Soru 1)

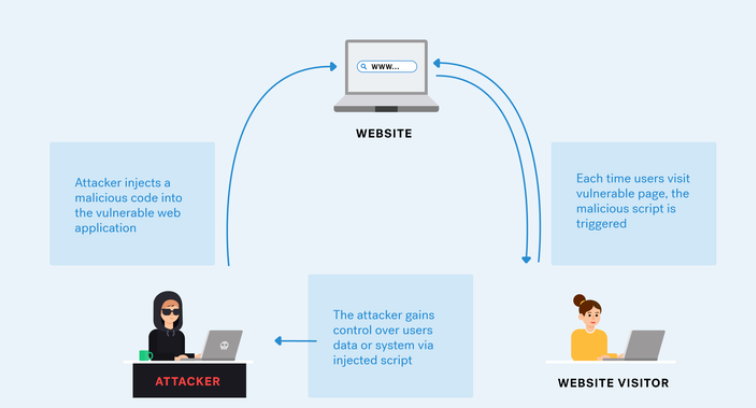
Uygulama güvenliği konusunda Java projelerinizde aldığınız önlemleri açıklayabilir misiniz? Cross-Site Scripting (XSS) veya SQL enjeksiyonu gibi güvenlik açıkları hakkında bilgi verebilir misiniz?

Uygulama güvenliği konusunda Java ile daha önce bir önlem almadım.

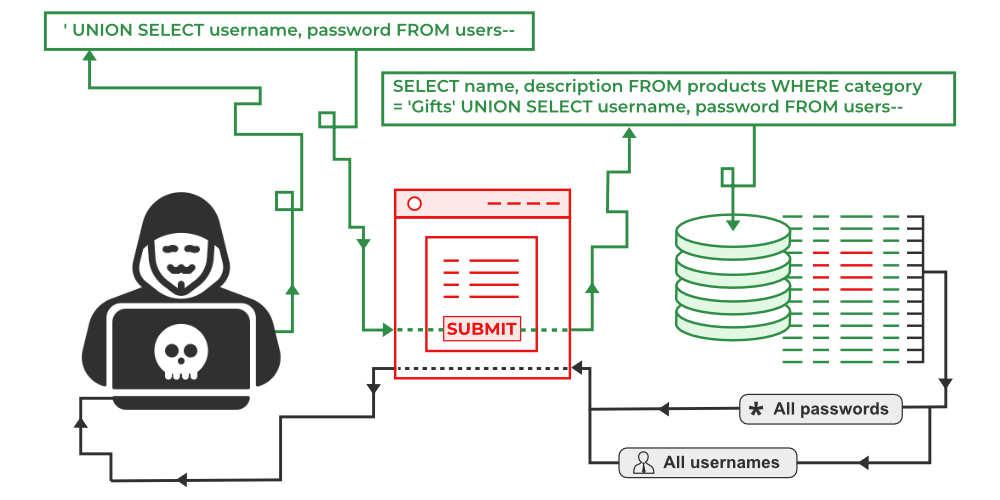
Oracle ve Sap tarafında bu güvenliği sağlayan modül uzmanlarının yönlendirmeleriyle hareket ediyoruz.

Cross-Site Scripting (XSS), bir web uygulamasında kullanıcı girdilerinin doğrudan bir web sayfasına eklenmesi veya gösterilmesi durumunda ortaya çıkan bir güvenlik açığıdır

