Sprint-2 Plan

Employee Detail Management

1. Establishing Project Folder Architecture
2. Server App Initialization
3. **Sprint-2 Execution** [2(a), 2(b), 2(c), **2(d)**(i)(ii)**(iii)**(iv)(v), 2(e), 2(f)]
4. Completing Sprint-1 Objectives

2(a) Employee Module - Analysis, Design & DB-Preparation
2(b) Employee View
2(c) Employee Search

2(d) Employee Insert

2(e) Employee Update
2(f) Employee Delete

==========

2(d) Employee Insert

1. Folder and Project preparation with Backups

2. Prepare Supportive Data for the Insert Form
   - (A) Establish and Test Server Services need for Supportive Data List
     - (1) genders/list
     - (2) designations/list
     - (3) employeestatus/list (entity/dao/controller)
   - (B) Establish and Test for Client Services
     - (1) GenderService, DesignationService, EmployeestatusService (entity/services)
     - (2) Test with using initialize()

3. **Prepare Regex(Regular Expression) Service for the Insert Form**
   1. **Define Validation Criteria for Attribute of the Selected Entity**
   2. **Define Regex for required attributes**
   3. **Add Annotation into the Entity Class from the Java Validation Framework**
   4. **Implement Regex Pattern Annotation and Regex Processor**
   5. **Implement Regex Controller and Test**
   6. **Implement and Test Regex Service using the Client App**

4. Client App
5. Server-App

==========

Validation Criteria with Regex Patterns for Text Base Attributes

id	Auto Incremented	
number	"^\\d{4}$"	
fullname	"^([A-Z][a-z]*[.]?[\\s]?)*([A-Z][a-z]*)$"	
callingname	"^([A-Z][a-z]+)$"	
photo	BLOB	
gender	FK	
dobirth	Date	
nic	"^(([\\d]{9}[vVxX])|([\\d]{12}))$"	
address	"^([\\w\\/\\-,\\s]{2,})$"	
mobile	"^0\\d{9}$"	
land	"^0\\d{9}$"	
doassignment	Date	
designation	FK	
empstatus	FK	
description	"^.*$"	

Prepare Regex (Regular Expression) Service for the Insert Form

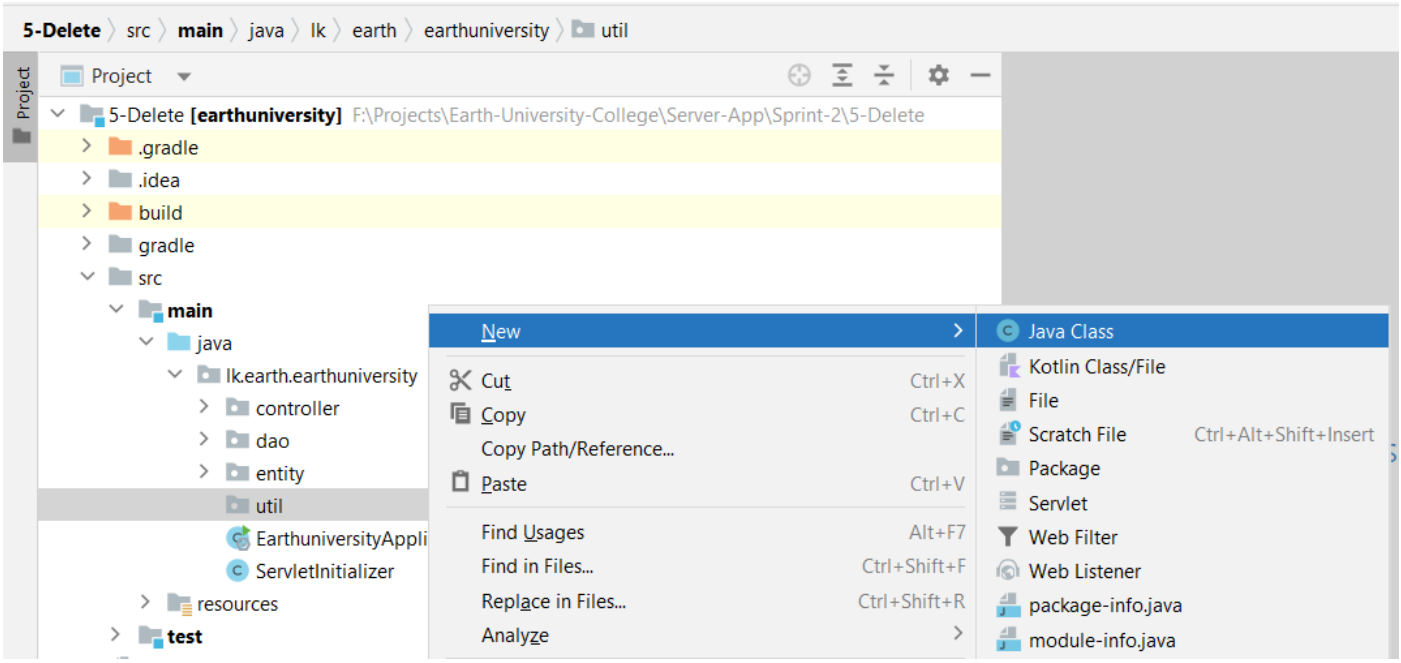
1. **Define Validation Criteria for Attribute of the Selected Entity**
2. **Define Regex for required attributes**
3. **Add Annotation into the Entity Class from the Java Validation Framework**
4. **Implement Regex Pattern Annotation and Regex Processor**
5. **Implement Regex Controller and Test**
6. **Implement and Test Regex Service using the Client App**

