Dersi Veren : Alpay ERTÜRKMEN (Alpay@btgrubu.com)

CaliberRM

-Gereksinim Yönetimi

-İzlenebilirlik

-Dokümantasyon alanlarında bize yardımcı olabilecek bir program …

Versant adında ticari bir database kullanıyor arka planda ..

-Bir gereksinim çeşitli yöntemlerle doğrulanabilir olmalı , ölçülebilecek gereksinimler oluşturulmalı(Caliber'de ilgili gereksinimin **Validation** adında bir bölümü var.Bu bölüm , bu işi hallediyor.)

-**References** Bölümünde gereksinimle ilgili bir web url , text ya da dosya eklenebiliyor.Sadece referans atılıyor , dosyanın kendisi server'a atılmıyor.

-Yeni bir proje oluşturmak için Başlat->Programlar->CaliberRM->Administrator programı seçiliyor.Buradan projeye kimlerin giriş yapabileceği de seçilebiliyor.

-**Glossary** : Index tarzı bir durum.Projenin içinde olan yerleri gösteriyor.Mesela kısaltmalar glossary'si eklendiğinde KOM kelimesinin üzerine gelince KOM'un glossary'de tanımlı uzun hali görünüyor.

-**Document Factory** : Buraya önce bir **.dot** dosyası veriliyor (.dot dosyası word belgesi şablonudur , hazır antetli kağıtlar gibi)

-Sonrasında bağlanılacak server'ı seçiyoruz.

-İstediğimiz gibi gereksinim filtrelemesi yapabiliyoruz.(Önce filtre tanımlamak lazım.**Requirement Grid**'deki filtreyi kaydetmeliyiz.)

-Mesela sadece kabul edilmiş gereksinimleri yazdırabiliriz.

StarTeam

-Versiyon Yönetimi

-Konfigürasyon Yönetimi

-Değişiklik Yönetimi için kullanılan bir araçtır.

VSS : **Visual Studio** ile birlikte kullanılan bir versiyon yönetim aracıdır (ücretsiz)

CVS : Genelde **Unix** kabuk komutları yazanlar kullanıyor

SVN : ?

Git : **Unix** işletim sistemini geliştirenlerin kullandığı bir versiyon yönetim aracı ..

Klasörde **ünlem işareti** olması , o klasörün kendi bilgisayarımızda bulunmadığını gösteriyor (lokalde)

Dosyanın durumu(**status**) **missing**(kayıp) ise sağ tıklayıp **check out** diyoruz ve sunucudan(**server**) kendi makinamıza alıyoruz.Bu olaydan sonra dosyanın durumu **current**(güncel) oluyor.

Dosyaya çift tıklayınca **Open** action'u çalışıyor.

Bir dosyayı check out diyip kendi makinamıza aldıktan sonra üzerinde değişiklik yapınca dosyanın durumu **modified**(değişmiş) hale geliyor.

Üzerinde değişiklik yaptığımız dosyayı **check in** diyerek kendi bilgisayarımızdan sunucuya aktarıyoruz.Bu işlemden sonra dosyanın durumu tekrardan **current** oluyor.

Eğer bir dosyanın durumu **out of date** (zaman aşımına uğramış) ise , bu sunucudaki dosyanın güncel , bizdekinin eski olduğu anlamına gelir.**Check out** komutu ile kendi bilgisayarımızdaki dosyayı güncel hale getirmeliyiz.

Ekranın sol alt köşesinde mevcut **view**'in durumu var.Şu an çalışırsak **current** (güncel) modda oluyor. 3 sene öncesinde gidersek o anki durumu gösteriyor.

Dosyaya sağ tıklayıp arşiv özelliklerine bakınca orada dosyanın adının **hash**lenmiş hali bulunuyor.Mesela hash **abc1888** ile başlasın.bu dosya **archives** klasörünün içinde **ab** klasörünün içinde **c** klasörünün içinde **zip**li olarak bulunur.Bu da demektir ki eski dosyaların eski versiyonlarını da görebiliriz harddiskimizde.

Var olan projeye yeni bir dosya ekleyince bu dosya mevcut **view**'in içinde bulunmadığından durumu **not in view** olarak görünür.Dosyaya sağ tıklayıp **add files** komutunu kullandıktan sonra dosya sunucuya atılır ve durumu **current** olur.

Eğer **istemci** program içinden dosya silmeye kalkarsak , **sunucu**dan da siliyor dosyayı.Bu yüzden mümkün olduğunca **Windows explorer**'dan silmeliyiz projeden silinecek dosyaları.

