

AWS

1. AWS ile mi çalışıyorsunuz?

- EC2 bulut sunucularıyla çalışıyorum.
- Temel olarak, bu benim sanal makinelerim.
- Selenium Grid'e sahip olduğumda, farklı sanal makinelerim oluyor ve her makinem ayrı çalışıyor.
- Örneğin, regresyon testleri için gereken zamanı en aza indirmek için gerçekten etkilidir, şirketimize çok zaman kazandırır.

2. AWS nedir?

- AWS, bulut sanal makinesi sağlıyor. Bir EC2 örneği oluşturun.
- Bu örneği uzak masaüstü ile kullanabilirim. Aslında, öğle yemeğimi yedikten sonra, normal bir bilgisayar gibi kullanıyorum.

3. Temel sayfa nedir?

- Ortak işlevlerimizi bir temel sınıfta depolar ve daha sonra bu temel sınıfı genişletir ve diğer sınıfta kullanırız.

Sayfa 126

Çerçevenizi sıfırdan nasıl oluşturabilirsiniz?

- Test Çerçevesi - test senaryoları oluşturmak ve tasarlamak için kullanılan yönergeler ve kurallar

1. Ortamı kurun; JDK, MAVEN, ECLIPSE IDE'yi kurun

2. Maven projesi oluşturun

- ArtifactID - projenizin adı
- GroupID - projenizi tüm projelerde benzersiz bir şekilde tanımlar

3. Maven Deposundan Bağımlılıklar Ekleyin

- Selenium Java
- Salatalık Sandviç
- <özellikler> içinde JRE Sistem kitaplığı 1.8
- TestNG
- WebDriverManager_BoniGarcia

- Salatalık Java
- Salatalık TestiNG
- Apache POI.XML
- APACHE POI
- JDBC
- Huzurlu
- Gson
- Log4j

4. Çerçeve Yapısı (paketler) oluşturun

- Sayfalar
 - ben. Web öğeleri ve yöntemler
- POJO / Fasulye
 - ben. Özel sınıflar
- Koşucu
 - ben. Cuckesrunner - kodlar üretir ve html raporunu hedefe kaydeder
 - ii. Sigara içmek
 - iii. Regresyon
- StepDefinitions
 - ben. Gerçek kodlar ve kanca sınıfı
- Testler
 - ben. Veriye dayalı testler
- JDBC
- API
- Yardımcı Programlar
 - ben. ConfigurationReader
 - ii. Sürücü sınıfı (Singleton)
 - iii. browserUtils
 - iv. DBUtils
 - v. ApiUtils
- Configuration.properties
- Testng_runner.xml
 - ben. Runner sınıfından biri Paketleri com.app.utilities ile çağırın

126

Sayfa 127

5. Kaynaklarda unsur dosyası (.feature) ile unsur klasörü oluşturun

- Kornişon bir dilde yazılmış yürütülebilir dosya özelliği

6. Senaryonuzu yazmayı bitirdikten sonra , cuckesRunner'ınızı dryRun = false ile çalıştırın, bu size StepDefinition sınıfında depolayacaksınız

7. Kodunuzu nasıl çalıştırırsınız?

- Koşucu sınıfını kullanın - kodları çalıştırın ve salatalık raporu ve html raporları oluşturun

8. Çerçeveyi GitHub veya SVN'ye gönderin

- Yeni depo oluşturun
- git url'yi kopyalayın ve tutulmaya gidin
- git deposunu yapılandırın ve url'yi ekleyin
- Projeyi sağ tıklayın, ekibi tıklayın, tamamlamayı tıklayın ve git aşamasına geçeceksiniz
- ARTIK KODUNUZ GITHUB'DA