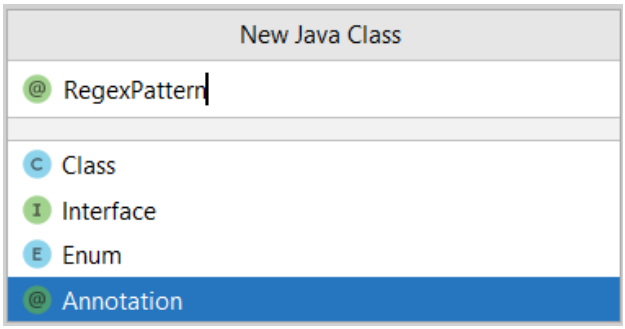
Add Annotation to the Entity Class from the Java Validation Framework

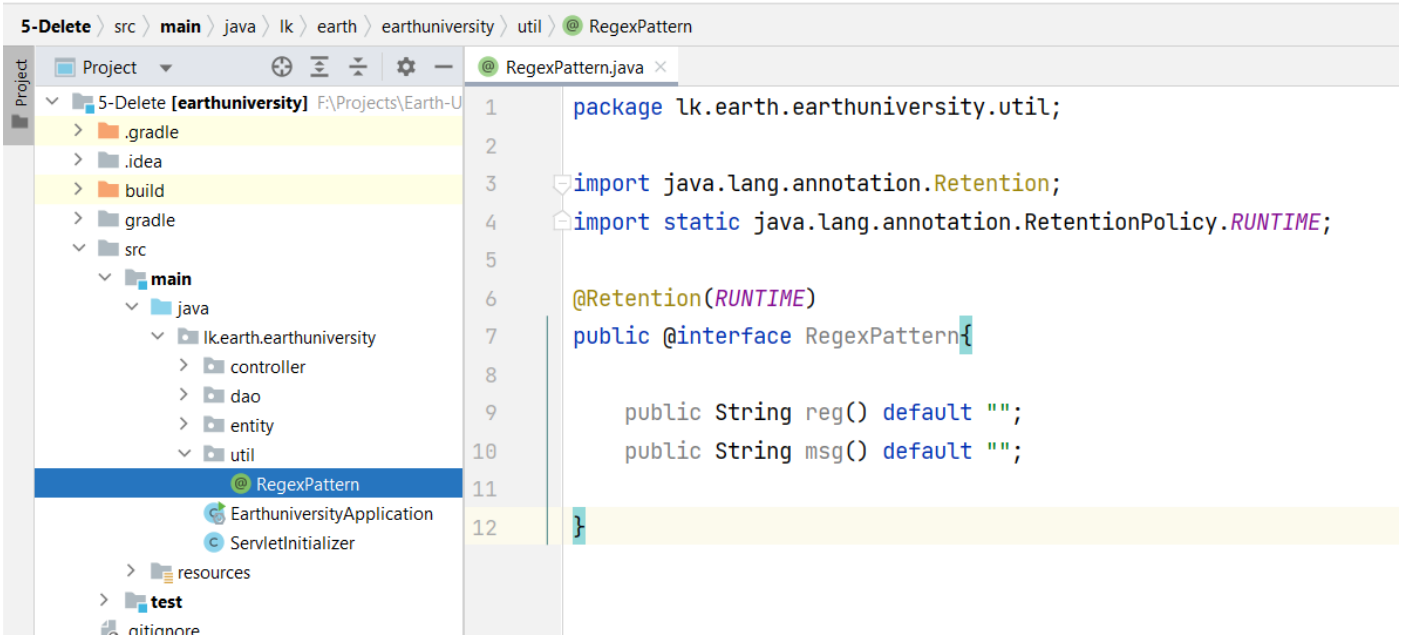
Open Server App → Open Employee Entity → Add Annotation as Follows

Employee.java

```java
import javax.validation.constraints.Pattern;

@Entity
public class Employee {
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Id
    @Column(name = "id")
    private Integer id;
    @Basic
    @Column(name = "number")
    @Pattern(regexp = "^\\d{4}$", message = "Invalid Number")
    private String number;
    @Basic
    @Column(name = "fullname")
    @Pattern(regexp = "^([A-Z][a-z]*[.]?[\\s]?)*([A-Z][a-z]*)$", message = "Invalid Fullname")
    private String fullname;
    @Basic
    @Column(name = "callingname")
    @Pattern(regexp = "^([A-Z][a-z]+)$", message = "Invalid Calligname")
    private String callingname;
    @Basic
    @Column(name = "photo")
    private byte[] photo;
    @Basic
    @Column(name = "dobirth")
    private Date dobirth;
    @Basic
    @Column(name = "nic")
    @Pattern(regexp = "^(([\\d]{9}[vVxX])|([\\d]{12}))$", message = "Invalid NIC")
    private String nic;
    @Basic
    @Column(name = "address")
    @Pattern(regexp = "^([\\w\\/\\-,\\s]{2,})$", message = "Invalid Address")
    private String address;
    @Basic
    @Column(name = "mobile")
    @Pattern(regexp = "^0\\d{9}$", message = "Invalid Mobilephone Number")
    private String mobile;
    @Basic
    @Column(name = "land")
    @Pattern(regexp = "^0\\d{9}$", message = "Invalid Landphone Number")
    private String land;
    @Basic
    @Column(name = "doassignment")
    private Date doassignment;
    @Basic
    @Column(name = "description")
    @Pattern(regexp = "^.*$", message = "Invalid Description")
    private String description;
```

