Phase 6: Apex Programming

Project: EdTech Enrollment Automation | Author: Parth Khare

Objective

Extend automation beyond Flow Builder by implementing Apex triggers for backend logic, audit logging, and data integrity across the Enrollment__c object.

Step 1: Identify Trigger Use Cases

Based on Flow limitations and project needs:

- Enforce business rules (e.g., no future enrollment dates)
- Auto-assign ownership for accountability
- Log status transitions for audit and reporting
- Prepare for future cross-object logic (e.g., Course capacity updates)

Step 2: Create Apex Triggers

Trigger 1: Prevent Future Enrollment Dates

Object: Enrollment__c | Events: before insert, before update

```
trigger PreventFutureEnrollmentDate on Enrollment__c (before insert, before update) {
for (Enrollment__c enroll : Trigger.new) {
 if (enroll.Enrollment_Date__c > Date.today()) {
 enroll.addError('Enrollment date cannot be in the future.');
 }
 }
}
```

Purpose: Reinforces validation rule at backend level, ensuring API and Flow-based inserts are also blocked.

Trigger 2: Auto-Assign Enrollment Owner

Object: Enrollment__c | Event: before insert

```
trigger AutoAssignEnrollmentOwner on Enrollment__c (before insert) {
 for (Enrollment__c enroll : Trigger.new) {
 if (enroll.OwnerId == null) {
 enroll.OwnerId = UserInfo.getUserId();
 }
 }
}
```

Purpose: Ensures every Enrollment record has an owner, supporting audit and accountability.

Trigger 3: Log Status Changes

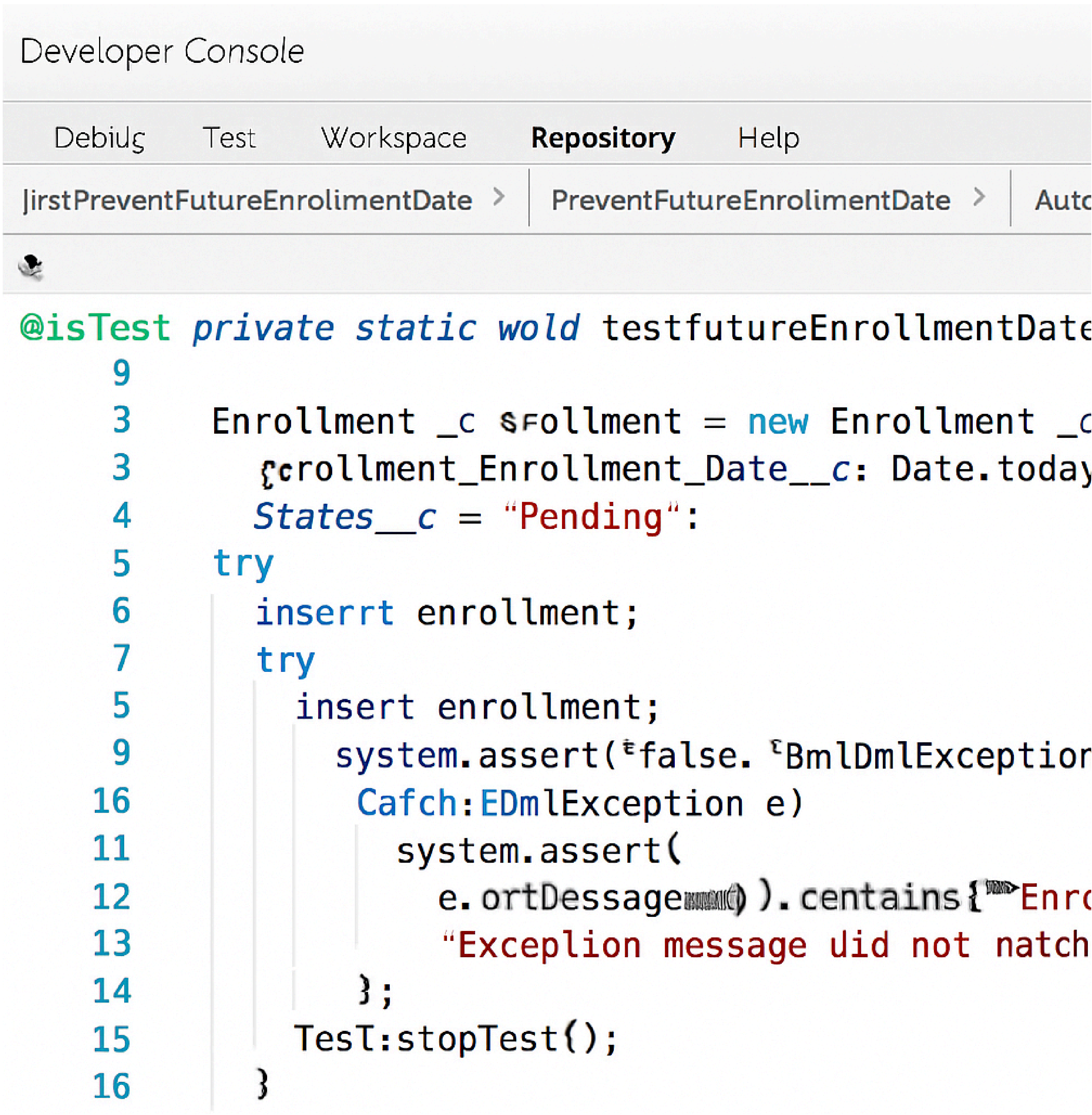
Object: Enrollment__c | Event: after update

```
trigger LogEnrollmentStatusChange on Enrollment__c (after update) {
 for (Enrollment__c enroll : Trigger.new) {
 Enrollment__c oldEnroll = Trigger.oldMap.get(enroll.Id);
 if (enroll.Status__c != oldEnroll.Status__c) {
 Enrollment_Log__c log = new Enrollment_Log__c();
 log.Enrollment__c = enroll.Id;
 log.Old_Status__c = oldEnroll.Status__c;
 log.New_Status__c = enroll.Status__c;
 log.Changed_By__c = UserInfo.getUserId();
 insert log;
 }
 }
}
```

Purpose: Tracks status transitions for audit and reporting, complements Flow-based automation.

Step 3: Build Test Classes

Create @isTest classes for each trigger
Use realistic test data (e.g., valid Student__c,
Course__c) Ensure ≥75% code coverage for deployment



```
@isTest private static wold testfutureEnrollmentDate
    Enrollment _c sFollment = new Enrollment _c
      {crollment_Enrollment_Date__c: Date.today
      States__c = "Pending":
    try
      inserrt enrollment;
      try
        insert enrollment;
          system.assert(‘false. ‘BmlDmlException
           Cafch:EDmlException e)
             system.assert(
               e.ortDessage[illegible]).centains{"Enro
               "Exceplion message uid not natch
           };
        TesT:stopTest();
      }
```

Step 4: Document Trigger Logic

Add inline comments in Apex code
Create markdown documentation:
- Trigger name and purpose
- Events and object references
- Related fields and logic
- Test class summary

Step 5: Deploy via Change Set

Include:

- Apex Triggers
- Test Classes
- Custom Objects (e.g., Enrollment_Log__c)
- Metadata dependencies (fields, layouts)

Step 6: Post-Deployment Validation

- Create test Enrollment records
- Confirm:
  - Owner assignment
  - Status change logs
  - Error handling
- Monitor logs and Flow behavior for 48 hours