STARTEAM EĞİTİMİ

Mesela bir projeye başladık. ERP olsun bu. Bunun dizin yapısı şu şekilde olmalı ;

StarTeam'de ERP diye bir proje açtıktan sonra

Mesela d:\Projeler\ERP

\Kaynak Kod

\Dokümanlar

\Kurulumlar olarak dizinlemeliyiz.

Sonrasında bu projede kullanılacak component olsun, dll olsun, exe olsun bunları farklı bir projede dizinlemek lazım.

Bunu da

\Kütüphaneler

\Component1

\DevExpress

\.dll'ler gibi dizinlemeliyiz.

Ve bu projeyi kullandığımız projeye **share** edeceğiz. Kullanıcılar olarak bu Kütüphaneler projesinde değişiklik yapabileceğiz.

Proje geliştirirken yeni versiyonlar çıkarabiliriz. Ama bu durumda **.dpr ve .dproj** dosyalarını kilitlememiz gerekir.Yoksa Projede değişiklik yapamayız.

StarTeam'de genelde yeni bir proje oluşturduğumuzda önce **Project Properties**'den gerekli ayarlamaları yapmalıyız. 3'lü checkbox'daki her şeyi seçmeliyiz. Ama .dpr ve .dproj dosyalarını read only yapmamalıyız.

Daha önce arşivden bahsetmiştik. Hash kodu **abcd43**… olan **check in** edilmiş bir text dosyası **Archives** klasöründe **ab** klasöründe **c** klasörünün içinde sıkıştırılmış (.gz) olarak duruyordu. Yeni bir versiyon yazılıp **check in** edilince arasındaki farkı ekliyor zipin içine. Mesela 1.0 versiyonundan 1.2 versiyonuna geçerken aradaki farkları ekleyerek 1.2 versiyonuna güncelliyor.

Ama hazır versiyonları **Cache** klasöründe. O yüzden **check out** dediğimiz zaman 1.2 versiyonu Cache'de varsa direk getiriyor.

Sunucuda C:Projeler içinde bir proje var. Biz bunu kendi lokalimize alırken farklı bir yerde oluşturmak istiyorsak bunun için **View Properties**'den **Alternate** kısmına istediğimiz diziniz yazarız ve yazdığımız yerde sunucudaki proje ilgili dizin formatıyla oluşur. Mesela kodu yazdık, uygulama dosyasını (.exe) proje_update'e atmak istiyorsak Kurulumlar klasörünün **Properties**'ine girerek buradan proje_update klasörünün yerini belirtmeliyiz.

Eğer sadece 2 dosya için **View Label** oluşturmak istiyorsak o 2 dosyayı seçerek sağ tıklayıp sonrasında **Labels->New** seçeneğinden yapabiliyoruz.

Birden çok **revision label** ile bir **view label**'i ilişkilendirmek istiyorsak bütün dosyaları seçip sağ tıklayıp **Labels->Attach** seçeneğinden yapabiliyoruz.

Kesinlikle bir view oluşturup garantiledikten sonra **view**'den **Select** diyip mevcut view'i dondurursak (**freeze** seçeneği) sonradan yanlışlıkla yapılabilecek oynamaları engelleriz.

Mesela bir projede 5 parça var. Bir adet genel revision label tanımlayıp, projenin 5 parçasını oluşturanlar kendi revision labellarını genel revision label ile ilişkilendirirse, 1 seferde **view label'i** projenin 5 parçasına birden uyarlayabiliriz. Bu genel revision label'ı ortak kullandığımızdan freeze etmemeliyiz.

HAMDULLAH HOCANIN DERSİNDEN NOTLAR

Jedi adında Open Source bir Delphi kütüphanesi var. İndirip araştırmakta yarar var.

Pascal/Delphi Code Style Guide-> Kod yazma standartları, bunu da araştırmakta yarar var.

Delphi.about.com -> Yeni kullanıcılar için delphi adında bir bölüm var(**Visual Form İnheritance** kısmı detaylı bir şekilde anlatılıyor.)

