Phase 5 – Apex Programming

rce-app > main > default > triggers > TrainingTrigger.trigger > TrainingTrigger

```
trigger TrainingTrigger on Training__c (after insert, after update) {
    if(Trigger.isAfter & Trigger.isInsert){
        TrainingTriggerHandler.afterInsert(Trigger.new);
    }
    if(Trigger.isAfter && Trigger.isUpdate){
        TrainingTriggerHandler.afterUpdate(Trigger.new, Trigger.oldMap);
    }
}
```

```
public with sharing class TrainingTriggerHandler {

    public static void afterInsert(List<Training__c> listNew){
        Map<Id,String> mapContactIds = new Map<Id,String>();
        for(Training__c item : listNew){
            mapContactIds.put(item.Contact__c,item.Status__c);
        }
        List<Contact> listContact = [SELECT Id,Training_Status__c FROM Cor
        for(Contact item : listContact){
            if(mapContactIds.containsKey(item.Id)){
                item.Training_Status__c = mapContactIds.get(item.Id);
            }
        }
        if(!listContact.isEmpty()){
            update listContact;
        }
    }

    public static void afterUpdate(List<Training__c> listNew, Map<Id,Trair
        Map<Id,String> mapContactIds = new Map<Id,String>();
        for(Training__c item : listNew){ // Status = 'Not Started' -> 'In
            Training__c oldRecord = mapOld.get(item.Id);
            if(item.Status__c != oldRecord.Status__c){
                mapContactIds.put(item.Contact__c,item.Status__c);
            }
        }
        List<Contact> listContact = [SELECT Id,Training_Status__c FROM Cor
        for(Contact item : listContact){
            if(mapContactIds.containsKey(item.Id)){
                item.Training_Status__c = mapContactIds.get(item.Id);
            }
        }
        if(!listContact.isEmpty()){
```

```
import { LightningElement, wire, api } from 'lwc';
import { refreshApex } from '@salesforce/apex';
import getTrainings from '@salesforce/apex/TrainingService.getTrainings';
import updateStatus from '@salesforce/apex/TrainingService.markCompleted';

export default class TrainingList extends LightningElement {

    @api recordId;

    @wire(getTrainings, { contactId: '$recordId' }) trainings;

    handleMarkCompleted(eve (parameter) event: any
        const trainingId = event.target.dataset.id;
        updateStatus({ trainingId })
            .then(() => {
                return refreshApex(this.trainings);
            });
    }

}
```

```
public with sharing class TrainingService {

    @AuraEnabled(cacheable=true)
    public static List<Training__c> getTrainings(Id contactId){
        return [SELECT Id, Course_Name__c, Status__c FROM Training__c WHERE Contact__c = :contactId];
    }

    @AuraEnabled
    public static void markCompleted(Id trainingId){
        Training__c t = [SELECT Id, Status__c FROM Training__c WHERE Id = :trainingId LIMIT 1];
        t.Status__c = 'Completed';
        update t;
    }
}
```