CS544

Enterprise Architecture Final

Exam 2 May 2017

Name_____

Student ID _____

NOTE: This material is private and confidential. It is the property of MUM and is not to be disseminated.

1. [15 points] ***CIRCLE*** which of the following are TRUE/FALSE concerning EAI :

   T F Spring Enterprise Integration is essentially another name for Spring Remoting [JMS,AMQP,HTTP etc.]
   EXPLAIN:

   T F Spring Integration is based on lightweight messaging which is based on JMS.
   EXPLAIN:

   T F A Content Router basically determines routing based on IF-THEN-ELSE logic.
   EXPLAIN:

   T F A Message Channel is a core component of Spring Integration
   EXPLAIN:

   T F A basic capability of ESB is transport conversion.
   EXPLAIN:

2. [10 points] Consider the following AOP Aspect:

Create an Advice that executes before an *annotated* application joinpoint that has a pointcut that would identify service methods that have a OrderItem object as a parameter.

Note you need to create an additional pointcut AND the Advice.

The Advice method should calculate and print out the cost of the inventory using quantity and price fields from OrderItem.

Also show a complete implementation of an example join point.

Two Pointcuts are provided for your use.

AccountingAspect.java

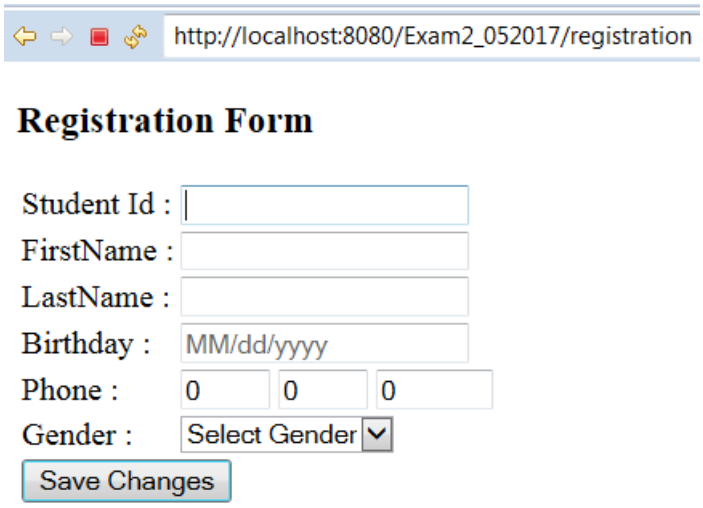
```
@Pointcut("execution(* edu.mum.service..*(..))")
public void accountingMethod() {}

@Pointcut("@annotation(edu.mum.validation.Accounting)")
public void accounting() {}
```

3. [20 points] Spring has 3 SPEL expressions that can be used to provide annotation-based access control on Service methods. We commonly used @PreAuthorize as the annotation.
   a. Explain the use & give examples of the 3 expressions using the @PreAuthorize.
   b. Explain how Spring facilitates RBAC[include the use of 1 of the 3 expressions] ***by example***
   c. Explain how Spring facilitates ABAC [include the use of 1 of the 3 expressions] ***with example***

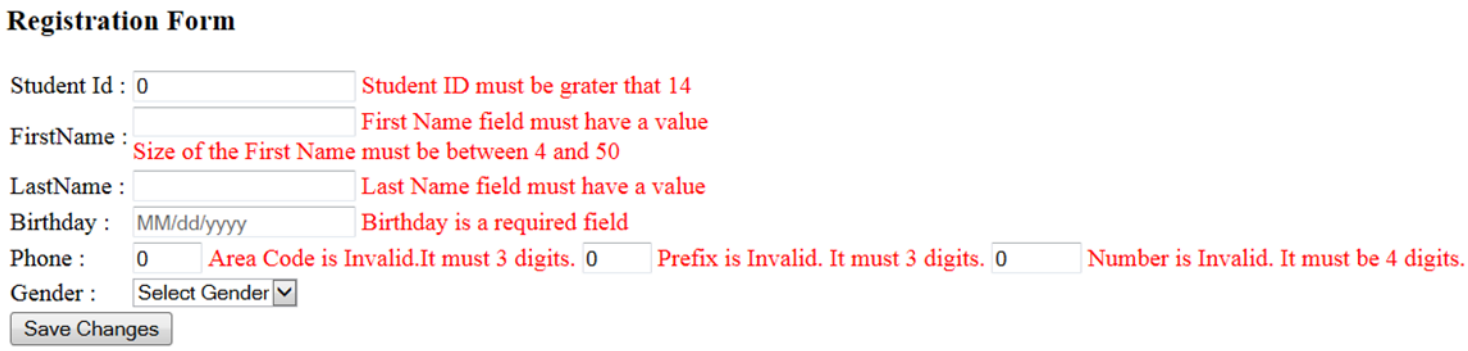
4. [15 Points] This is a student Registration form. There is validation required before the student can be entered into the system successfully. If the student information is entered correctly then a JSP page success.jsp is displayed. Below you can see the error messages resulting from wrong input. Fill in ALL the content of the supplied resources. USE BEST PRACTICES…

NOTE: In the interest of time, you only need to annotate ONE of the phone fields [all 3 look about the same]. For "3 digits", test for values between 100 & 999.