Ben bir dosyada değişiklik yaparken başka birisi değişiklik yapıp bunu sunucuya atarsa bendeki dosyanın durumu **merge**(birleşme) olarak görünür.Bu durumda **check out** dememiz gerekir. Bundan sonra zaten program mevcutları birleştireyim mi diye soruyor.Biz de birleştirme işlemini yaptıktan sonra **check in** komutunu kullanınca bizim yaptığımız değişiklikle birlikte dosyanın son halini sunucuya atmış oluyoruz.

Bir dosyanın **history** özelliğinden var olan 2 versiyonunu seçip sağ tıklamadan sonra **compare contents** dersek 2 versiyonun karşılaştırmasını yapıyor.

Benzer şekilde **Tools** menüsünden **Compare** özelliğini seçince alt tarafta lokal makinadaki son hali ve dosyanın o anki son hali karşılaştırılıyor.Tabi istediğimiz versiyonu seçerek onu da otomatik karşılaştırma ve farklarını görme imkanımız var.

Project Properties menüsünden **Options** kısmına girip alt taraftaki üçlü **checkbox**'daki **exclusive lock** özelliğini seçersek bundan sonra kilitlemediğimiz dosyayı (**lock**) düzenleyip **check in** komutu ile sunucuya atamıyoruz.Bir altındaki seçeneği seçtiğimiz zaman kilitlemediğimiz dosyaları otomatik olarak **salt okunur** (**read only**) hale dönüştürüyor.

Access Rights özelliğinden kullanıcı seçip , onlara özel haklar verilebiliyor.Ayrıcalık tanımlamaya yarıyor.

2 projeyi aynı anda açıp bir dosyayı **CTRL** tuşuna basarak diğerinde bir klasöre sürüklersek dosyayı paylaşmış oluyoruz.Bunu CTRL tuşuna basmadan yaparsak dosyayı diğer projeye taşımış oluyoruz.

Herhangi bir kolona (mesela **status** kolonu) sağ tıklayıp **sort and group** dersek gruplama ve sıralama özelliklerini ayarlayabiliyoruz.**Filtreleme** için önce **SQL** gibi bir sorgu parametresi oluşturmak lazım(**query**). Sonra yeni bir filtre oluşturup sorgu parametremizi(query) bu filtrenin içine ekliyoruz.

Sistemin belli bir andaki durumuna **konfigürasyon** deniyor.İstersek projenin ilk açıldığı haline dönebiliriz.Ve eğer **check out** komutunu kullanırken **force check out** özelliğini kuşanırsak , kendi bilgisayarımızda ne değişiklik yaparsak yapalım sunucudaki dosyayı almış oluruz.

View menüsünden **Label** alarak dosyalara versiyon ataması yapılabiliyor.Bu etiketler ile istediğimiz konfigürasyona çok daha etkili ve güvenli bir şekilde ulaşabiliriz.Burada tarih yerine label seçebiliyoruz.Bir dosyanın label özelliğinden label bilgisini kaydırarak daha eski bir versiyonunun kullanılmasını sağlayabiliyoruz.Bütün dosyalar için bu şekilde kendi label sekmelerinde değişiklik yapılabilir.

Mesela bir projenin çalışan bir sürümün koyduktan sonra üzerinde değişiklik yaptık ve test ediyoruz.Ama çalışan her versiyona label atmak ve versiyonların çok numaralı olması ihtimaline karşı , **promotion state** diye bir şey tanımlayabiliyoruz.Çalışan projenin ait olduğu label'ı **View** menüsünün

Promotion kısmında seçerek oluşturulacak **Promotion State**'e bir isim veririz ve **Select Configüration** kısmından bu state'i seçerek o anki çalışan kısma dönebiliriz.

Labellar , Stateler , Saatler bunlar hep belli bir şeyi aramak için kullanılıyor.Bunu kullanıcı eliyle gireceğine hazır bulmalı , daha program açılınca proje seçilince burada çalışan bir label'a ya da belli bir saate ya da bir state'e girebilmeli.Bu yüzden mevcut **konfigürasyon**u bir view olarak kaydedip projenin içine gömüyoruz.Bunun için **view** menüsünde **new** diyoruz.Türünü seçerken **read only reference** dersek meydana gelen view'de değişiklik yapamayız.Diğerlerini seçersek onlar üzerinde değişiklik yapabiliriz.

Mesela bir program yazdık.Onun bir üst modelini yazarken , kullanıcı eski bir sürümde bir bug bulunca , ve aradaki zaman uzun olacaksa , dallara ayrılıp **bug**(hata)'ı temizleyip yeni sağlam sürümü sunup yeni versiyonu da o arada paralel olarak çıkarmaya çalışırız.Bu **dallanma** olayını viewlerle halledeceğiz.