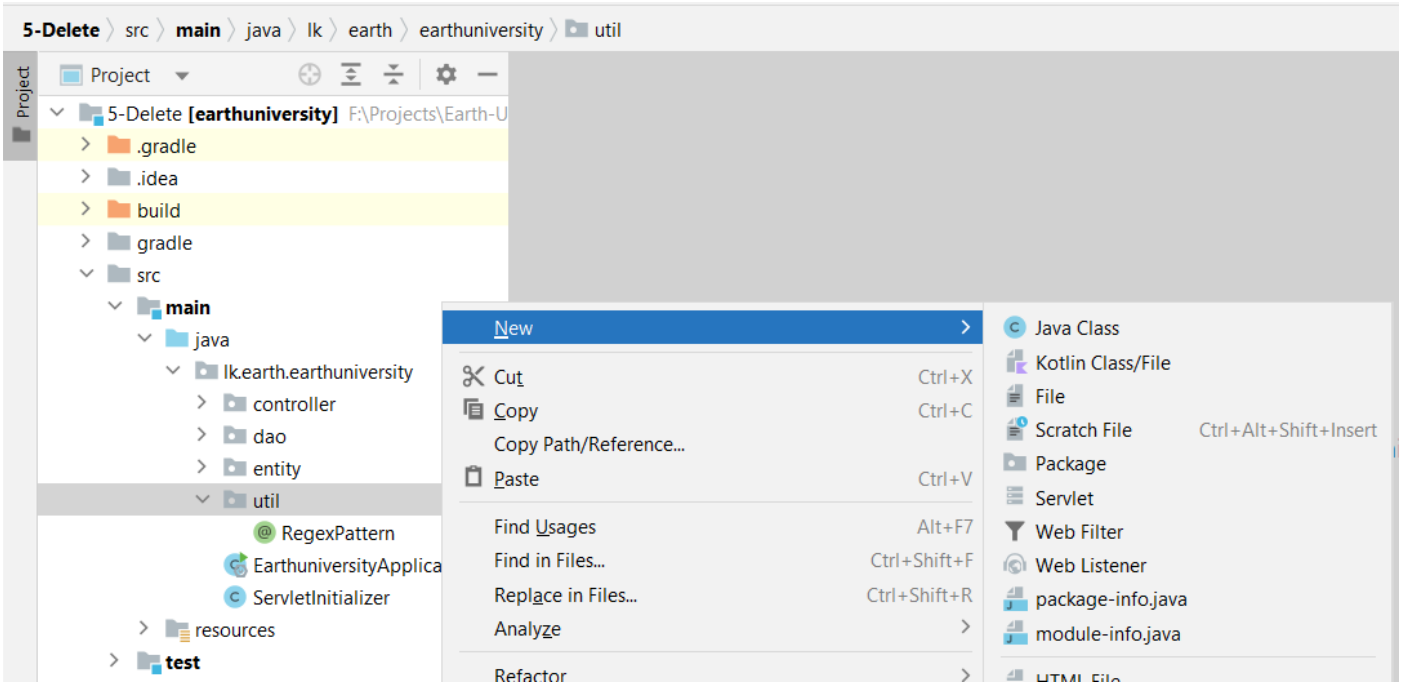


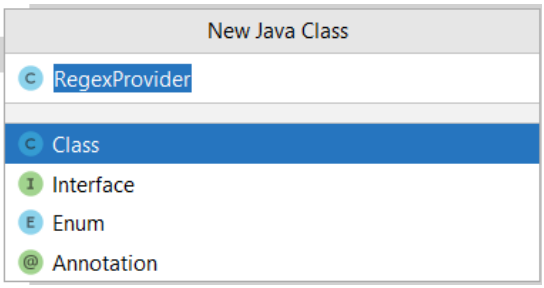


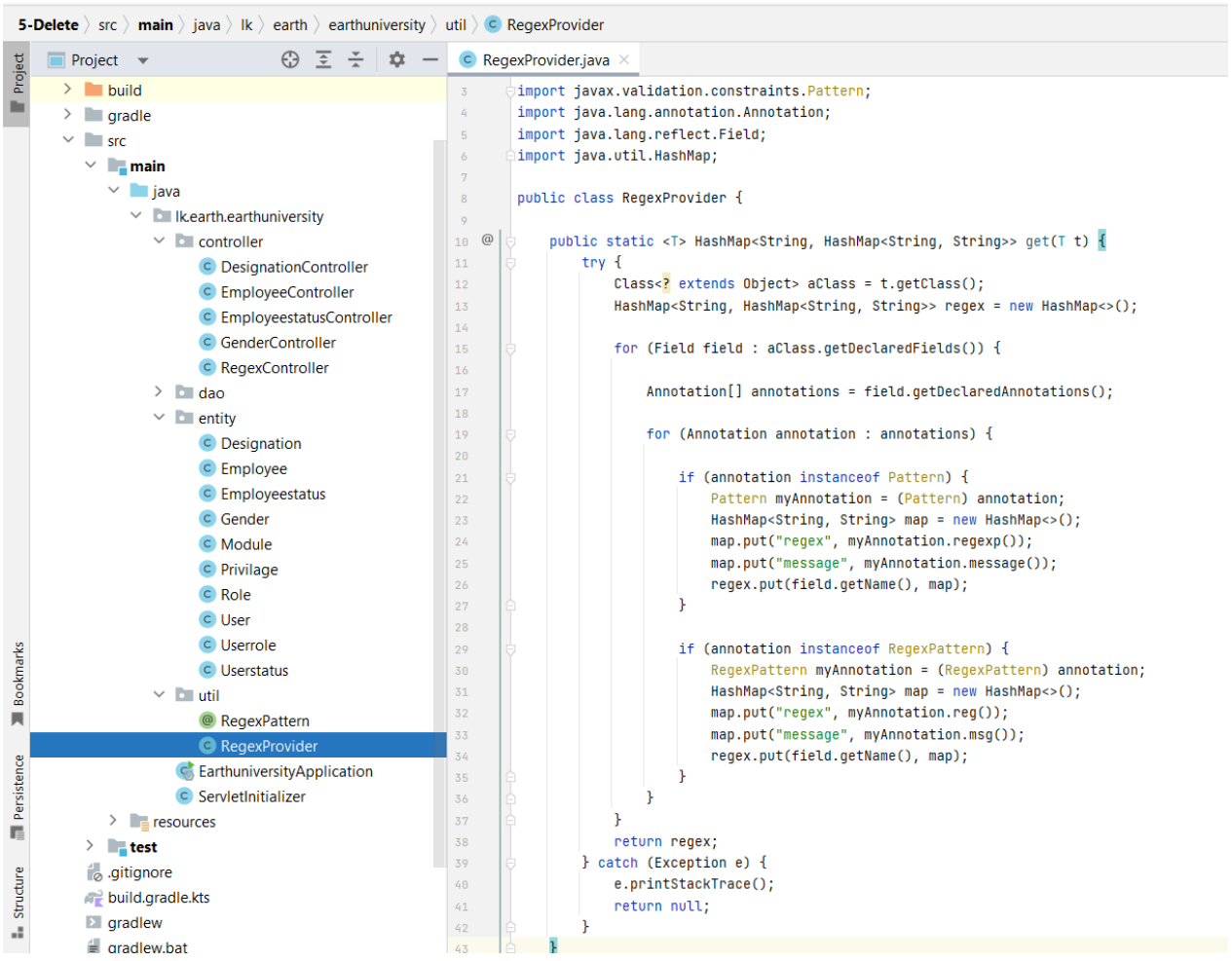


This annotation will be useful in situation like price format validation “234.75” which is not applicable with Java Validation Framework due to Data Type conversion from String to Double.
(Date format validation can also be applied with this Annotation)

Other than all of the above you need to learn how to define, use and process annotation.


5-Delete src main java lk earth earthuniversity util
Project
5-Delete [earthuniversity] F:\Projects\Earth-U
.gradle
.idea
build
gradle
src
main
java
lk.earth.earthuniversity
controller
dao
entity
util
RegexPattern
EarthuniversityApplica
ServletInitializer
resources
test
New
Cut Ctrl+X
Copy Ctrl+C
Copy Path/Reference...
Paste Ctrl+V
Find Usages Alt+F7
Find in Files... Ctrl+Shift+F
Replace in Files... Ctrl+Shift+R
Analyze
Refactor
Java Class
Kotlin Class/File
File
Scratch File Ctrl+Alt+Shift+Insert
Package
Servlet
Web Filter
Web Listener
package-info.java
module-info.java