INITIAL SCREEN:



NO INPUT & "SAVE CHANGES"



WITH VALID INPUT & "SAVE CHANGES"

Registration Form

Student Id : 22
FirstName : Will
LastName : Pickett
Birthday : 12/12/1212
Phone : 222 333 4444
Gender : Female
Save Changes

SUCCESSFUL "SAVE CHANGES"

Student Save successfully

Student Id: 22
Student FirstName: Will
Student LastName: Pickett
Student Phone: 222-333-4444

StudentController.java

```
@RequestMapping(                                                    )
public String showForm(                                            ){




}


@RequestMapping(                                                    )
public String processForm (                                        ){


}


@RequestMapping(                                                  )
public String success(                          ) {


}
```

Student.java

```
public class Student {

        private int StudentId;

        private String firstName = null;

        private  String lastName  = null;

        private String gender = null;

        private Date birthday;

        private Phone phone;
```

Phone.java

```
public class Phone  {

        private Integer area;

        private Integer prefix;

        private Integer number;
```

Errormessages.properties

5. [20 points] The validation for the Student Registration in the previous problem [#4] deals with a Web UI scenario. Consider 2 other alternative scenarios for Student Registration:

   A REST service and a Batch service.

   BATCH simply requires validating: lastName & phone

   REST requires: ALL fields to be validated: The fields in problem 4 PLUS gender field. Assume the gender field requires a value.

   Fields to be validated & when to validate them:

	BATCH	WEB UI	REST
studentId		**X**	**X**
firstName		**X**	**X**
lastName	X	**X**	**X**
gender			**X**
birthday		**X**	**X**
phone	X	**X**	**X**

   Implement Validation groups for the 3 options. Give examples of annotating the fields for each group. For example annotate firstName, lastName, gender.

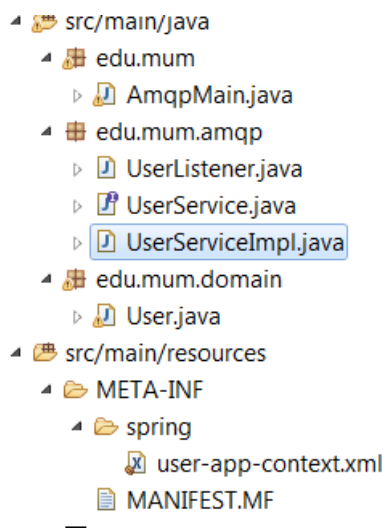
   WEB UI & REST can use the Spring MVC validation functionality for handling groups because they are HTTP based. On the other hand, BATCH group validation uses the Hibernate Validator. It is on a different layer of the N-tier architecture.

   Explain a robust validation strategy related to an N-tier architecture. Describe the value of validation at different layers in the N-tier.

6. [15 points] Complete the following AMQP messaging application.
   The project structure and UserService interface are provided.
   To be completed:
   - user-app-context.xml
   - UserServiceImpl.java
   - UserListener.java

It is a direct Exchange. The exchange name is ***user***. There is a single queue named ***userQueue***. The binding between the exchange & queue is through ***user.key***.

Project Structure:



UserService.java

```
public interface UserService {
    public void publish(RabbitTemplate rabbitTemplate);
}
```

user-app-context.xml

Fill in the parts indicated by underline _____

```
<rabbit:connection-factory id="connectionFactory" host="localhost" username="joe" password="joe"/>
<rabbit:admin connection-factory="connectionFactory" />


<!-- *********** EXCHANGE ***************** -->
      <rabbit:queue name="_____________________________________________" durable="true"/>


<rabbit:direct-exchange name="_____________________________________________" durable="true">
  <rabbit:bindings>
    <rabbit:binding  ___________="“_________________”     ______________="“_____________________” />
  </rabbit:bindings>
 </rabbit:direct-exchange>


<!-- **************** PRODUCER  -->
 associated with it...] -->
    <rabbit:template id="userTemplate" connection-factory="connectionFactory"

          reply-timeout="2000" routing-key =  "_____________________________________"

          exchange="_____________________________________________________________" />


<!--  ****************  CONSUMER ************************** -->
 <rabbit:listener-container connection-factory="connectionFactory">


 <rabbit:listener ref="queueListener" method="____________" queue-names="_________________________"
</rabbit:listener-container>

<!-- **************** LISTENER  -->

<bean id="___________________________" class="_____________________________________________" />
```

UserServiceImpl.java

```
public class UserServiceImpl implements UserService {
    public void publish(RabbitTemplate rabbitTemplate) {

       // Send 2 User messages
       User user = new User("Bill", "Mee", "bMee@usa.com");





        user = new User("Paul", "Tree", "pTree@mav.com");





    }
}
```

UserListener.java

```
public class UserListener {

       public void onMessage(User user) {



       }
```