Dunnington Software Development Incorporated

Moving At The Pace Of Integrity!

SSA KONULARI PLANI ÇALIŞMA REHBERI

SSA 24.2 SERTIFIKASYON KONULARI

DUNNINGTON SOFTWARE DEVELOPMENT INCORPORATED

Dunnington Software Development Incorporated

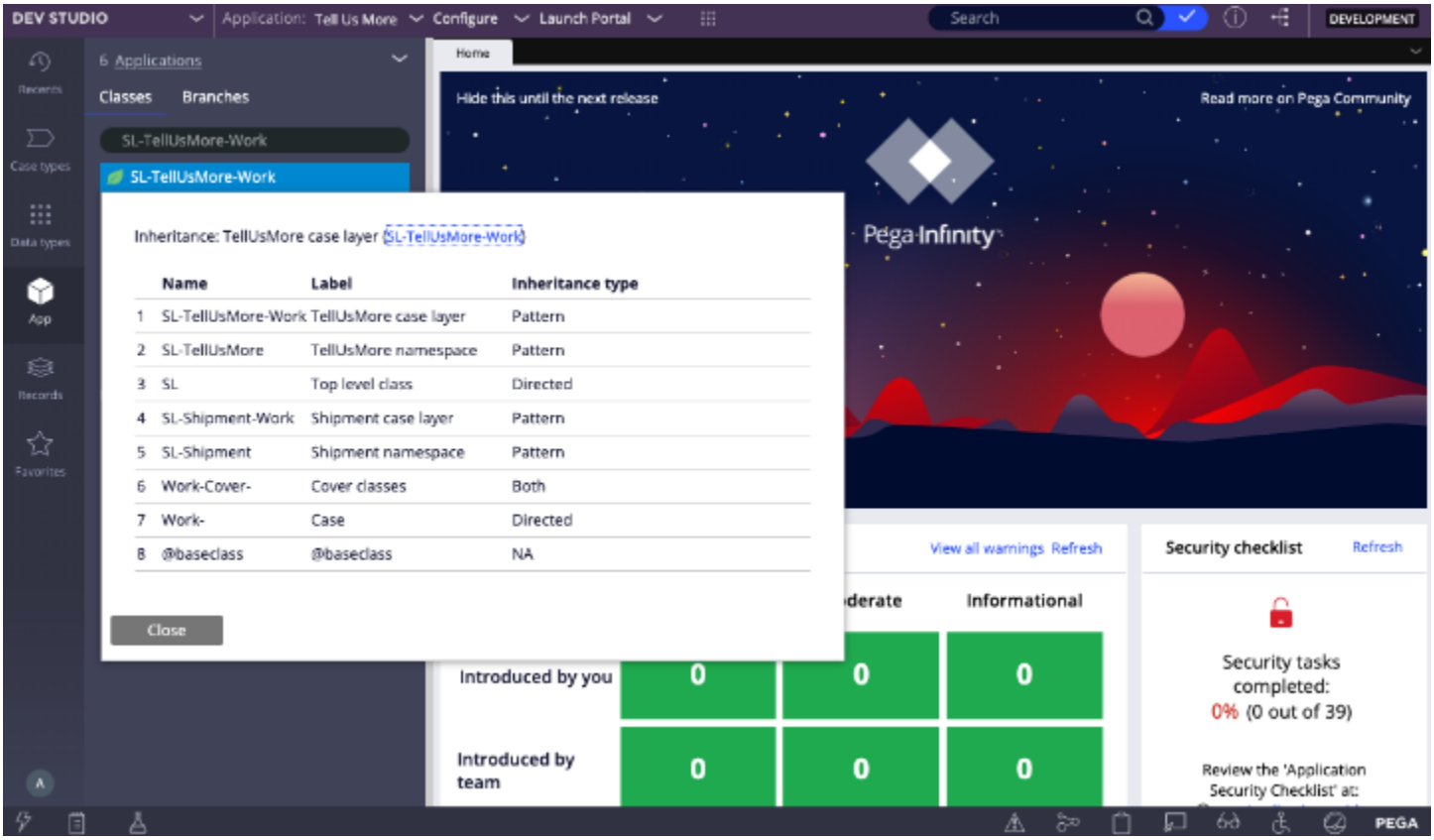
Moving At The Pace Of Integrity!