9. Sırada JENKINS ENTEGRASYONU var

- Jenkins
 - ben. Açık kaynak otomasyon sunucusu
 - ii. Yazılım geliştirme sürecinin insan dışı kısmını otomatikleştirmeye yardımcı olur
 - iii. Sürekli entegrasyona izin verir
 - iv. Düzen aralıklarında kodu paylaşılan bir depoya entegre etmek için geliştirici gerektiren geliştirme uygulaması
 - v. Bağlantı noktası 8081 localhost'tur
- Projenizi jenkins'te çalıştırmak için
 - ben. Jenkins hesabına giriş yapın
 - ii. Proje oluşturun - serbest stil
 - iii. Eklentileri yükleyin - salatalık raporu ve git
 - iv. Kaynak kodu yönetimi altında git ve geçmiş git url'sini seçin
 - v. Tetikleyici oluşturun - düzenli aralıklarla oluşturmaya seçin
 - vi. Üst düzey maven'i çağırın
 1. Maven sürümü; MAVEN_HOME
 2. Hedefler; temiz doğrulama -Drunner = smoke_runnerxml

- vii. Derleme sonrası eylemler altında
 - 1. Salatalık raporlarını seçin
 - 2. Düzenlenebilir e-posta bildirimini seçin
- viii. Düzenlenebilir e-posta bildirimi
 - 1. Derleme günlüğünü ekleyin; derleme günlüğünü seçin
 - 2. Gelişmiş ayarları tıklayın
- ix. Arıza-Herhangi
 - 1. Gelişmiş'i tıklayın
 - 2. Alıcı listesi - raporu alacak e-posta adresi. Birden çoksa virgül ekleyin
 - 3. Her zaman hata gibi tetikleyici ekle'yi tıklayın
 - 4. Derleme günlüğünü ekleyin; derleme günlüğünü ekle'yi seçin
 - 5. Kaydet
- x. Son adım
 - 1. Şimdi oluştur'a tıklayın ve test çalışacak ve salatalık raporunuzu verecektir

Sayfa 128

Yapımdan sonra salatalık raporu hakkında konuşun

1. Sayfa Nesne Modeli nedir

- a. Kod fazlalığını azaltır ve kodu düzenler
- b. Öğeleri tanımlamaya ve bir sayfa nesnesi değişkeni olarak depolamaya yardımcı olur
- c. Saklandığı yere bağlayabilirsiniz
- d. PageFactory tasarım deseni eklendi

2. Otomasyon aracım olarak Selenium WebDriver

- e. Önce manuel olarak test edin:
 - ben. Başlangıç aşaması
 - İşlevsel test
 - ii. Arka uç
 - Veritabanı - SQL Developer IDE
 - API - Postacı
- f. Selenium'u aşağıdakilerle entegre edin:
 - ben. Uzman
 - Paketi test edin
 - Yardımcı program paketi
 - a. UI
 - b. DB
 - c. API
 - Yapılandırma dosyası
 - a. Özellikleri
 - Sürücü sınıfı
 - a. Singleton tasarım deseni
 - a. Özel bir kurucuya sahip olmak
 - ii. Salatalık BDD
 - BDD süreci sırasında işbirliğini kolaylaştırın
 - Hikayeyi ve kabul kriterlerini kolay bir dille açıklamayı sağlar.
 - iii. Git - kaynak denetimi
 - iv. Jenkins
 - v. Java
 - Koleksiyon Çerçevesi
 - Apache İÇN
 - JDBC
 - Huzurlu

3. Davranış Odaklı Geliştirme

- Müşteri için doğru standartları karşıladığından emin olmak için müşteri ile birlikte gelişmek

4. Veriye Dayalı Geliştirme

- Farklı veri kümelerine karşı aynı test senaryosunun yürütülmesi
- Test akışı verilere göre değişmemelidir

5. Salatalık raporlama

- Hedef klasör
- Jenkins

Sayfa 129

Çerçevenizi ve Testlerinizi tanımlayın

ÇERÇEVEM ve görüşmecii 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 karşılaştırm
// 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 oluşturma nesneleri; webElements'i tanımlayın ve bir webelement değişkeni olarak depolayın,

POM = ELEMENT / YÖNTEM TABANLI SAYFA HEDEFİN YENİDEN KULLANILABİLİRLİĞİ

```
// Web öğelerini @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 deserializasyon 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 olarak 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 aracı da var

// ve son olarak raporlarımı 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 - koşullar 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 komutunu çağırırdı - 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 indirir ve ardından karşılaştırır.

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.