New Java Class
RegexProvider
Class
Interface
Enum
Annotation

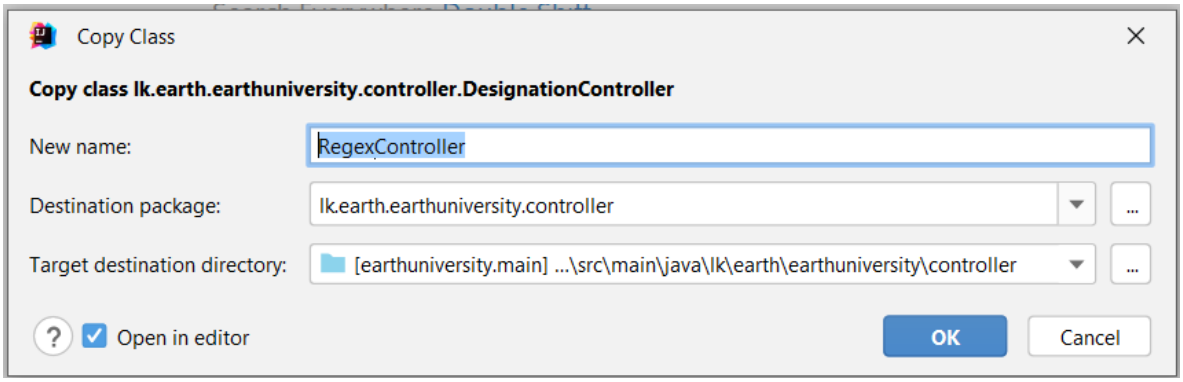

5-Delete src main java lk earth earthuniversity util RegexProvider
Project
RegexProvider.java
build
gradle
src
main
java
lk.earth.earthuniversity
controller
DesignationController
EmployeeController
EmployeestatusController
GenderController
RegexController
dao
entity
Designation
Employee
Employeestatus
Gender
Module
Privilage
Role
User
Userrole
Userstatus
util
RegexPattern
RegexProvider
EarthuniversityApplication
ServletInitializer
resources
test
.gitignore
build.gradle.kts
gradlew
gradlew.bat
import javax.validation.constraints.Pattern;
import java.lang.annotation.Annotation;
import java.lang.reflect.Field;
import java.util.HashMap;
public class RegexProvider {
public static <T> HashMap<String, HashMap<String, String>> get(T t) {
try {
Class<? extends Object> aClass = t.getClass();
HashMap<String, HashMap<String, String>> regex = new HashMap<>();
for (Field field : aClass.getDeclaredFields()) {
Annotation[] annotations = field.getDeclaredAnnotations();
for (Annotation annotation : annotations) {
if (annotation instanceof Pattern) {
Pattern myAnnotation = (Pattern) annotation;
HashMap<String, String> map = new HashMap<>();
map.put("regex", myAnnotation.regexp());
map.put("message", myAnnotation.message());
regex.put(field.getName(), map);
}
if (annotation instanceof RegexPattern) {
RegexPattern myAnnotation = (RegexPattern) annotation;
HashMap<String, String> map = new HashMap<>();
map.put("regex", myAnnotation.reg());
map.put("message", myAnnotation.msg());
regex.put(field.getName(), map);
}
}
}
return regex;
} catch (Exception e) {
e.printStackTrace();
return null;
}
}

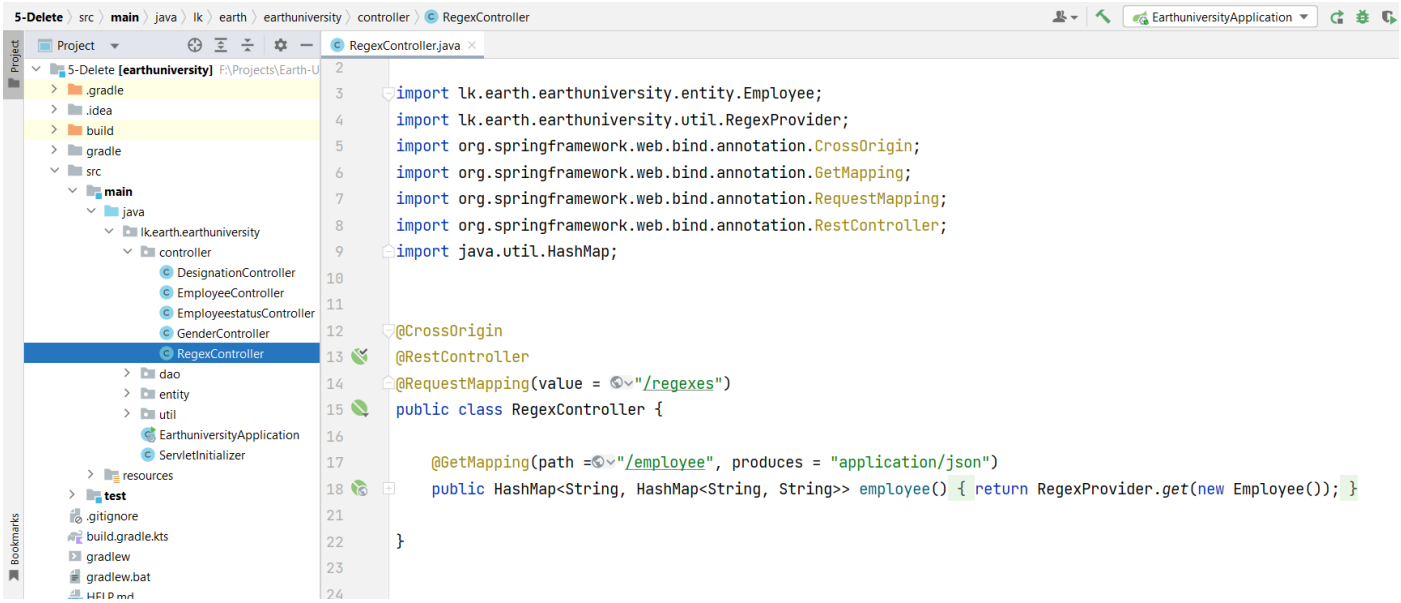
```java
package lk.earth.earthuniversity.util;

import javax.validation.constraints.Pattern;
import java.lang.annotation.Annotation;
import java.lang.reflect.Field;
import java.util.HashMap;

public class RegexProvider {

    public static <T> HashMap<String, HashMap<String, String>> get(T t) {
        try {
            Class<? extends Object> aClass = t.getClass();
            HashMap<String, HashMap<String, String>> regex = new HashMap<>();

            for (Field field : aClass.getDeclaredFields()) {

                Annotation[] annotations = field.getDeclaredAnnotations();

                for (Annotation annotation : annotations) {

                    if (annotation instanceof Pattern) {
                        Pattern myAnnotation = (Pattern) annotation;
                        HashMap<String, String> map = new HashMap<>();
                        map.put("regex", myAnnotation.regexp());
                        map.put("message", myAnnotation.message());
                        regex.put(field.getName(), map);
                    }

                    if (annotation instanceof RegexPattern) {
                        RegexPattern myAnnotation = (RegexPattern) annotation;
                        HashMap<String, String> map = new HashMap<>();
                        map.put("regex", myAnnotation.reg());
                        map.put("message", myAnnotation.msg());
                        regex.put(field.getName(), map);
                    }
                }
            }
            return regex;
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }
}
```