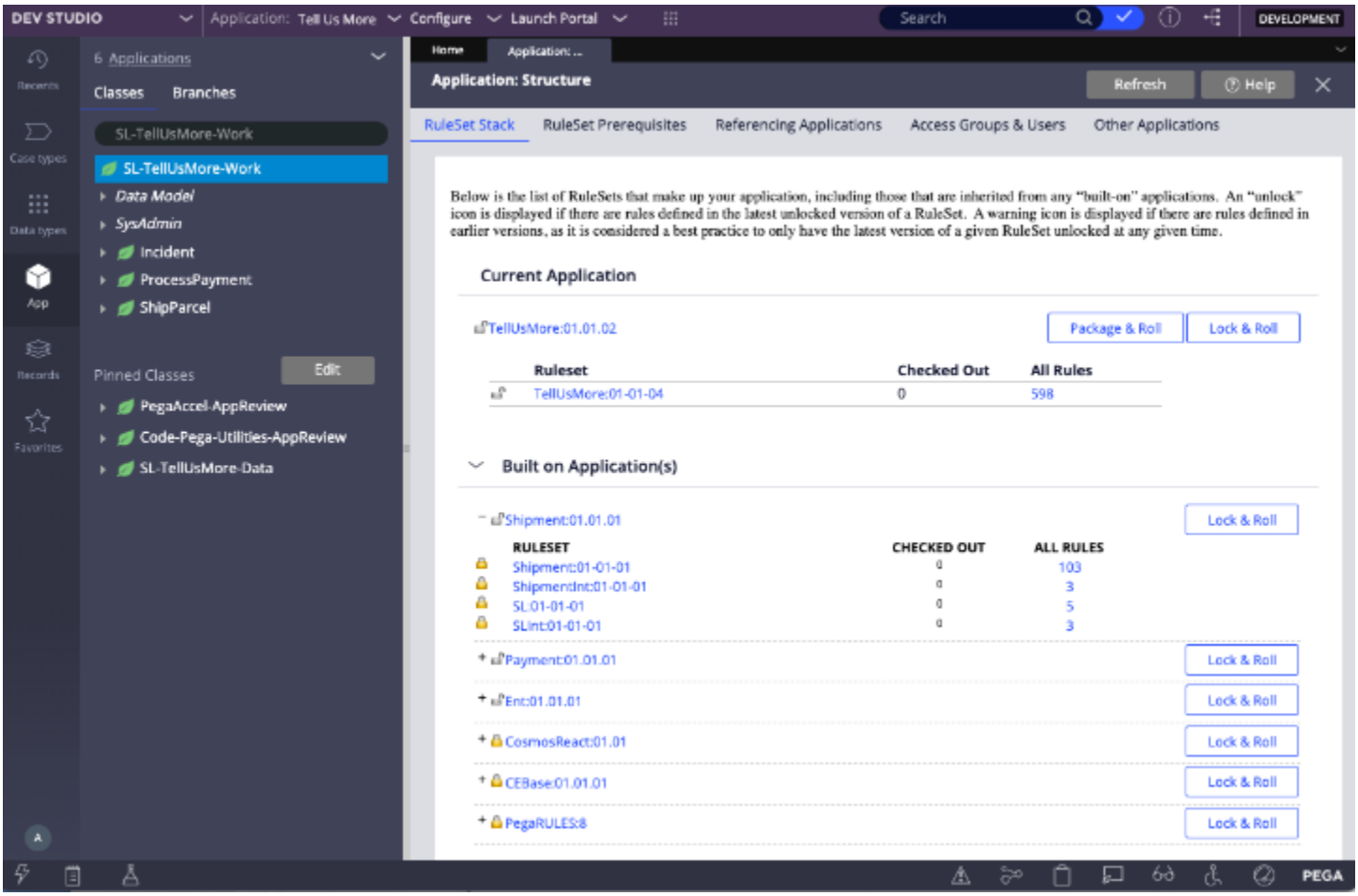
Soru: Enterprise Sınıf Yapısı nedir ve kural yeniden kullanımı ile polimorfizmi nasıl teşvik eder?

TANIM/TANIM(LAR): Kurumsal Sınıf Yapısı (ECS), Pega'da katmanlı bir sınıf hiyerarşisidir ve Pega Platform'un nesne yönelimli mimarisinin temel bir parçasıdır. Tek bir kuralın genel bir sınıfta oluşturulmasına ve daha spesifik çocuk sınıflarda yeniden kullanılmasına izin vererek polimorfizmi teşvik eder. Bu yeniden kullanım**,** miras yoluyla sağlanır; burada alt sınıflar, ana sınıflarında tanımlanan kuralları miras alıp uzmanlaştirabilir. ECS dört anahtar katmandan oluşur:

- **Pega Platform katmanı:** Pega Platformu için temel kuralları içerir.
- **Organizasyon katmanı:** Tüm şirket veya organizasyon için kuralları içerir.
- **Bölüm katmanı:** Organizasyon içindeki belirli bölümler için kuralları içerir.
- **Çerçeve ve Uygulama katmanları:** Belirli uygulamalar için kurallar içerir.

KONUM/RESMİ(ler): Enterprise Sınıfı Yapısı, temel bir mimari kavramdır ve belirli bir esere tek bir navigasyon yolu içermez. Pega uygulaması içindeki sınıf hiyerarşisi ile tanımlanır. Yapı, yeni bir sınıf veya kural oluşturulduğunda görünürdür.





KULLANIM DURUMU/UYGULAMA(lar): ECS, kuralların yeniden kullanılabilir olmasını ve uygulamaların kolayca genişletilip bakımının yapılmasını sağlamak için kullanılır. Örneğin, bir bankacılık uygulamasında, Organizasyon katmanında tüm müşterilere uygulanan kurallarla genel bir "Müşteri" sınıfı tanımlayabilirsiniz; bu sınıf ürünlerine bakılmaksızın tüm müşteriler için geçerlidir. Uygulama katmanında, genel "Müşteri" kurallarını devralan ama aynı zamanda ipotek müşterileri için özel kuralları olan daha spesifik bir "İpoteki" sınıfı olabilir. Bu yaklaşım, farklı müşteri türleri için aynı kuralı birden fazla kez oluşturmak zorunda kalmamanızı engeller, geliştirme süresinden tasarruf sağlar ve hata riskini azaltır.

