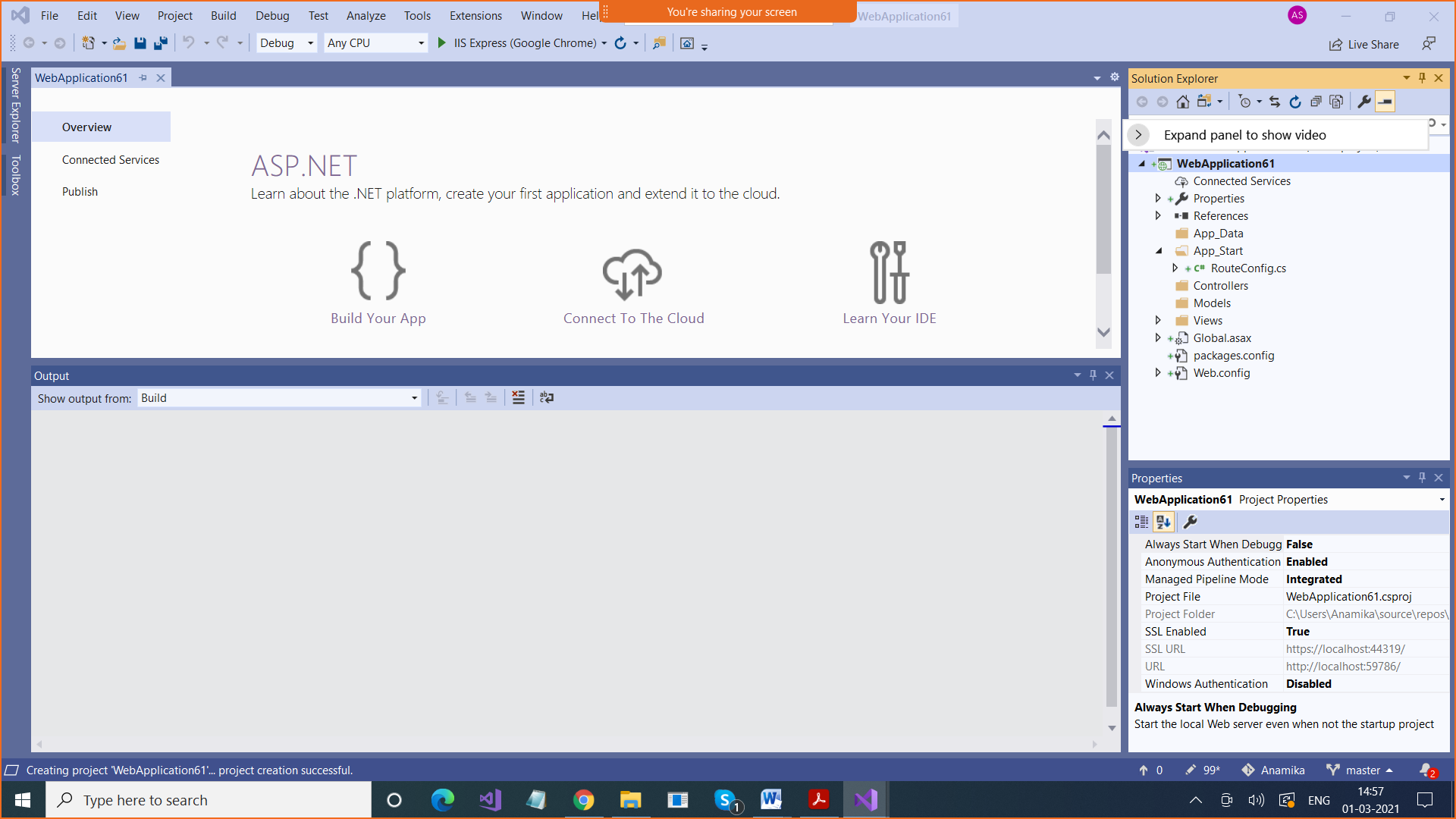
Controller > It takes input from user and routes request to the appropriate method

