The idea

- Create the infrastructure repository to implement a generic repository with unit of work class.
- Use Raven-DB is a No-Sql database to makes it light and easy to use with repository.

Efficiency:

- Clearly code and simple flow process.
- Optimize performance when use Raven-DB to access data
- Easy to maintenance and upgrade, integration.

Apply

In the Shipping Service project:

- **Step 1**: Create interface `IRepository,` a repository class including CRUD function

```
public interface IRepository<T> where T : class
    {
        void Add(T entity);

        void Add(IEnumerable<T> entities);

        void Update(T entity);

        void Delete(T entity);

        void Delete(Expression<Func<T, bool>> where);

        T GetById(long id);

        T GetById(string id);

        T Get(Expression<Func<T, bool>> where);

        IEnumerable<T> GetAll();
    }
```

Step 2: Create abstract class `EntityRepository` to perform implement CRUD generic, that use Raven-DB to stored object model class.

```
public abstract class EntityRepository<T> : RepositoryBase<IDocumentSession> where T :
class
    {
        protected EntityRepository(IDataContextFactory<IDocumentSession>
dataContextFactory) :
            base(dataContextFactory)
        {
        }

        public virtual void Add(T entity)
        {
            DataContext.Store(entity);
        }

        public virtual void Add(IEnumerable<T> entities)
        {
            foreach (var entity in entities)
                DataContext.Store(entity);
        }

        public virtual void Update(T entity)
        {
            DataContext.Store(entity);
        }

        public virtual void Delete(T entity)
        {
            DataContext.Delete(entity);
        }

        public virtual T GetById(long id)
        {
            return DataContext.Load<T>(id);
        }

        public virtual T GetById(string id)
        {
            return DataContext.Load<T>(id);
        }

        public virtual IEnumerable<T> GetAll()
        {
            return DataContext.Query<T>().ToList();
        }

        public virtual T Get(Expression<Func<T, bool>> where)
        {
            return DataContext.Query<T>().Where(where).FirstOrDefault();
        }
    }
```

Step 3: Create interface `IShippingRepository` to inherit `IRepository<Basket>.`

```
namespace Raven.Service.ServiceInterface
{
    public interface IShippingRepository : IRepository<Basket>
    {
    }
}
```

Step 4: Then, we have `ShippingRepository`. This class will inherit all method from interface `IShippingRepository` and abstract class `EntityRepository`

```
namespace Raven.Service.Implementation
{
    public class ShippingRepository : EntityRepository<Basket>, IShippingRepository
    {
        public ShippingRepository(IDataContextFactory<IDocumentSession>
dataContextFactory) : base(dataContextFactory)
        {
        }
    }
}
```

In the Shipping Web project

Step 5: Create a Controller allow other controllers inherit two methods execute of web app.

```
public class ShippingFactoryController : Controller
    {
        protected override void OnActionExecuting(ActionExecutingContext filterContext)
        {
            if (filterContext.IsChildAction)
                return;
            base.OnActionExecuting(filterContext);
        }

        protected override void OnActionExecuted(ActionExecutedContext filterContext)
        {
            if (filterContext.IsChildAction)
                return;
            if (filterContext.Exception == null)
                RavenSessionFactory.DisposeSession();

            base.OnActionExecuted(filterContext);
        }
    }
```

Step 6: Perform call interface service into Shipping controller

```
namespace RavenWeb.Controllers
{
    public class ShippingController : ShippingFactoryController
    {
        private readonly IShippingRepository _shippingRepository;

        public ShippingController()
        {
            _shippingRepository = new ShippingRepository(new RavenSessionFactory());
        }

        public ActionResult Index()
        {
            var model = _shippingRepository.GetAll().ToList();
            return View(model);
        }

        public ActionResult Detail(int shippingId)
        {
            var model = _shippingRepository.GetById(shippingId);
            return View(model);
        }

        public ActionResult Add()
        {
            return View();
        }

        [HttpPost]
        public ActionResult Add(Basket entityProduct)
        {
            if (ModelState.IsValid)
                _shippingRepository.Add(entityProduct);
            return Redirect("Index");
        }

        [HttpPost]
        public ActionResult Delete(Basket product)
        {
            if (product != null)
                _shippingRepository.Delete(product);
            return Redirect("Index");
        }
    }
}
```

Step 7: Initial Raven-DB database session. This class contains setting information and methods to execute when web app running.

```
namespace RavenWeb.RavenFactory
{
    public class RavenSessionFactory : IDataContextFactory<IDocumentSession>
    {
        private static IDocumentStore _store;
        private static IDocumentSession _currentSession;

        public IDocumentSession GetContext()
        {
            if (_store == null)
                Init();
            if (_store != null) _currentSession = _store.OpenSession();
            return _currentSession;
        }

        public void Dispose()
        {
            if (_store != null)
                _store.Dispose();
        }

        public static void Init()
        {
            var url = ConfigurationManager.AppSettings["Raven/Server/Url"];

            _store = new EmbeddableDocumentStore
            {
                ConnectionStringName = "RavenDB",
                DefaultDatabase = "Documentation",
                Url = string.IsNullOrEmpty(url) == false ? url : null,
                UseEmbeddedHttpServer = string.IsNullOrEmpty(url),
                RunInMemory = true,
                Conventions = {IdentityPartsSeparator = "-"}
            };
            _store.Initialize();
        }

        public static void DisposeSession()
        {
            if (_currentSession != null)
            {
                _currentSession.Dispose();
            }
        }
    }
}
```

And don’t forgot call this in Global.asax file. Finally, press F5 to running web application.

```
public class MvcApplication : HttpApplication
    {
        protected void Application_Start()
        {
            …//…
            RavenSessionFactory.Init();
        }
    }
```