Dunnington Software Development Incorporated

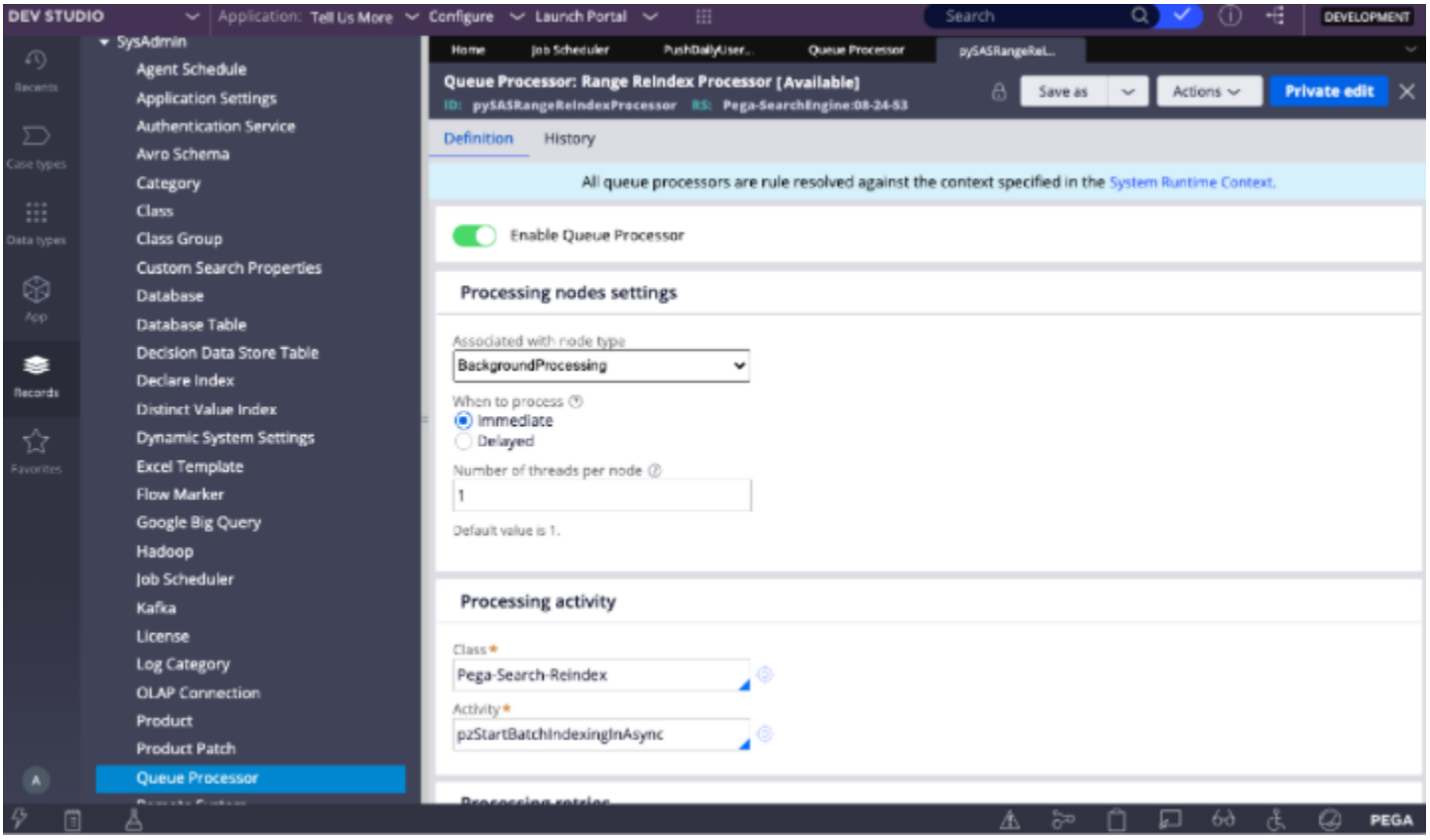
Moving At The Pace Of Integrity!