Prepare Regex (Regular Expression) Service for the Insert Form

1. Define Validation Criteria for Attribute of the Selected Entity
2. Define Regex for required attributes
3. **Add Annotation into the Entity Class from the Java Validation Framework**
4. **Implement Regex Pattern Annotation and Processor**
5. **Implement Regex Controller and Test**
6. **Implement and Test Regex Service using the Client App**

Copy DesignationController and Past on “controller”



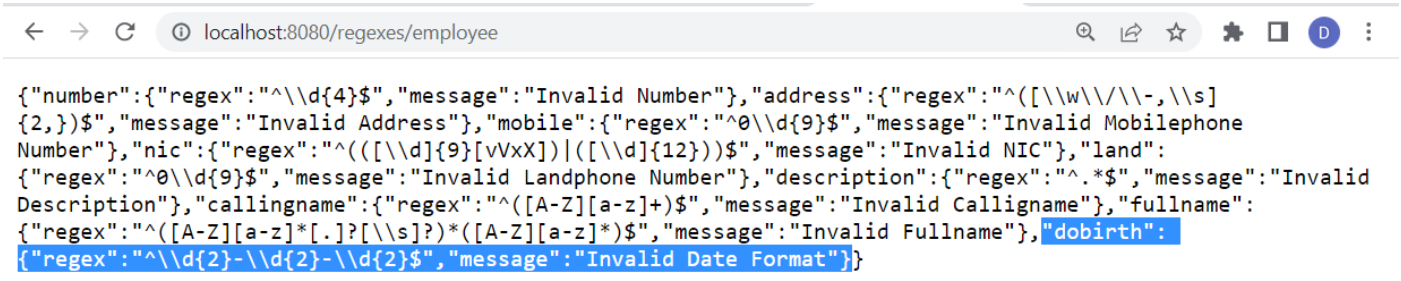


Re Run the Project and Test


```
{"number":{"regex":"^\\d{4}$","message":"Invalid Number"},"address":{"regex":"^([\\w\\/\\-,\\s]
{2,})$","message":"Invalid Address"},"mobile":{"regex":"^0\\d{9}$","message":"Invalid Mobilephone
Number"},"nic":{"regex":"^(([\\d]{9}[vVxX])|([\\d]{12}))$","message":"Invalid NIC"},"land":
{"regex":"^0\\d{9}$","message":"Invalid Landphone Number"},"description":{"regex":"^.*$","message":"Invalid
Description"},"callingname":{"regex":"^([A-Z][a-z]+)$","message":"Invalid Callingname"},"fullname":
{"regex":"^([A-Z][a-z]*[.]?[\\s]?)*([A-Z][a-z]*)$","message":"Invalid Fullname"}}
```


Use “RegexPattern” annotation to add a pattern to date of birth.