Oluşturulan viewleri karşılaştırıp birleştirip yeni bir versiyon üretebiliriz.Bu iş için **view** menüsünden **Compare/Merge** seçilir.Bu viewlerdeki bütün dosyaları halleder.**Promote** seçeneği 2 taraftan merge yapar.**Rebase** ana hattan hedefe eksiklikleri aktarır.**Replicate** kaynağın aynısından başka bir yerde oluşturur.**Compare only** de sadece karşılaştırma yapar.

İşlem bittikten sonra klasörün içinde soru işareti varsa bu çözülmemiş bir şeyler olduğu anlamına gelir.**Merge Status** eğer **unchanged** ise herhangi bir değişiklik yapılmamış , eğer **resolved** olup **ignore** ise dosyalar farklı ama zaten dosya ana hatta güncel olduğundan değişikliğe gerek yok , ama **resolved** olup **merge** durumunda ise merge işlemi yapılması lazım demektir.

Bu konu başlığının adı esasen **VCM**. İlgili menüsü menü çubuğunda çıktığından buradan detaylı bilgi edinilebilir.

Eğer bir projenin sadece belli klasörlerini içeren bir **view** oluşturmak istiyorsak , **new** menüsünden **reference viewi** oluşturmamız gerekiyor.Ve bu reference viewde yapılan bir değişiklik , esas viewde de otomatik olarak görünüyor.

Folder menüsünde **audit** adında bir şey var.Burada kimin projede ne değişiklik yaptığı görülebiliyor.**Topic** kısmında C**aliber**'deki gibi bir dosya üzerinden tartışma yapılabiliyor. Yapılan tartışma konusunu cevaplama özelliği de var.

Requirement , **Change Request** ve **Task**'e **Process Item** deniyor.**Requirement**'e gereksinim yazıp koyuyoruz.**Change Request** sistemde var olan bir durumun düzeltilmesi ya da geliştirilmesi için yapılan isteklere deniyor.

Yapılması gereken iş , öncelikle bir **change request** oluşturmak , bu request'i bir **task**'e bağlamak ve bu task doğrultusunda kodlarda değişiklik yapmaktır.Kodu taska bağlayınca linklerinde otomatik olarak görünür.

Project Properties'den mesela **check in** faaliyetini sınırlandırabiliriz.Yani **task**'i bitirdikten sonra ya da change request gerçekleşmeden check in yapma diyerek kullanıcıyı zorlayabiliriz.

MS Project ile birlikte kullanılırsa , MS Project ile hazırlanan görevler otomatik olarak **StarTeam** ile senkronize olur.**Task** kullanımı şimdilik zor ama c**hange request**'lere başlarsak bizim için son derece yararlı olacak.

Bir veya birden çok dosyada yapılan değişiklikler ile ilgili rapor almak istersek seçilen dosyaları sağ tıklayıp **reports** kısmından istediğimiz türü seçerek rapor çıkarabiliriz.Sadece dosyalar değil , **requirement**lar , **change request**ler , **task**ler , **audit**lerin de raporları alınabiliyor.

Ayrıca dosyaları çekip bunlardan **chart** ya da **grafik**ler de çıkartabiliyoruz.Dosyaya sağ tıklayıp **chart** seçeneğini seçersek grafikleri isteğe göre şekillendirip çıkarıyor.

Daha önceden label'ı kaydırıp , bazı dosyaların eski versiyonlarını kullanmıştık.Bunu aynı anda birden çok dosya için yapmak istiyorsak bunun için **revision label** kullanmalıyız.**Label** menüsünden oluşturulabiliyor.

Oluşturduğumuz yeni versiyonu **revision label** ile eşleştirmek için önce dosyaya **sağ tıklayıp select by label** seçeneğini seçiyoruz.Burada ilgili revision label daki bütün dosyaları gösteriyor.Sonrasında gene sağ tıklayıp **Label** seçeneğinden **attach** bölümünü seçip üst tarafa ilgili label'ı , alt tarafa da revision label'i seçerek ilgili revision label'lı dosyaların ilgili label'lara atanmasını sağlıyoruz.

Yeni bir dosya oluştururken bize revision label soruyor.Buradan seçince otomatik olarak **revision label** ataması yapıyor ilgili dosyaya.Dosya üzerinde değişiklik yapıp **check in** komutu ile sunucuya gönderdiğimiz zaman **revision label** menüsünün altında check box olarak *otomatik olarak revision label'ı bir üst versiyona taşıyayım mı* diye soruyor program.Eğer yazdığın kodun çalışırlığına güveniyorsan aktar diyorsun ve **yeni versiyon** ilgili revision label'a sahip olarak oluşuyor.