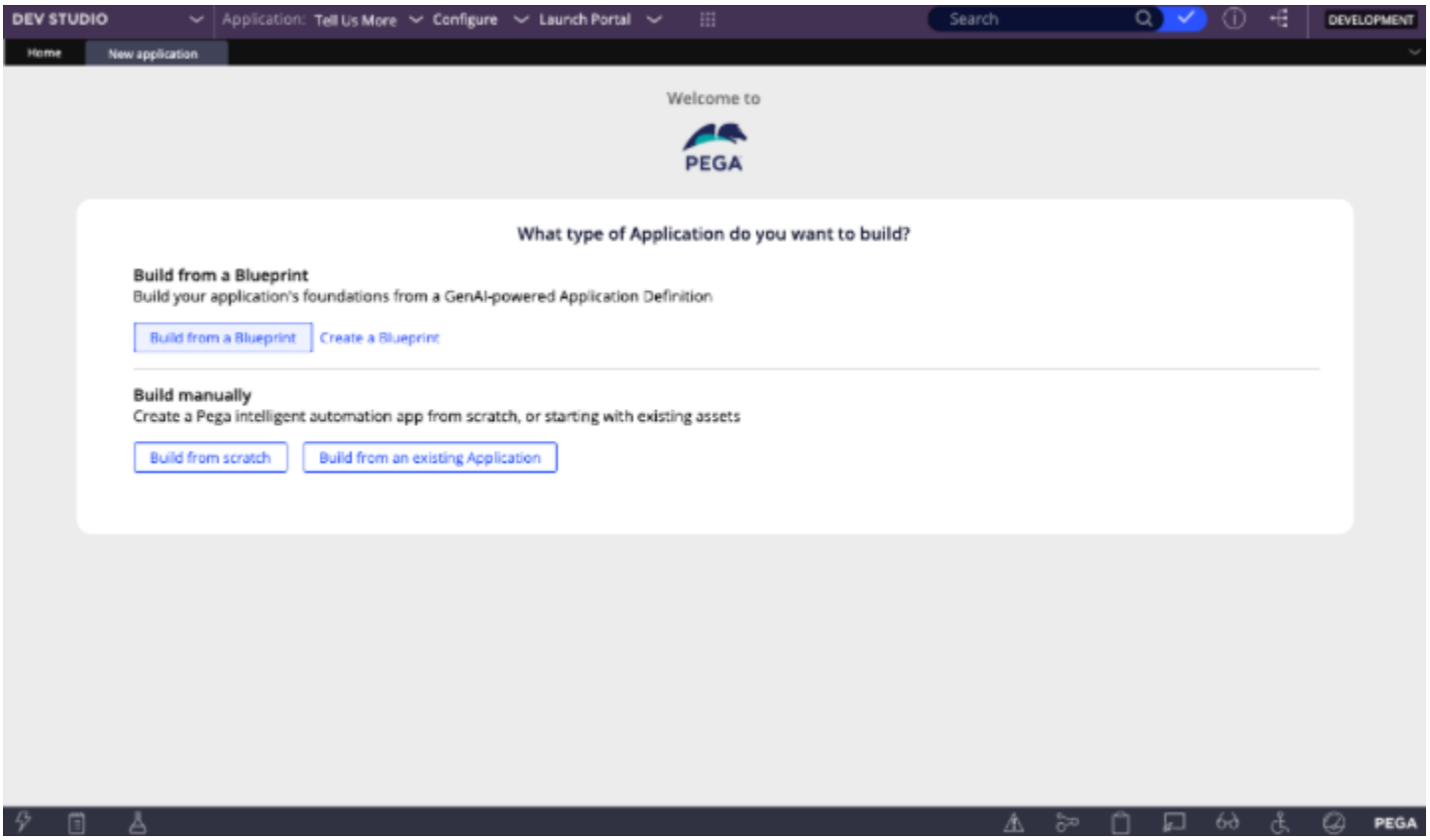
Soru: Yeni Uygulama sihirbazı kullanılarak bir uygulama nasıl oluşturulur?

TANIM/TANIM(LAR): Yeni Uygulama sihirbazı**,** geliştiricileri yeni bir Pega uygulaması oluşturma sürecinde rehberlik eden bir araçtır. Uygulamanın temel yapısını, adı, uygulama türü ve ilk erişim gruplarını tanımlamaya yardımcı olur. Sihirbaz, uygulama kuralı, başlangıç vaka tipleri ve varsayılan bir skin gibi temel uygulama artefaktlarını oluşturur.

KONUM/GÖRSEL(ler): Dev Studio'dan Yeni Uygulama sihirbazına erişmek için:

- Dev Studio**'ya gidin**.
- Yeni Başvuru**'ya tıklayın**.
- Uygulamaya isim vermek için sihirbazın adımlarını takip edin, bir uygulama türü seçin ve ilk rolleri ile kullanıcıları tanımlayın.





KULLANIM DURUMU/UYGULAMA(lar): Yeni Uygulama sihirbazı, herhangi bir yeni uygulama geliştirme projesinin başında uygulamanın temel yapısını hızlıca oluşturmak için kullanılır. Örneğin, yeni bir "Kredi Başlatma" uygulaması başlatmak için, bir geliştirici uygulama kuralı ve varsayılan "Kredi Başvurusu" vaka tipini oluşturmak için sihirbazı kullanır. Sihirbaz, gerekli sınıfları ve kural setlerini otomatik olarak oluşturur ve uygulamanın iş mantığını oluşturmak için sağlam bir başlangıç noktası sağlar.

