Sayfa 129

Çerçevenizi ve Testlerinizi tanımlayın

ÇERÇEVEM ve görüşmeci için nasıl açıklanacağı - Andy Lam ve biraz Alex

// **Veriye Dayalı ve Davranış Odaklı** - Karma çerçeveye dayalı

// **Maven** - yalnızca bağımlılık yönetimi için değil, aynı zamanda pom xml dosyasını kullanarak bir komut istemi aracı olarak oluşturma aracı, I ayrıca **dumanımı** çalıştıran belirli bir **xml dosyası** var,

// regresyon ve işlevsellik testleri

// Programlama dili olarak **Java** - çalışan ön uç, arka uç (api) ve veritabanı depolamak için **Java Koleksiyonu çerçevesini** kullanıyorum veri ve karşılaştır

// Ayrıca hassas / yeniden kullanılabilir verileri depolayan bir **özellikler** dosyam var - URL, şifre, tarayıcı

// ve hataları bulmak için verileri java olarak biçimlendirdikten sonra akışı kontrol etmek ve verileri doğrulamak için **TestNG** test aracını kullanıyorum

Ayrıca, yalnızca bir evrensel web sürücüsü oluşturmak ve kullanmak için **tekli desen** kullanan yardımcı program paketinde **Sürücü sınıfını** da oluşturdum.

FRONT END ; **selenium webdriver** ve çerçevemde tasarım **modelim** olarak **Sayfa nesne modelini** kullanıyorum ; - sayfa oluştur nesneler; webElements'i tanımlayın ve bir webelement değişkeni olarak depolayın,

POM = ELEMENT / YÖNTEM TABANLI SAYFA HEDEFİN YENİDEN KULLANILABİLİRLİĞİ

// Web öğelerimi @FindBy kullanarak somutlaştırmak için **Page Factory tasarım modelini** de kullanıyorum - daha kolay / kullanışlı

// yardımcı program; tarayıcı kullanımı - tarayıcıyı otomatikleştirmek için kodları kolaylaştırmak için hayatınızı kolaylaştıran statik yeniden kullanılabilir kod;

GERİ SONU (Api)

// kullanarak **RESTASSURED** süreci için sırayla çerçevesinde **deserialization** ve **serileştirme** oluşmaya

// **Json yanıtını** bir java toplama veri yapısında depolamanın (/ i yüksek seviyeli Pojolar ve harita nesneleri üretir) ve beklenen değere sahip veriler (ayrıca java veri yapısında saklanır)

// ayrıca bir api yardımcı program sınıfım var - yeniden kullanılabilir kodlar - bir satırın bir Pojo oluşturduğu yöntem

// ÇERÇEVİMİ DAVETMEDEN ÖNCE manuel test için **postacı** kullanıyorum

VERİTABANI testi

// SQL sorguları üretmek için manuel I **sql geliştiricisi**

// OTOMASYON İÇİN; oracle veritabanından **BAĞLANTI** alarak java'yı entegre etmek için **JDBC kitaplığını** kullanıyorum

// daha sonra SQL sorgularını kullanarak **STATEMENTS** oluşturmak ve ardından verileri bir **RESULTSET** nesnesine depolamak .

// İçindeki verileri depolamak ve karşılaştırmak için java veri yapılarını kullanıyorum

// ve **DATA DRIVEN ve CUCUMBER BDD** çerçevesini kullandığım için, bu testlerin tümü **özellik dosyalarının** içinde saklanıyor

// **ÖZELLİK DOSYASINDAN** kodlar oluşturmaya ve bunları b adlı bir dosyaya uygulamaya yardımcı olan **RUNNER sınıflarım** var

// ayrıca tüm testlerimden önce ve sonra çalışan kodlarımı uygulayan **HOOK sınıfı** da var - burası benim **Senaryo** arayüzünü kullandığımda tetiklenen **TAKESCREENSHOT** arayüzü (senaryo başarısız olduğunda)

// başarısız olan adımdayken fotoğraf çek

// SD - bu, **kornişon dilinin** beklenen değerini temel alan kodlarımı sakladığım yer

DDT

// az miktarda test verisiyle çalışıyorsam, **senaryo ana hatlarıyla** çalışacağım , burada **örnekler** oluşturuyorum ve ardışık düzen kullanarak verileri depolayın

// test verileri büyük miktarda i kullanma böylece onun genellikle harici dosyası (Excel) varsa **Apache POI** için **INVOKE DDT EXCEL OTOMASYON** ve excel dosyasından oku ve verileri sakla

// java veri yapısına

Sayfa 130

// Temelde yüksek riskli **kodlarımı** günlüğe kaydetmek için **log4j2** adında bir günlüğe kaydetme **aracım** da var