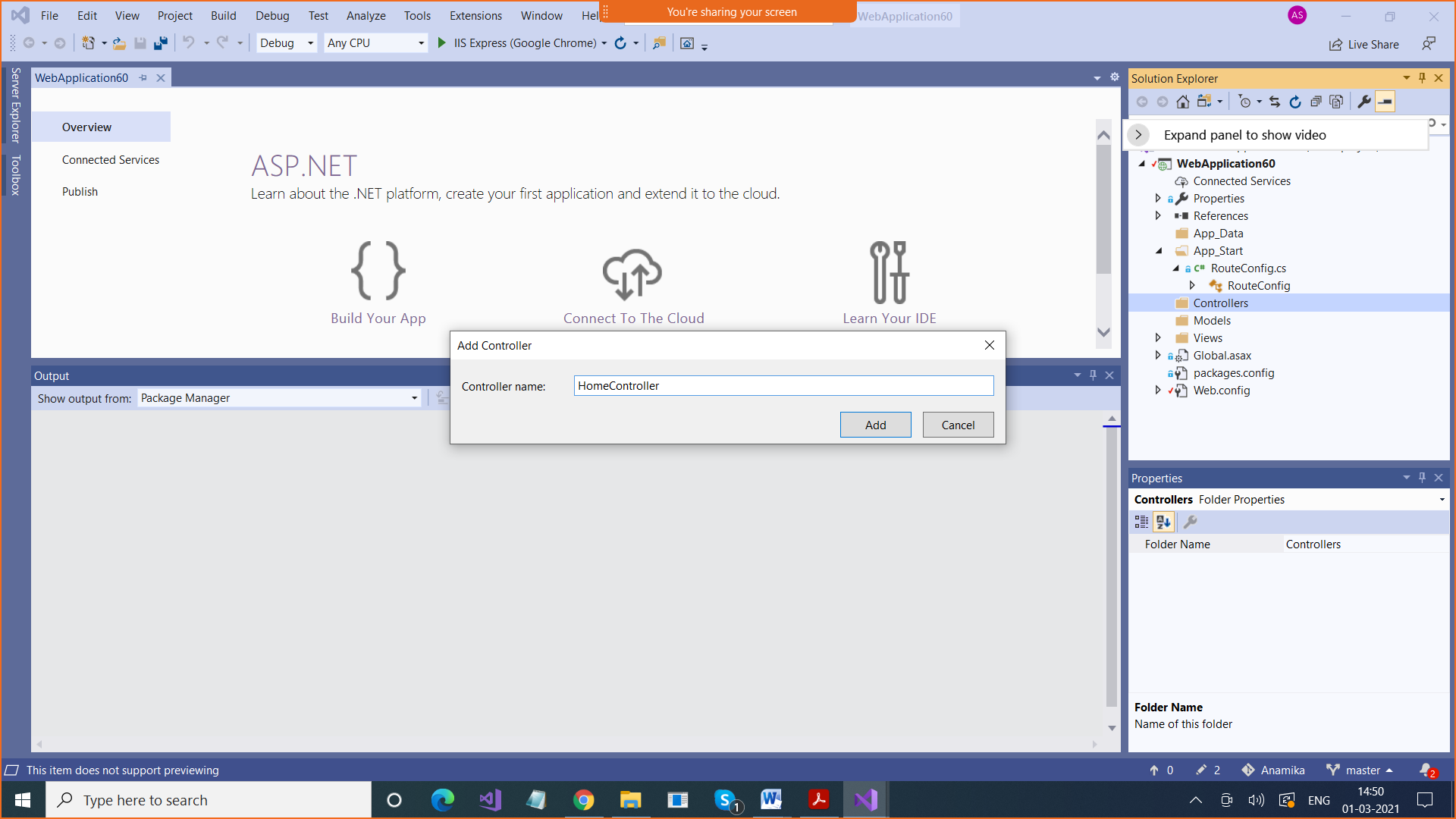
MVC application are light weigth











Path followed is <https://localhost:44319/Home/Method3>

URL / Controller /ActionMethodName

/home/index

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace WebApplication60.Controllers

{

**public class HomeController : Controller**

{

// GET: Home

public ActionResult Index() > **ACTION METHOD**

{

return View();

}

}

}