Dunnington Software Development Incorporated

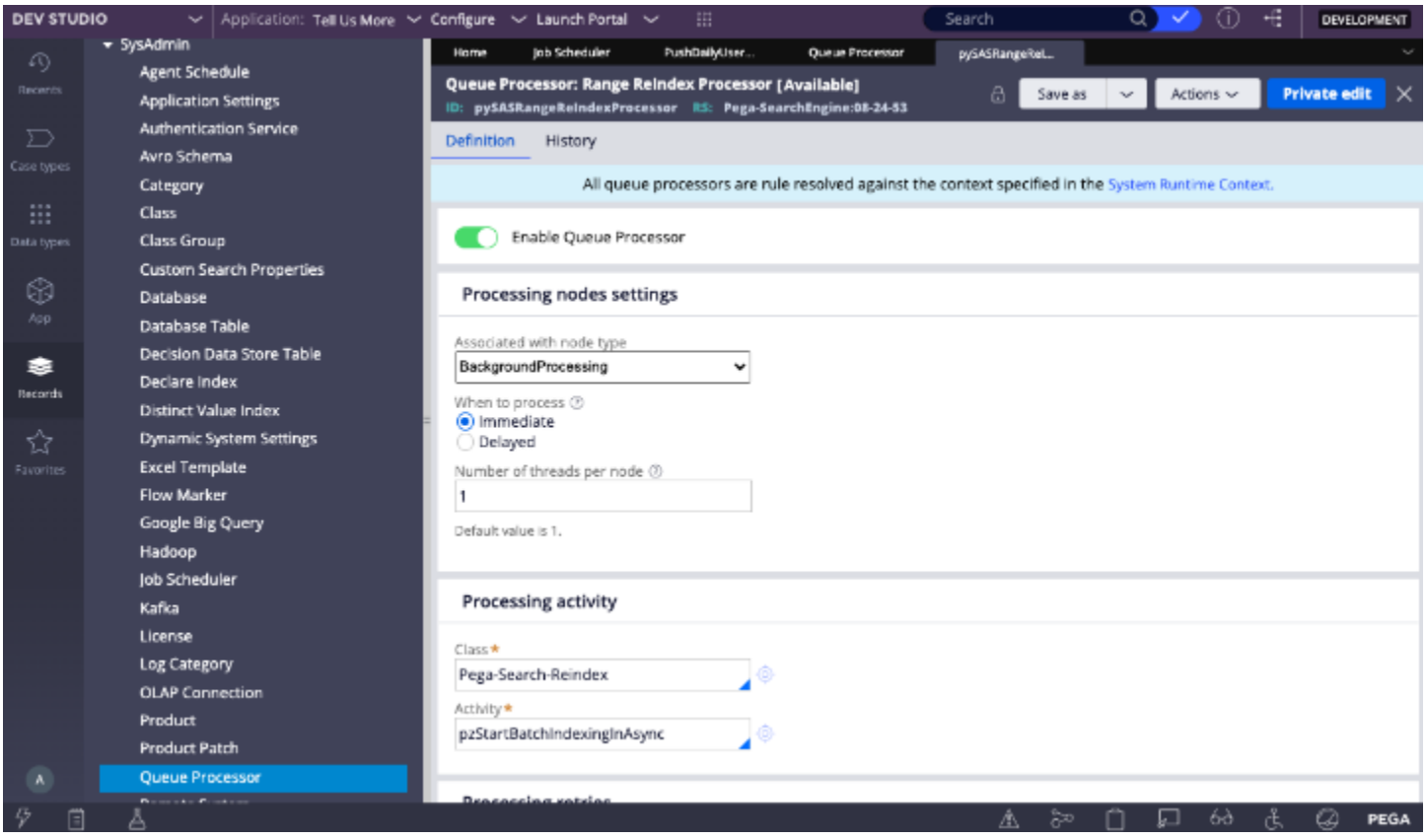
Moving At The Pace Of Integrity!

Soru: Kural çözümleme süreci nasıl işliyor ve kural erişilebilirliğini nasıl ayarlayabilirsiniz?

TANIM/TANIM(lar): Kural çözümü, Pega Platform'un belirli bir bağlam için en uygun kuralı bulma sürecidir. Pega, sınıf hiyerarşisi, kural listesi ve kural erişilebilirliği gibi faktörleri dikkate alarak en iyi kuralı bulmak için çok adımlı bir algoritma kullanır. **Kural erişilebilirliği**, kuralın sistem tarafından kullanılıp kullanılamayacağını belirleyen bir kural ayarıdır. Kullanılabilirlik durumları **Erişilebilir**, **Geri** Çekilmiş, **Engellenmiş** ve **Final**'dir.

KONUM/GÖRSEL(ler): Dev Studio'da kural kullanılabilirliğini ayarlamak için:

- Dev Studio**'ya gidin**.
- Kayıtlara **gidin**.
- İstenilen kuralı açın (örneğin, bir Akış Eylemi veya Bir Etkinlik).
- Kural formunda **Tarih** sekmesine gidin.
- **Kullanılabilirlik** alanında istenen durumu seçin (örneğin, Çekildi).