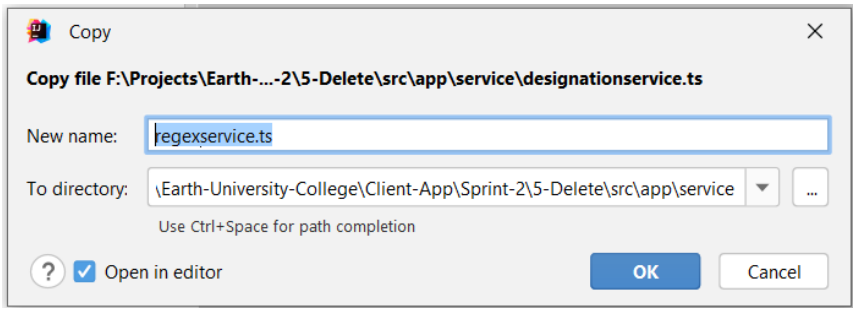
Re Run the Project and Test



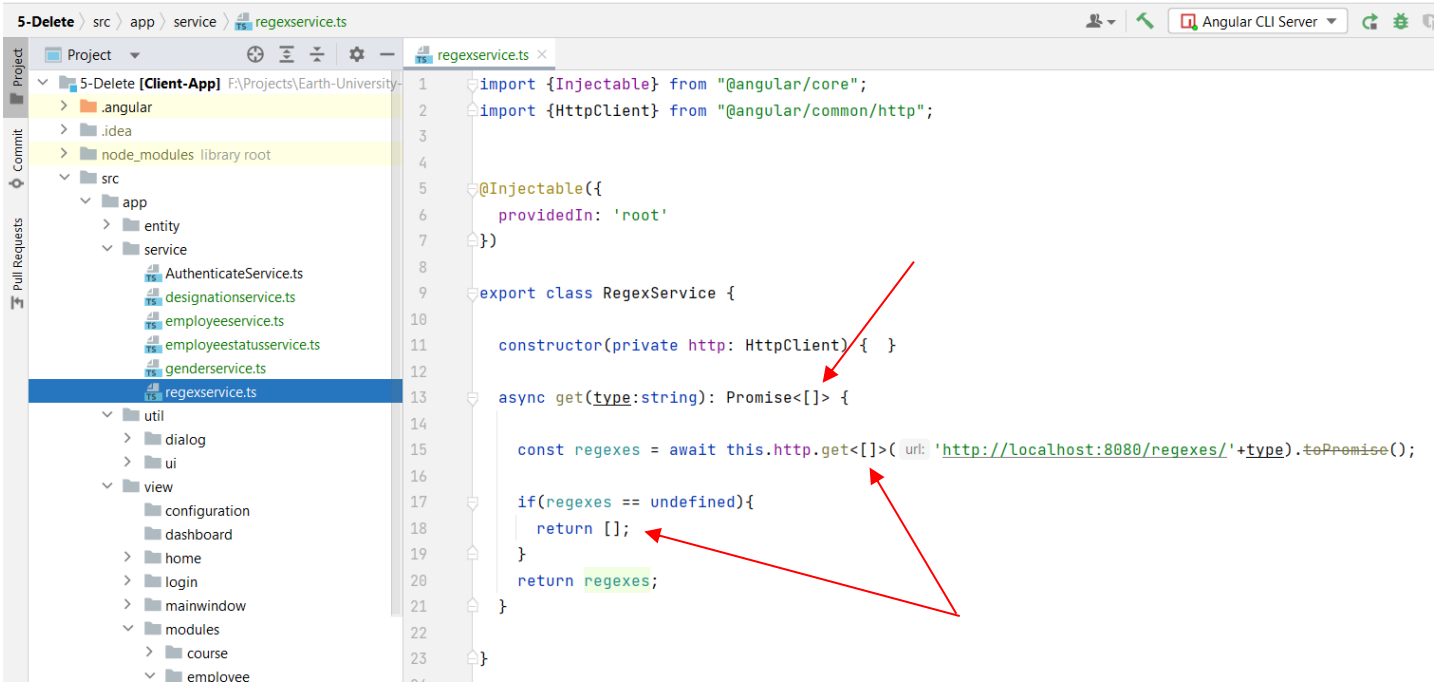
Prepare Regex (Regular Expression) Service for the Insert Form

1. Define Validation Criteria for Attribute of the Selected Entity
2. Define Regex for required attributes
3. Add Annotation into the Entity Class from the Java Validation Framework
4. Implement Regex Pattern Annotation and Processor
5. **Implement Regex Controller and Test**
6. **Implement and Test Regex Service using the Client App**

Copy "DesignationService" and Paste.
Rename "regexservice" instead of "designationservice"



Adjust Codes as follows



Open EmployeeComponnet.ts and complete following modification

employee.component.ts

```
41  imageurl: string = '';
42  @ViewChild(MatPaginator) paginator!: MatPaginator;
43
44  genders: Array<Gender> = [];
45  designations: Array<Designation> = [];
46  employeestatuses: Array<Employeestatus> = [];
47
48  regexes: any;
49
50  uiassist: UiAssist;
51
52  constructor(
53    private es: EmployeeService,
54    private gs: GenderService,
55    private ds: DesignationService,
56    private ss: EmployeestatusService,
57    private rs: RegexService,
58    private fb: FormBuilder,
59    private dg: MatDialog ) {
60
61    this.uiassist = new UiAssist(this);
62
```

employee.component.ts

```
41  imageurl: string = '';
42  @ViewChild(MatPaginator) paginator!: MatPaginator;
43
44  genders: Array<Gender> = [];
45  designations: Array<Designation> = [];
46  employeestatuses: Array<Employeestatus> = [];
47
48  regexes: any;
49
50  uiassist: UiAssist;
51
52  constructor(
53    private es: EmployeeService,
54    private gs: GenderService,
55    private ds: DesignationService,
56    private ss: EmployeestatusService,
57    private rs: RegexService,
58    private fb: FormBuilder,
59    private dg: MatDialog ) {
60
61    this.uiassist = new UiAssist(this);
62
```