// ve son olarak raporlarım için, **çerçevemde** salatalık tarafından oluşturulan "rerun: target / rerun.txt" **salatasında Rerun.txt kodunu** kullanıyorum sandviç kütüphanesi

// bu başarısız salatalık özellik dosyalarımı depolayacak

// daha sonra başarısız senaryoların (rerun.txt) konumuna sahip olan **failScenario runner sınıfım** da var

// başarısız bir Senaryo xml dosyası oluşturuyorum

// bu yüzden başarısız özellik dosyalarım olduğunda mvn komutunu kullanıyorum; **mvn -Drunner = failScenarios.** Başarısız testlerimi çalıştırmak için **xml** dosyası

// raporlama - salatalık raporları adı verilen hedef klasörde bulunan html raporu kullandım - "html: hedef / salatalık-raporu"

Paralel test - koşucular oluşturmak için **salatalık jvm-paralel eklentisi** ve testleri çalıştırmak için **hata korumalı eklenti** kullandı

İçin **sürekli entegrasyon (Jenkins)**
// devops yapılandırmayı halleder
// github yolu var
// ancak araç bir mvn komutum çağrıldı - mvn doğrulayın -drunner = smoketest.xml test cihazı tarafından sağlanan - xml dosyası
// raporlar için her yapıda, testin grafik bilgilerini ve ekran görüntüsünü veren bir salatalık raporu olacaktır

Kaç test durumu / senaryosu / özellik dosyası: Sprint başına 23 test durumu, 150 özellik dosyası, 400 senaryo Duman testim
2 özellik dosyası, her biri 2 senaryo, günde bir kez 5 dakikalık duman testi yapılır.

Benim gerilemem
150 özellik dosyası
400 senaryo
3 yıl ; 1 test kullanıcısı 3 yıl çalıştı, ikincisi 2 yıl çalıştı

Duman testiniz ne sıklıkla, ne kadar sürer ve neyi test eder? Her sabah saat 5'de, yaklaşık 5 dakika, 5 öğrenci oluşturur
gerekli alanlara sahip iki tarayıcıda, hem kullanıcı arayüzünden hem de veritabanından doğru şekilde kaydedilip kaydedilmediğini kontrol eder, ardından raporu
tarayıcı ve ayrıca bir excel dosyası olarak indirin ve ardından karşılaştırın.

Regresyon testiniz ne sıklıkla, ne kadar sürer ve neyi test eder? Haftada bir 1am, yarım saat, tüm ilçeler her gün,
sadece SIMS. Hafta sonları doğrulama regresyon testleri, 5 ayrı EC2 makinesi, her okul için paralel test, başvuru,
doğrulama, çapraz doğrulama ve sertifikasyon. Testimiz SCS ile doğrulanıyor.

Örnek Senaryo: SIMS raporunun indirildiği ve dosya türünün bir Excel Dosya türü olduğu göz önüne alındığında, ilk sütun başlığı
"LASID" ve ilk sütundaki her hücre alfanümerik olmalı ve ilk sütundaki her hücre 9 basamaklı ve tümü LASID olmalıdır
sayılar benzersiz olmalıdır.

Örnek Test Vakası: Kullanıcının oturum açtığı göz önüne alındığında, SIMS raporu yürütüldüğünde, ilk sütundaki her hücre 9 olmalıdır.
rakamlar ve ayrıca alfanümerik.
Olmalıdır: ön koşul, adımlar, test verileri, beklenen sonuç, gerçek sonuç

Uç Durum Senaryosu: boş, negatif sayılar, boş liste / dize, yinelenen kontrol, limitleri kontrol etme, aşırı durumlar (uzunluk, boyut)

Risk temelli test: Tam regresyon testi yapmak için zaman olmadığında, yalnızca önemli olan, ilgili kısımları test edersiniz.

Gereksinimler olmadan test etme: Üretim kusurlarının genellikle herhangi bir gereksinimi yoktur ve
durumu daha iyi ve sonra test edin.

Sayfa 131

Aşırı yükleme yöntemi için örnek : Beklemeler için BrowserUtils'te çeşitli aşırı yüklenmiş yöntemler. Konum belirleyiciye göre açık bekler veya
WebElement.
Geçersiz kılma yöntemi örneği: Aşağıda

Miras / soyutlama örneği : Üst çubuk ve kenar çubuğu sayfaları soyut sınıflardır ve bazı yöntemler soyuttur
çünkü nerede olduklarına ve nereye tıkladıklarına göre farklı uygulamalarımız var. Okul / Bölge işlevleri
açılır menülerde farklı şekilde doldurulur.