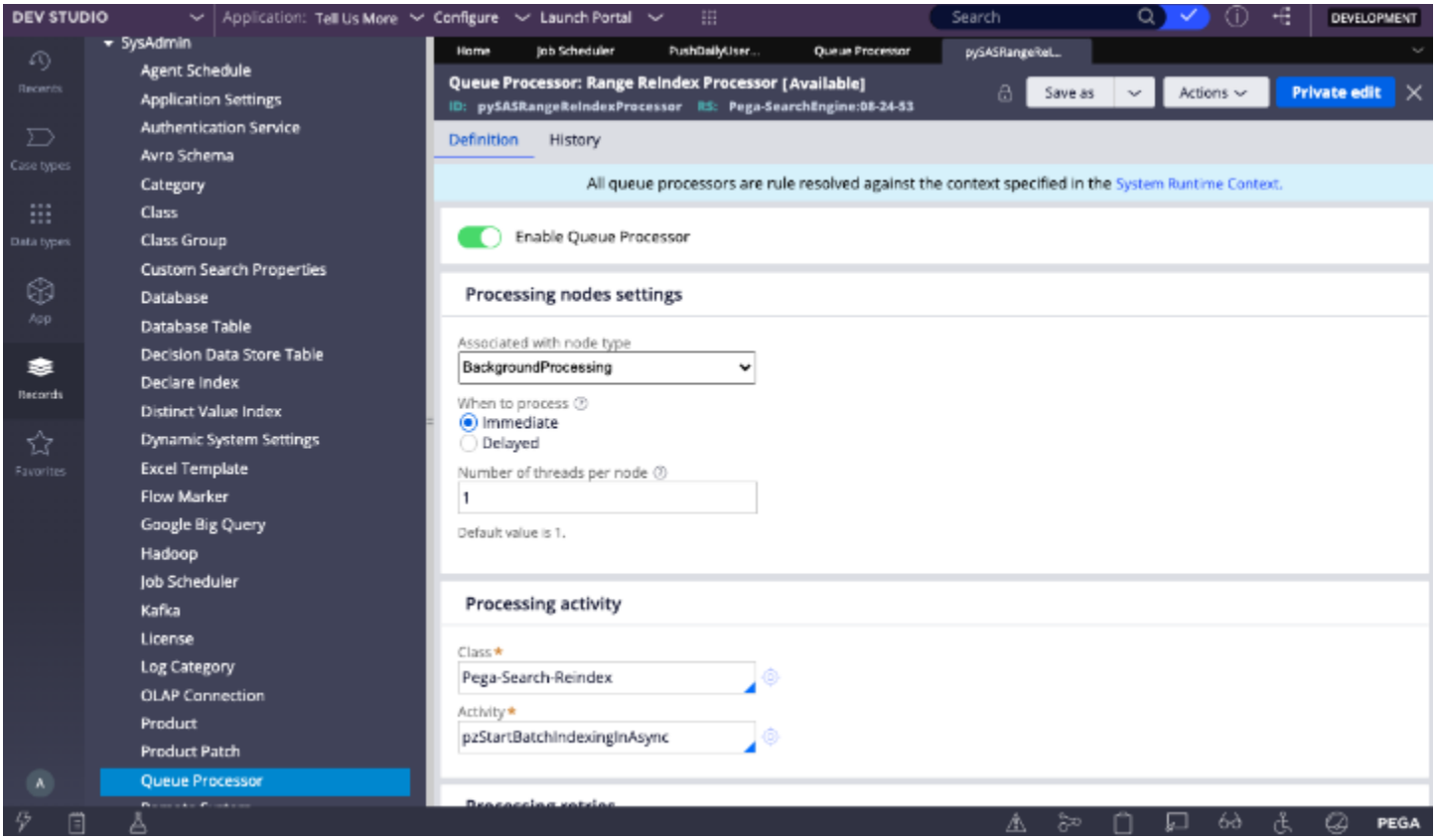
KULLANIM DURUMU/UYGULAMA(lar): Kural çözümü, Pega'nın iş mantığını nasıl yürüttüğünde temel bir parçadır. Yaygın bir kullanım durumu, eski bir kural versiyonunun artık gerekli olmamasıdır. Kuralın kullanılabilirliğini **Geri Çekilen olarak** ayarlayarak, kural çözüm sürecinin bunu görmezden gelip daha yeni, daha uygun bir kuralı seçmesini sağlayabilirsiniz. Bu, farklı kural sürümlerini yönetmek ve doğru davranışın her zaman uygulanmasını sağlamak için kritik bir uygulamadır.

Soru: Durum uygulama davranışını nasıl etkiler?

TANIM/TANIM(LAR): Durum belirleme , belirli koşullar altında çalışılan bir kuralın özel bir versiyonunu oluşturmanıza olanak tanıyan bir Pega özelliğidir. Kuralın durumlu versiyonu, temel kuralın ek koşullarla bir kopyasıdır ve yalnızca bu koşullar sağlandığında kullanılır. Şartlar sağlanmadığında, temel kural kullanılır.

KONUM/GÖRSEL(ler): Dev Studio'da bir kuralı kondurmak için:

- Dev Studio**'ya gidin**.
- Durumu belirlemek istediğiniz temel kuralı açın.
- Araç çubuğunda, **kuralın yeni bir versiyonunu** oluşturmak için Save As seçeneğini seçin.
- Kaydet **olarak** diyalogundan Durum **seçeneğini seçin.**
- Bir mülk değeri veya tarih aralığı gibi durum koşullarını yapılandırın.