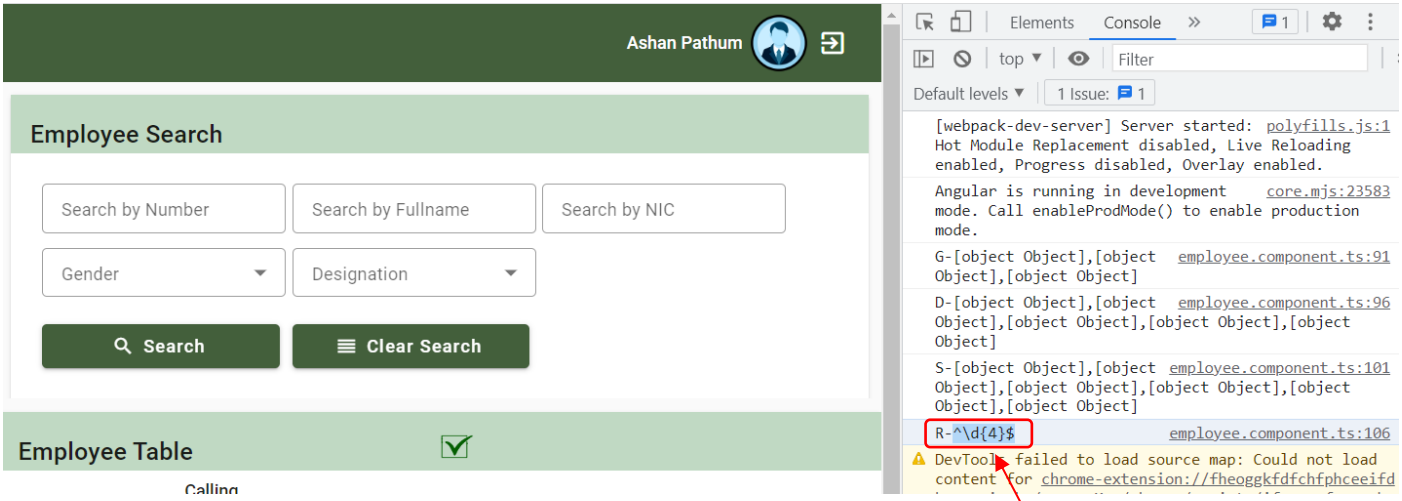
employee.component.ts

```
82  ngOnInit() {
83    this.initialize();
84  }
85
86  initialize() {
87    this.createView();
88
89    this.gs.getAllList().then((gens: Gender[]) => {
90      this.genders = gens;
91      console.log("G-" + this.genders);
92    });
93
94    this.ds.getAllList().then((dess: Designation[]) => {
95      this.designations = dess;
96      console.log("D-" + this.designations);
97    });
98
99    this.ss.getAllList().then((stes: Employeestatus[]) => {
100     this.employeestatuses = stes;
101     console.log("S-" + this.employeestatuses);
102   });
103
104   this.rs.get('employee').then((regs: []) => {
105     this.regexes = regs;
106     console.log("R-" + this.regexes['number']['regex']);
107   });
108
109   this.createSearch();
110   this.createForm();
111 }
```

localhost:8080/regexes/employee


```
{"number":{"regex":"^\\d{4}$","message":"Invalid Number"},"address":{"regex":"^([\\w\\/\\-,\\s]{2,})$","message":"Invalid
Address"},"mobile":{"regex":"^0\\d{9}$","message":"Invalid Mobilephone Number"},"nic":{"regex":"^(([\\d]{9}[vVxX])|([\\d]
```

Testing



Employee.java

```
10
11  import javax.validation.constraints.Pattern;
12
13  @Entity
14  public class Employee {
15      @GeneratedValue(strategy = GenerationType.IDENTITY)
16      @Id
17      @Column(name = "id")
18      private Integer id;
19      @Basic
20      @Column(name = "number")
21      @Pattern(regexp = "^\\d{4}$", message = "Invalid Number")
22      private String number;
23      @Basic
24      @Column(name = "fullname")
25      @Pattern(regexp = "^([A-Z][a-z]*[.]?[\\s]?)*([A-Z][a-z]*)$", message = "Invalid Fullname")
26      private String fullname;
27      @Basic
28      @Column(name = "callingname")
29      @Pattern(regexp = "^([A-Z][a-z]+)$", message = "Invalid Calligname")
30      private String callingname;
31      @Basic
32      @Column(name = "photo")
33      private byte[] photo;
```

This design ensure that the validation regex pattern is written in the Entity Class at the Server App.
This Pattern will be reused by both the Client App and the Server App

Discuss following Classes

Server App

- **@Pattern ← From Validation Framework**
- **@RegexPattern ← Yours**
- **RegexProvider ← Yours**
- **RegexController ← Yours**

Client App

- **RegexService ← Yours**
- **EmployeeComponnet ← Use and Test**

Department of Information Technology
Earth University College