REM Object , Rename Expert ve IB Expert(Firebird programı) adlı programları araştırmakta yarar var.

Database Normalization adında bir kurallar kümesi var. 3 adımlık bir kurallar bütünü

- Bir tabloda aynı alandan bir veri birden fazla çıkıyorsa göçer(tabloda alan tekrarı olmayacak)
- Bir kaydı tanımlayabileceğimiz minimum alan primary key'dir.

Mesela Delphi 2010 kısayolundan bir tane kopyalayıp kısayola sağ tıklayıp hedef(target) kısmına –r yazıp konfigürasyon adını yazarsak, Delphi buna göre açılır. Bir proje için DevExpress kullanıp başkası için kullanmayabiliriz.

GExpert adlı componenti indirmemiz yararlı olur(Component isimlendirmede standart kurallar belirleyip uygulayabiliyoruz.)

CTRL + K + C diyince alt alta aynı şeyden kopyalama imkanımız var.

CTRL + SHIFT + R ile bir **makro** kaydederiz, mesela bir metni düzenleyeceğiz. Aynı yazılarda ard arda işlemler yapacağımız zaman **CTRL + P** ile makroyu çalıştırıyoruz. Otomatik düzeltme yapıyor.

Regedit'ten kayıt defterinde **Delphi'de Known IDE Packages** içinde kullanmayacağımız bir şey varsa data kısmında başına alt çizgi (_) koyarak devre dışı bırakabiliriz. Böylece program daha hızlı bir şekilde açılabilir.

Project Options'da **Base**'in içinde **Release** kullanıcıya verilen, **Debug** ise breakpoint eklenmiş, daha çok şişkin ve zaman alan kısım var. O yüzden release modda yaparsak programı , daha hızlı ve az yer kaplayarak çalışır programımız.

Debug ve Release içine farklı özelikler koyup bu programın farklı durumlarda farklı şekillerde çalışmasını sağlayabiliriz.

```
{$IFDEF KULLANICI}
    Button1.Visible := false ;
{$ELSE}
    Button1.Visible := true ;
{$ENDIF}
```

Şeklinde düzenleyebiliyoruz.

ALT + Yukarı Ok ile bir sınıf (class) tanımına direkt olarak gidebiliriz.

İnherited İtem'larda (kalıtılabilir birimler) atadaki değişiklik çocuğu etkiler ama çocuktaki değişiklik atayı etkilemez.Çocukta bir değişiklik yaptıktan sonra atada yapılan bir değişikliğin çocuğu etkilemediğini görebiliriz.

Çocuk bir componente sağ tıklayıp **Revert To Inherited** komutunu çalıştırırsak atadan ilk oluşturulduğu halini alır.

Kalıtım (inheritance) kullanımında JAVA'daki gibi **protected** olarak procedure tanımlayabiliriz.Bu ilgili procedure yada function'u sadece kalıtılmış sınıflar görebilir diğerleri göremez demektir.**Virtual** ise JAVA'daki abstract olarak kullanılıyor.Yani ata sınıfta procedure tanımı boş bırakılır.Ama çocuktan çocuğa o procedure'in içi değişebilir.

Initialization yazarsak koda hiçbir formu create etmeden ilk olarak programda burayı çalıştırır.

MMX , Smart Inspect , EurekaLog adlı componentleri araştırmakta yarar var.Hamdullah Bey bu componentleri kullanıyor.

Class helper for dediğimiz zaman mesela bu işlemi cxGrid'e yaparsak, onun üzerine ekleme yapabiliyoruz. Çünkü cxGrid'i inherit edip yeni bir grid tanımlarsak cxGrid'deki bütün fonksiyonları çocuk sınıfta tanımlayıp doldurmamız gerekir. Bir de kullanmayacağımız fonksiyonlar varsa bu hem sistemden hem de performanstan zarar etmemize neden olur.