KULLANIM DURUMU/UYGULAMA(lar): Koşullar belirleme, belirli koşullara göre birden fazla ayrı kural oluşturmadan değişen uygulama davranışını yönetmek için kullanılır. Yaygın bir kullanım durumu, gönderim ücretinin yurtiçi ve uluslararası siparişler için farklı olduğu bir gönderim uygulamasıdır. Her biri için ayrı bir kural oluşturmak yerine, yurtiçi ücreti hesaplayan bir temel kural oluşturabilir ve ardından

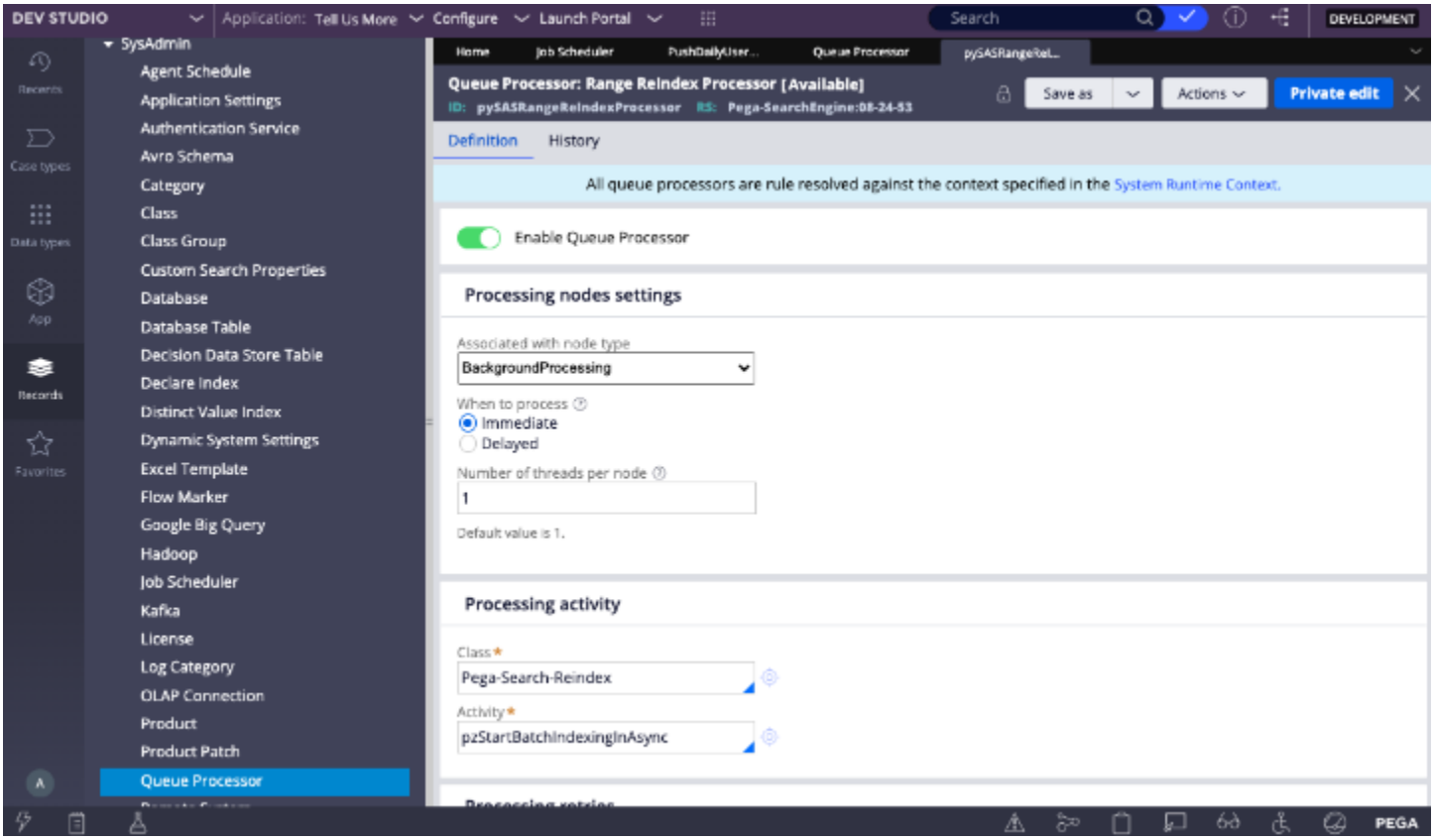
ShippingCountry mülkü "Kanada" eşit olduğunda geçerli olan bir durum versiyonu oluşturarak uluslararası ücreti hesaplayabilirsiniz. Bu yaklaşım, iş mantığını düzenli ve kolay sürdürülür.

Soru: Kurallar tek bir değişken veya birden fazla değişken üzerinde nasıl belirlenir?

TANIM/TANIM(LAR): Kurallar, tek bir değişken üzerinde **tek bir değerli** durum kullanılarak veya **çoklu** değişkenler üzerinde çok değişkenli bir durumla koşullar oluşturulabilir. Tek değerli bir durum, tek bir mülkün değerine dayanan basit bir durumdur; örneğin Ülke = "ABD". Çok değişkenli bir **durum, Ülke = "ABD" VE CustomerType = "Premier" gibi mülk değerlerinin birleşimine** dayalı bir kural uygulamak için bir durum şablonu **ve** bir durum tanımı kullanır.

