MODULE 2 DATABASE PROGRAMMING

Integration Testing

TECH
ELEVATOR
A Stride Company

YESTERDAY…

How do we connect to a database?

What is DAO?

What is CRUD?

What is Unit Testing?

Integration Testing

- **Integration Testing** is a broad category of tests that validate the integration between units of code or code and outside dependencies such as databases or network resources.

Integration tests:

- Use the same tools as unit tests (i.e. JUnit or MSTest)
- Usually slower than unit tests (but often still measured in ms)
- More complex to write and debug
- **Can have dependencies on outside resources like files or a database**

Tests should be...

Repeatable: If the test passes/fails on first execution, it should pass/fail on second execution if no code has changed.

Independent: A test should be able to be run on it's own, independently of other tests, OR together with other tests and have the same result either way.

Obvious: When a test fails, it should be as obvious as possible why it failed.

How do we handle the data?

ELEVATE YOURSELF

Remotely Hosted Shared Test Database

- *An RDBMS is installed on a remote server and shared by all developers on the team for testing.*
  - *Advantages:*
    - Easy setup, often already exists
    - Production-like software and (possibly) hardware
  - *Disadvantages:*
    - Unreliable and brittle
    - Lack of test isolation
    - Temptation to rely on existing data (which can change)

Locally Hosted Test Database

- *An RDBMS is installed and hosted locally on each developer's machine. (This is the approach we will use)*
  - *Advantages:*
    - Production-like software
    - Reliable (local control)
    - Isolation
  - *Disadvantages:*
    - Requires local hardware resources
    - RDBMS needs to be installed and managed

Embedded, In-memory Database

- *An in-memory, embedded database server is started and managed by test code while running integration tests.*
  - *Advantages:*
    - Very reliable
    - Consistent across development machines
    - (managed by source control)
    - Lightweight
  - *Disadvantages:*
    - Not the same software used in production
    - Cannot use proprietary features of production RDBMS

Still, what about *repeatable?*

A **transaction** is a single unit of work. When it is successful, it should be "committed". If an error is encountered at any point it should be cancelled or rolled back.

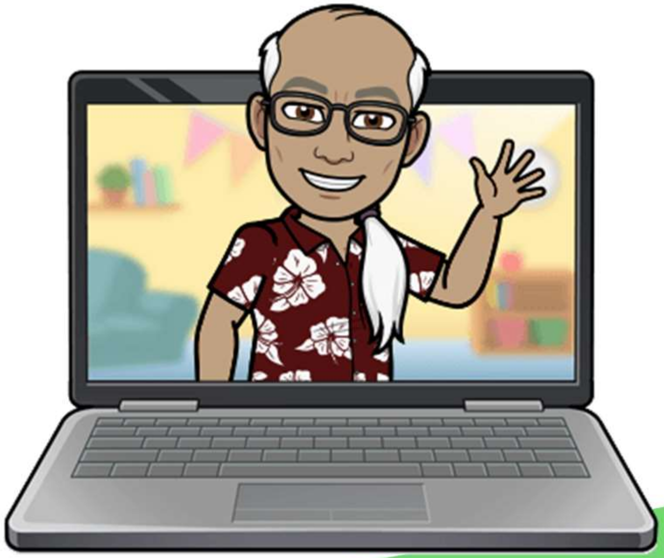
Test Decorations

@Before

@After

@Test

LET'S CODE!



ELEVATE YOURSELF

Safe Guarding Data

- Q: When you forget your password, how come the web site doesn’t just send you the password?
- A: A secure web site can’t see your password…ever.

1. We need to be able to *verify* a password but not *recover* it.
2. A system administrator with access to credential data should not be able to determine a password.
3. Any hacker that steals a database or set of credentials should not be able to read the passwords.
4. **Even with supercomputing capabilities, no one should be able to access the data within any reasonable amount of time**

Password Hashing

Use a one-way function to obfuscate the plain-text password prior to storage.

Use the password supplied by the user, re-hash it, and compare it to the stored password hash value.

Salt the passwords in order to make it take longer to calculate all possibilities.

Hashing Requirements

- Input to outputs are constrained
  - Infinite inputs => limited output means duplicates
- Relationship between input and output should appear random
  - If TomA => asdfg then TomB should not be asdfh
- Inputs should be evenly distributed over the set of outputs

More Weapons in the good fight

- Encryption
  - The most effective way to achieve data security

- Securing Data at Rest
  - Data at rest can use a form of encryption called **symmetric key encryption**
  - Requires both parties to use the **same** key to encrypt and decrypt data.
  - Any party possessing the key can read the data.
  - Has difficulties of securing the symmetric key amongst multiple parties.
- Securing Data in Transit
  - It may be necessary to allow others to send you secure data without worrying that it be intercepted.
  - Giving the secure key away would not be a good decision.
  - Asymmetric algorithms allow us to create a **public key** and a **private key**.
  - The public key is distributed freely.
  - the private key is kept to ourselves.

Securing the Web

- SSL and TLS
  - Secure Socket Layer and Transport Layer Security are examples of asymmetric key encryption.
  - SSL was developed by Netscape in 1994 to secure transactions over the WWW.
  - TLS and SSL are recognized as protocols to provide secure HTTP(S) for internet transactions. It supports authentication, encryption, and data integrity.
- Digital Certificates
  - Ownership of a public key is certified by use of a digital certificate allowing parties to rely upon the signature generated by the private key.
  - A certificate authority is a trusted third-party that provides the certificate.
  - The CA prevents the attacker from impersonating a server by indicating that the certificate belongs to a particular domain.

WHAT QUESTIONS DO YOU HAVE?

Reading for Monday:

Networking and HTTP

Consuming RESTful APIs (Part 1)