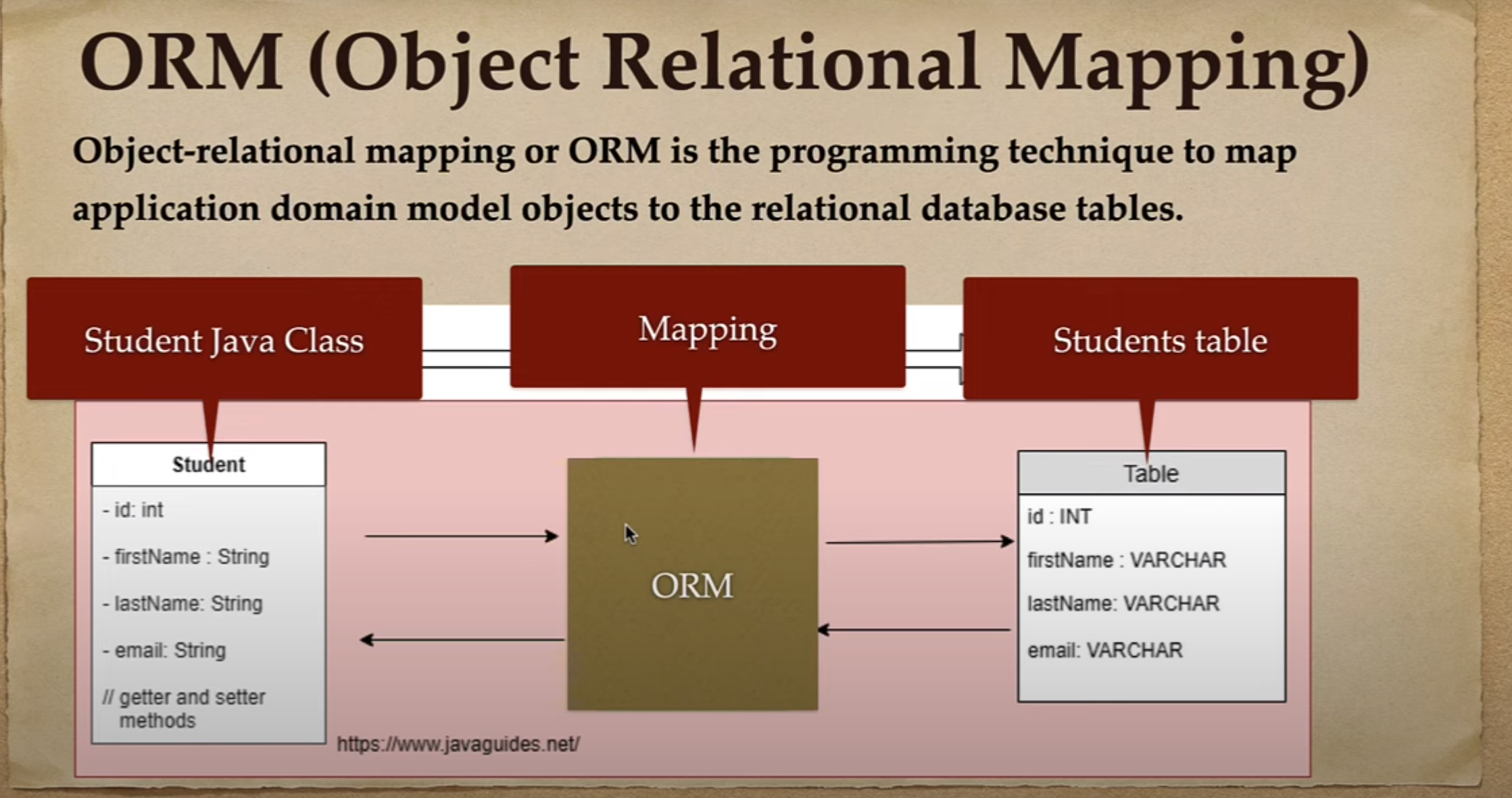
Genellikle bu sorun giriş doğrulaması ile çözülebilir. Form girişlerinde XML ve HTML’İ sınırlayıp sadece kullancının formda ilgili veri türlerini girebilmesini sağlayabiliriz.



SQL Enjeksiyonu, web uygulamalarında kullanıcı girdilerinin doğrudan SQL sorgularına dahil edilmesiyle ortaya çıkan bir güvenlik açığıdır



ORM (Object-Relational Mapping) gibi güvenli veritabanı erişim teknikleri kullanılmalıdır.

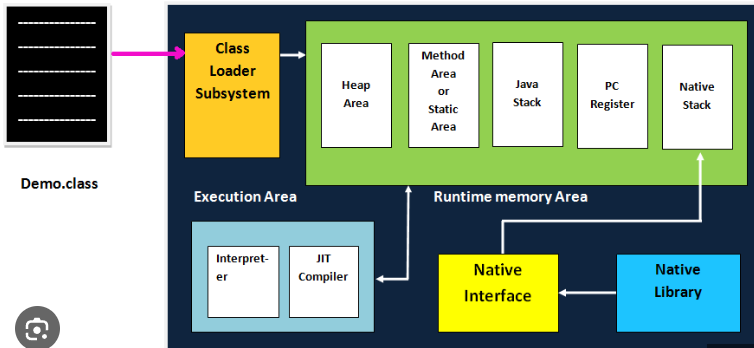


Where koşulunda Parametrelendirilmiş sorgular kullanılabilir.

Klasik çözümlerden biri de Firewall kullanmaktır. Özellikle Databasede kritik tablolar ve verilerin olduğu yerlere Erişim Firewall Üzerinden sağlanabilir.

