CS544
Enterprise Architecture Final
Exam 2 Example

Name_____Eyobeil Fesseha_____

Student ID _____109347_____

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

1. [15 points]Determine which of the following are TRUE/FALSE concerning REST :

   T **F** RESTful web services are based on well-defined industry standards

   EXPLAIN: they are architectural styles consisting of architectural constraints, not industrial standards.

   T **F** An XML payload offers an advantage in a RESTful web service in that provides data validation via XML schema definition.

   EXPLAIN: It's up to the server to validate that clients send correctly encoded data in the payload.

   T F RESTful operations map directly to HTTP Methods.

   EXPLAIN:REST makes use of HTTP Get, Put, Post, Delete

   T F Both a PUT and a POST REST operation modify data on the server. Therefore neither of them are idempotent.

   EXPLAIN: both PUT and Post Rest are idempotent.

   T F REST security requires authenticating the user on every request.

   EXPLAIN: REST Architectural constraint: No client context is stored on the server between requests.

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

AuditAspect.java

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

@Pointcut("execution(* edu.mum.service..list(Integer))")
public void auditMethodList() {}

@Pointcut("@annotation(edu.mum.validation.Audit)")
public void audit() {}

@Pointcut("args(object)")
public void argsMethod(Object object) {}

@Before("auditMethod() && argsMethod(object)")
public void  doAudit(Object object) throws Throwable {
```

A. Implement example application code where the Advice method is applied.

Be sure to identify the package, the class & the join point.

Package edu.mum.service.impl;
public class MemberServiceImpl implements edu.mum.service MemberService{

Public void save(Member member){

MemberDao.save(member);

}

}

3. [15 points] Enterprise Integration Patterns [EIP] are a fundamental definition of how to do integration in a company of any significant size. Spring Integration implements those patterns.

Here is a diagram describing a basic ESB flow:

A. Itemize step by step what happens in this flow.

List 5 steps – one step per component

Identify the component and detail what it does.

If a step requires Java code to be implemented, give a code example

Step1: AMQP message from RabbitMQ broker with its header(information about the message) is changed to Enterprise Integration Message format with the enterprise format of the header by the **InboundAdaptor** component. The inbound adaptor connects the channel with the RabbitMq broker.

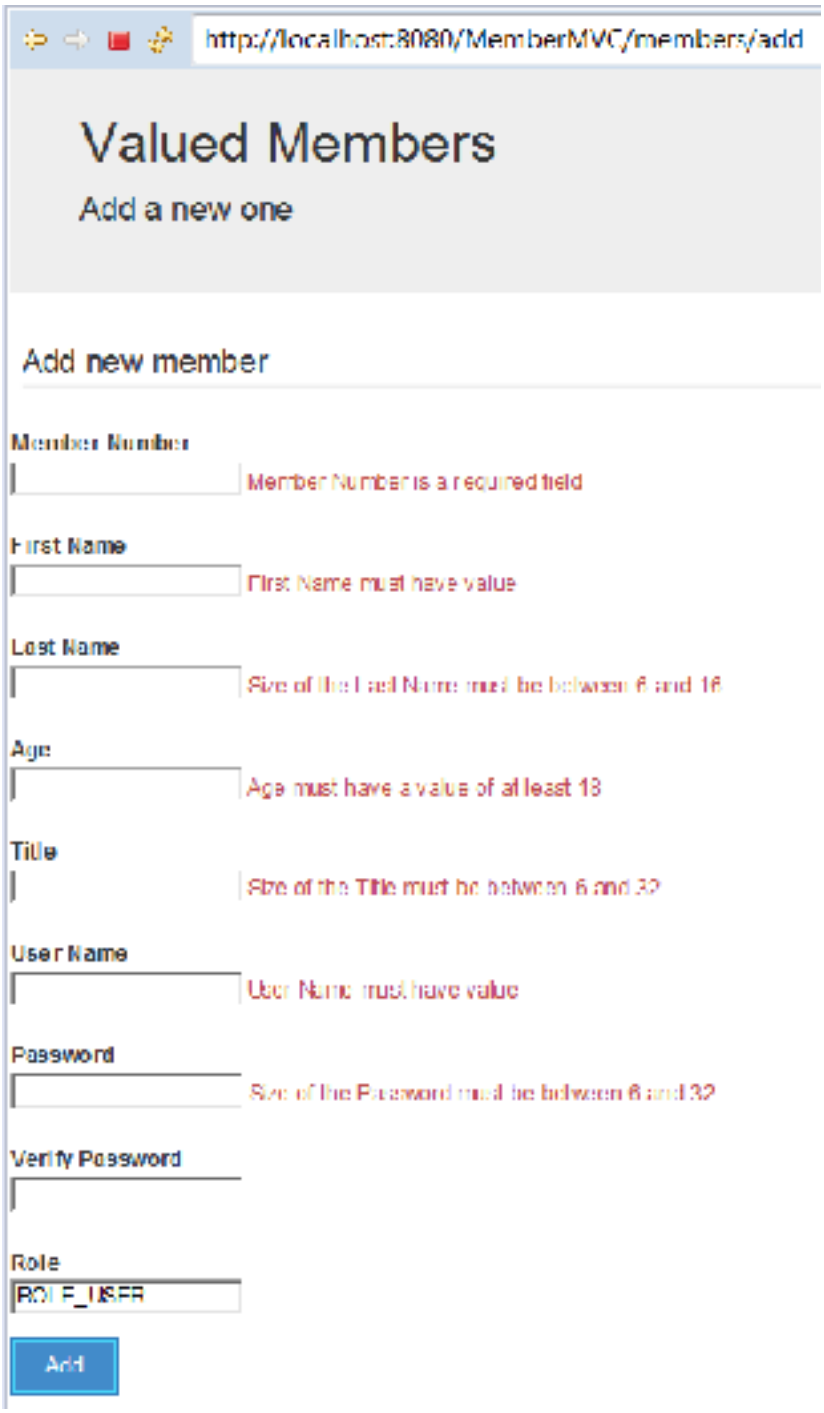
Step2: Now theEnterprise message is transmitted through a component called **Channel.**

Step3: The message is then changed from one format to another format by the **transformer** component.

Step4: again the new message format is transmitted thorough a **channel** to reach its Destination which is the JMS broker.

Step5: At last the message is in enterprise message format so it is automatically changed to JMS Message format automatically by the JMS **Outbound** adaptor. Then the ActiveMq broker delivers the message to its consumer which is in this case Warehouse. The outbound adaptor connects the channel to the outside system which is the ActiveMq broker.

4. [15 Points] The following screen displays the member entry screen. Annotate The Member & User Credentials domain objects to reflect the Validation requirements. The provided Member Controller also needs to be annotated. Also the controller methods need to be completed as necessary [including validation support].



Here is the relevant part of the Member Domain Class:

```
@Entity
public class Member {

	@Id
	@GeneratedValue(strategy=GenerationType.AUTO)
	private long id;



	@Column(length = 16)
	@NotEmpty(message="{String.empty}")
	private String firstName;



	@Column(length = 16)
	@Size(min=6, max=16, message="{Size.name.validation}")
	private String lastName;


	@Min(value= 18, message="{Min.age}")
	private Integer age;



	@Column(length = 32)
	@Size(min=6, max=32, message="{Size.title.validation}")
	private String title;


	@NotNull(message="{Integer.empty}")
	private Integer memberNumber;



	@OneToOne(fetch=FetchType.EAGER,  cascade = CascadeType.ALL)
	@JoinColumn(name="member_id")
	UserCredentials userCredentials;
```

Here is the UserCredentials:

```
@Entity(name = "Authentication")
public class UserCredentials {


        @Id
        @Column(name = "USER", nullable = false, unique = true, length = 127)
        @NotEmpty(message="{String.userName}")
       String userName;


        @Column(name = "PASSWORD", nullable = false, length = 32)
        @Size(min=6, max=32, message="{Size.password.validation}")
       String password;


        @Column( nullable = false, length = 32)
       String verifyPassword;
       Boolean enabled;



       @OneToOne(mappedBy="userCredentials", cascade = {CascadeType.PERSIST,
CascadeType.MERGE})
       private User user;
```

ErrorMessage.properties

```
      String.empty= {0} must have a value
 Size.name.validation= size of {0} must be between {2} and {1}
Min.age= {0} must have a value of at least {1}
Size.title.validation= size of the {0} must be between {2} and {1}
String.userName= {0} must have a value
Size.password.validation= {0} must be between {2} and {1}
Integer.empty={0} is required field.
MemberNumber= Member Number
firstName= First Name
age = Age
title= Title
UserCredentials.userName= User Name
UserCredentials.password= PassWord
```

@Controller
@RequestMapping(“/members”)
MemberController.java

```
public class MemberController {

       @Autowired
       private MemberService  memberService;


       @RequestMapping(value=“/add”, method= RequestMapping.GET)
       public String getAddNewMemberForm(@ModelAttribute("newMember") Member newMember) {
          return "addMember";
       }


       @RequestMapping(value= “/add”, method= RequestMapping.POST)
       public String processAddNewMemberForm(@Valid@ModelAttribute(“newmember”) Member
memberTobeAdded, BndingResult result)
{
     if(result.hasErrors()){
       return “addMember”
}


      memberService.save(memberToBeAdded);

            return "redirect:/members";

       }
```

5. [20 points] RBAC [Role Based Access Control] is widely used in Corporate Enterprises.

It has its limitations, in that it is ONLY Role based.

1. Explain the alternative Access Control Methodology, ABAC.
2. Be specific.
3. **Give an example [use case/scenario].**
4. Diagrams are excellent but be sure to explain them.
5. What are the Authorization features of Spring that accommodate an ABAC implementation?

ABAC is an attribute based access control in which its implementation is done by matching User attribute, Environment condition and Object attribute with the Access control policies or rules. The limitation of RBAC is that there is the user doesn't have the resource attribute , the user accesses the resources only through roles. ABAC solves this problem. ABAC has a set of scenarios such as Action, Asset, Environment and user which are bind together and governed byAccess control rules or policies.
Example lets say that there are many employees in a company with different branches such as department, Logistics, warehouse and soon. The company has various products and only the members of the department field are allowed to see the products on the last day of every month and they will have a discount on that day.
In this scenario the **users** are department, logistics and warehouse employees.
The **action** is reading the product
The **Asset(resources)** is the products
And finally the **environment condition** is the last day of every month.
The overall use case is the Access Control policy or rule. So the authorization engine is abide by the policy. So the users in this case the department employees have direct access to the resources based on the environment condition.
For Example they could access the list of products with this code.

```
 public  class productServiceImpl implements productService{
@PreAuthorize("hasAuthority('department_User')")
Public List<product> getAllProducts(){
   return ProductoDao.findAll();
}
}
```

The ABAC are used in various technologies such as micro services and their implementation is through XACML which is the extensible Access Control Markup language,,, although they solve the problem RBAC, they have some limitations like

auditing problem and matching the management of the rules and attribute is quite difficult.

The authorization features of Spring ABAC are preauthorize, post Authorize, pre filter and post filter. In the preauthorize feature,Spring ABAC supports rule based access, permission based access and custom rule authorization.