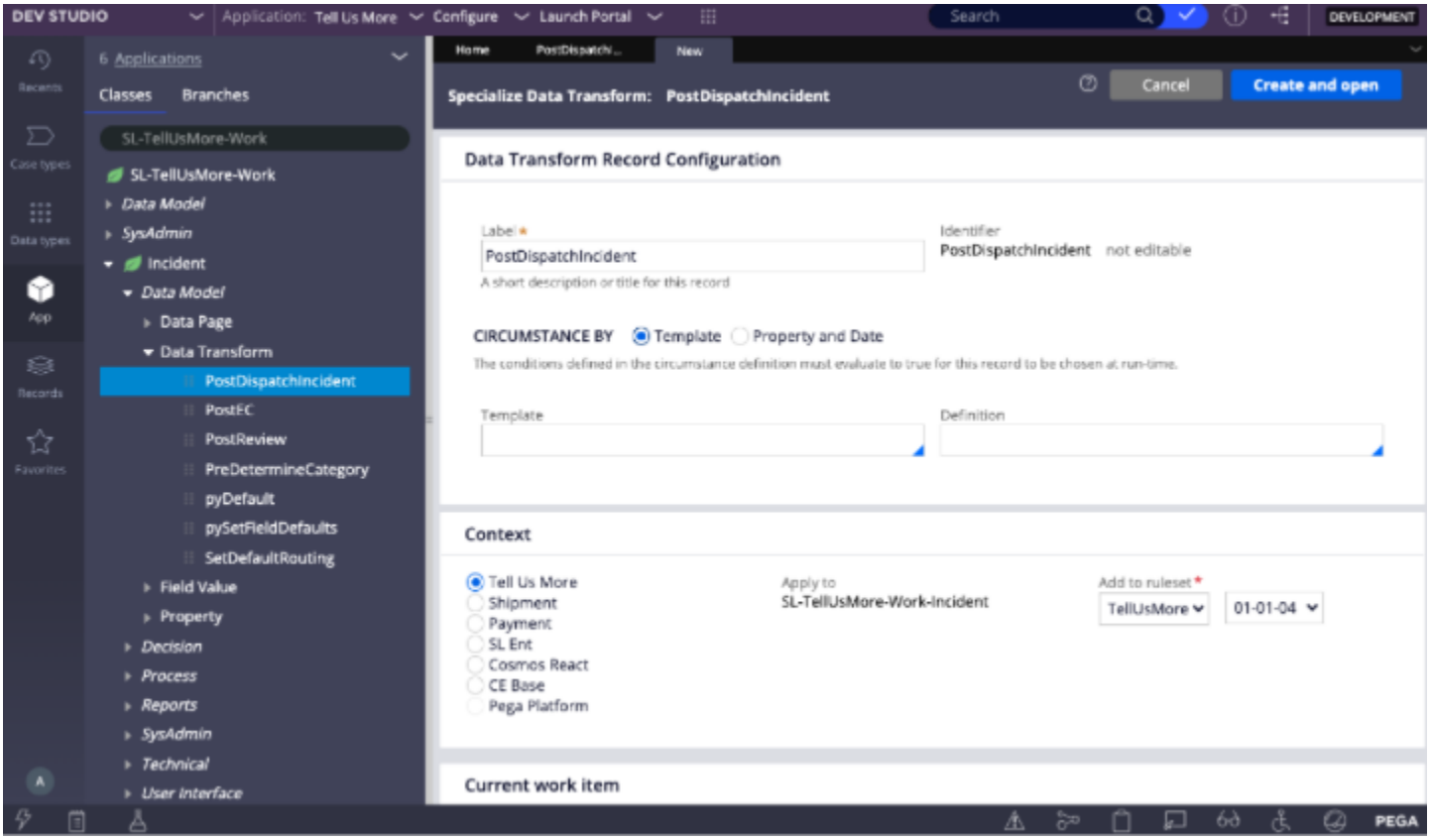
KONUM/GÖRSEL(ler): Dev Studio'dan bir kural durumu değerlendirmek için:

- Dev Studio**'ya gidin**. Uygulamaya tıklayın.
- Durumu belirlemek istediğiniz temel kuralı (örneğin, özellik, veri dönüşümü) açın.
- Araç çubuğunda, **Save As** ve **ardından Özel duruma göre** seçin.
- Tek değerli bir durum oluşturmak için **olduğu gibi** seçeneğini seçin ve tek bir özellik ile onun değerini belirtin.
- Çok değişkenli bir durum oluşturmak için, önceden yapılandırılmış bir **durum şablonu** seçin ve ilgili değerleri verin.



Dunnington Software Development Incorporated

Moving At The Pace Of Integrity!



KULLANIM DURUMU/UYGULAMA(lar): Tek bir değerli durum, tek bir koşula göre değişen basit iş kuralları için idealdir. Örneğin, bir krediyi "Onaylama" için yapılan bir akış eylemi, yalnızca Kredi Miktarı 10.000 $'dan az ise geçerli olan bir durumu olabilir. Daha karmaşık senaryolar için çok değişkenli bir durum kullanılır. Örneğin, bir müşterinin kredi limiti için bir iş kuralı hem CustomerType hem de CreditScore'larına göre değişebilir. Bu iki özellikle bir durum şablonu oluşturur ve ardından her belirli değer kombinasyonu için (CustomerType = "Gold") VE (CreditScore > 750) gibi bir kural oluşturursunuz.