Soru 2)

Java uygulamalarının performansını nasıl optimize edersiniz? JVM ayarları, veritabanı endekslemesi, kod incelemesi gibi optimizasyon stratejilerini açıklayın



JVM Ayarları

Heap Boyutunu Ayarlama: -Xms ve -Xmx parametreleri ile heap boyutunu başlangıç ve maksimum değerler olarak ayarlayarak JVM heap alanını optimize edebilir

Garbage Collection Ayarları: -XX:+UseG1GC veya -XX:+UseConcMarkSweepGC gibi GC algoritmalarını seçerek ve -XX:G1HeapRegionSize veya -XX:CMSInitiatingOccupancyFraction gibi parametreleri ayarlayarak garbage collection performansını iyileştirilebilir

Just-In-Time (JIT) Derlemesi: -XX:+TieredCompilation gibi JIT derleme ayarlarını kullanarak Java kodunun daha hızlı çalışmasını sağlayabilir

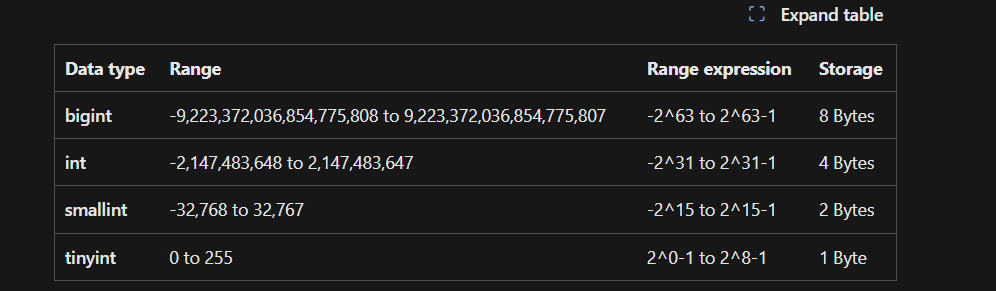
Veri Depolama Optimizasyonu: -XX:ObjectAlignmentInBytes gibi ayarlarla nesne hizalamasını optimize ederek bellek kullanımını azaltabilir

Veritabanı Optimizasyonu

Endeksleme: Veritabanı sorgularını hızlandırmak için uygun alanları SQL İndex kullanarak veritabanı performansını artırabilir

Veri saklarken gelecek veriler analiz edilip very tipleri ona göre kullanılmalıdır.

Okul Numarası 5 haneli ise int yerine smallint kullanılmalı



Kod İncelemesi ve Optimizasyon

Profiling Araçları Kullanma: Profiling araçları (örneğin, VisualVM, JProfiler) kullanarak uygulamanızın performansını analiz edebilir ve yavaş çalışan bölümleri belirleyebilirsiniz.

Etkinlik ve Kaynak Kullanımı: Kaynak yoğun bölümleri (örneğin, döngüler, ağ çağrıları) inceleyerek ve gereksiz kaynak kullanımını azaltarak performansı artırabilirsiniz.

3) Java projelerinizde hangi tasarım şablonlarını sıkça kullanıyorsunuz? Örnek vererek, belirli bir senaryoda hangi tasarım şablonunu seçtiğinizi açıklayın.

Java projem daha önce olmadı. Web tarafında hiç takip ettiğim bir tasarım şablonum olmadı. Her zaman o anki soruna ve ya konuya o anın şartları doğrultusunda bir reaksiyon vererek ilerledim.

Nasıl ilerlemem gerektiği hakkında bir fikrim yoksa genellikle internetten birine sorarak son dönemlerde de Chat GPT’ye sorarak hareket ediyorum.

Soru 4 )

RESTful veya SOAP tabanlı web servislerini geliştirme deneyiminizi anlatın. Veri formatları

(JSON vs. XML) ve güvenlik önlemleri hakkında bilgi verin.

Daha önce almış olduğum Sap Fiori eğitiminde RESTFUL mimarisine dayanan SAP OData servisler üzerine bir eğitim bloğu hazırlamıştım. Medium kanalımda Türkçe hallerine Kişisel blogumda ise ingilizce hallerine ulaşabilirsiniz.

Türkçe

<https://kaancancalkan.medium.com/sap-odata-servis-part-2-bc641e185737>

<https://kaancancalkan.medium.com/sap-odata-servis-giriş-cb424ef05c4e>

[https://kaancancalkan.medium.com/sap-hcm-abap-fiori-odata-servis-yazma-7d19da99074d](%20https://kaancancalkan.medium.com/sap-hcm-abap-fiori-odata-servis-yazma-7d19da99074d)

İngilizce

<https://www.kaancancalkan.dev/sap-odata-service-part-2/>

<https://www.kaancancalkan.dev/sap-odata-service-part-1/>

<https://www.kaancancalkan.dev/writing-sap-hcm-abap-fiori-odata-service/>

JSON ve XMLleri Sap tarafında Servis yazdığım için herhangi bir auttenthication token ile korumak zorunda kalmadım.

TEORIK BİLGİLER

JSON (JavaScript Object Notation)

Json’un Özellikleri ;

Basitlik ve Kolaylık: JSON, insanlar tarafından okunabilir ve yazılabilir bir formata sahiptir, bu nedenle veriyi oluşturmak ve anlamak kolaydır.

Web Uygulamaları için Popülerlik: JSON, JavaScript ile uyumludur ve modern web uygulamalarında sıkça kullanılır.

Veri Taşıma: Veri taşıması için hafif bir yapıya sahiptir ve genellikle RESTful web servislerinde kullanılır.

Güvenlik: JSON, verileri aktarırken güvenlik açısından bazı endişelere yol açabilir. Örneğin, JSON Injection saldırıları, kötü niyetli kullanıcıların JSON verilerine gömülü kötü amaçlı kodları kullanarak saldırı gerçekleştirmelerine neden olabilir. Bu tür saldırılara karşı korunmak için giriş doğrulaması, veri doğrulaması ve güvenli JSON ayrıştırma teknikleri gibi önlemler alınabilir.

XML (eXtensible Markup Language)

XML Özellikleri;

Esneklik: XML, çok amaçlı bir etiketleme dilidir ve verileri yapısallaştırmak için esnek bir şekilde kullanılabilir.

Kurumsal Uygulamalar: XML, büyük ölçekli kurumsal uygulamalarda sıkça kullanılır, özellikle veri değişimi ve depolama için.

Veri Doğrulama: XML Schema gibi araçlarla veri doğrulama ve belgelendirme kolaydır.

Güvenlik: XML ayrıştırıcılar, zararlı içerikleri engellemek için güvenlik duvarları ve güvenli işleme teknikleriyle birlikte kullanılabilir. Ancak XML, karmaşık yapısı nedeniyle kötü niyetli kullanıcıların XML Enjeksiyon saldırıları gerçekleştirmelerine olanak tanır. Bu tür saldırılara karşı korunmak için giriş doğrulaması, veri doğrulaması ve güvenli XML işleme teknikleri kullanılabilir.

Güvenlik Önlemleri:

Giriş Doğrulaması: Kullanıcı girişleri ve dış kaynaklardan gelen veriler doğrulanmalı ve güvenilir olup olmadığı kontrol edilmelidir.

Veri Doğrulaması: Gelen verilerin yapısı ve içeriği, belirlenen güvenlik kurallarına uygun olup olmadığı kontrol edilmelidir.

Güvenli Ayrıştırma: JSON veya XML verilerini ayrıştırırken, güvenli ayrıştırma teknikleri kullanılmalıdır. Bunlar, potansiyel olarak zararlı içerikleri engellemek için tasarlanmış kütüphaneleri ve API'leri içerir.

XML ve JSON Injection Saldırılarına Karşı Koruma: Bu tür saldırılara karşı önlem almak için girişleri temizlemek, kaçış karakterlerini işlemek ve güvenli veri ayrıştırıcılar kullanmak önemlidir.

Her iki formatta da, güvenlik, uygulamanın kullanım senaryolarına ve gereksinimlerine bağlı olarak farklılık gösterebilir. Ancak genel olarak, giriş doğrulaması, veri doğrulaması ve güvenli veri işleme teknikleri gibi standart güvenlik önlemleri, JSON ve XML tabanlı uygulamalarda önemlidir.

Soru 5 )

Aşağıdaki değerlere göre SELECT count(\*) AS cust\_not\_123\_total FROM orders WHERE customer\_id <> '123' sorgu sonucu ne olacaktır?

SELECT count(\*) AS total FROM orders;

+-------+

| total |

+-------+

| 100 |

+-------+

SELECT count(\*) AS cust\_123\_total FROM orders WHERE customer\_id = '123';

+----------------+

| cust\_123\_total |

+----------------+

| 15 |

+----------------+

123 idli kişi sayısı 15

Toplam kişi 100

123 idli olmayan kişi sayısı 100-15 =85