

CS544
Enterprise Architecture
Exam 2 July 2017

Name__ Yaried Tekie_____

Student ID _____109025_____

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 Security :

T F Spring security only supports the authentication model HTTP Basic defined by RFC 1945 which is the most popular authentication mechanism in the web.

EXPLAIN:

Not only HTTP Basic, it also supports HTTP Digest, HTTP X.509, JOSSO etc

T F Spring Security Groups can be used to implement RBAC with in the Spring Framework

EXPLAIN:

Spring security group allow authorization based on role level of the group.

T F Digest authentication uses Base64 encoding to transmit encrypted username/password

EXPLAIN:

IT is not Digest, it is Basic authentication that uses Base64 encoding to transmit username/password as plaint text

T F Authorization refers to validating unique identifying information about each system user.

EXPLAIN:

NO this is authentication. Authorization is the process of allowing a user to access a resource.

T F Permission Based Access allows for fine grained access control.

EXPLAIN: NO, custom based rule is the one that allow fine grained access control.

2. [15 points] AOP is a Spring Core Technology. It is used in numerous places within the Spring Framework, itself. Explain the fundamentals of Spring's AOP implementation; how it works, how it relates to AspectJ, with examples of its usage within Spring.

To help in your explanation of how it works consider the following use case:

A client application needs to access a server application over the network. For monitoring purposes, it is necessary to log calls to save [save(Object object)] methods at the service tier. Using AOP terminology, describe what would need to be implemented. Be specific with respect to Pointcut & Advice syntax.

For example:

```
Class FooServiceImpl {  
  
    public void save (Foo foo) {  
        fooDao.save (foo);  
    }  
  
    Public List<Foo> findAll() {  
        return fooDao.findAll();  
    }  
  
    Public Foo findOne(Long id) {  
        return fooDao.findOne(id);  
    }  
}
```

ANSWER:

AOP is Aspect oriented programming that is defined as the implementation of cross cutting concerns.it means it defines a in one place, functionality that is need to be in a multiple places in the code through out the application.

Aop supports separation of concern that is it separate the business logic from the cross cutting concerns. It centralize and modularize codes that are intermixed and scattered through out the application so that you can easily maintain them and change in single place.

Spring Aop is a proxy-based approach. That is it wraps the orginal object by a proxy object. At the time of DAO injection under service object spring find out that there is some aspect is configured for DAO, so it injects the proxy object instead of the actual object. Now when the actual call is made to any method inside DAO, proxy applies the aspect and then call the actual target object. This is called runtime waving.

Like Spring AOP, AspectJ is implementation of cross cutting concern. But is is intended to be the complete AOP solution. Unlike spring AOP , Aspectj supports compile time waving and and field level advice. It is more powerful but complex than spring AOP. It works for any java code, not only in spring managed Bean as of Spring AOP.

```
@Aspect
@Component
public class AspectLogging(){

    @PointCut("execution(* edu.mum.service..save(object))")
    public void loggingadvice(Object object){}

    @Before("loggingadvice(object)")
    public void saveAdvice(JoinPoint joinpoint, Object object) throws Throwable{

    }
}
```

3. [20 Points] This is a Member Registration form. There is validation required before the member can be entered into the system successfully. If the member information is entered correctly then a JSP page members.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...

INITIAL SCREEN:

localhost:8080/Exam2_072017Sol/members/addMember/

▼ 80% ↺ 🔍 Search

Travel Cost Sharing

TripMember ManagementPaymentReportsWelcome admin🔗 Logout

First Name

Last Name

Nick Name

Gender

Select a gender▼

Birth Date

Email



Phone

Username

Password

Add

NO INPUT & “ADD CHANGES”

localhost:8080/Exam2_072017Sol/members/addMember/80% Search

Travel Cost Sharing

[Trip](#) [Member Management](#) [Payment](#) [Reports](#) [Welcome admin](#) [Logout](#)

First Name	<input type="text"/>	First Name must have a value
Last Name	<input type="text"/>	Size of the Last Name must be between 3 and 50
Nick Name	<input type="text"/>	
Gender	<input type="text" value="Select a gender"/>	Gender is required
Birth Date	<input type="text"/>	
Email	<input type="text"/>	
Phone	<input type="text"/>	Phone must have a value
Username	<input type="text"/>	Username must have a value
Password	<input type="text"/>	Password must have length between 4 and 100
<input type="button" value="Add"/>		

Invalid Email input

localhost:8080/Exam2_072017Sol/members/addMember/80%Search

Travel Cost Sharing

TripMember ManagementPaymentReportsWelcome adminLogout

First Name

Peter

Last Name

Piper

Nick Name

Gender

Male

Birth Date

Email

pp

Not a well-formed Email address

Phone

123-456-7890

Username

Peter

Password

Add

Successful Member Addition:

localhost:8080/Exam2_072017Sol/members80%Search

Travel Cost Sharing

TripMember ManagementPaymentReportsWelcome adminLogout

Add Member

	First Name	Last Name	Nick Name	Gender	Birth Date	Email	Phone
1	Peter	Piper		MALE		pp@cc.com	123-456-7890

MemberController.java

```
@Controller
@RequestMapping("/members")
public class MemberController {

    @Autowired
    private MemberService memberService;

    @RequestMapping(value="", method=RequestMethod.GET)
    public String getMembers( Model model ) {

        model.addAttribute("listMember", memberService.findAll());

        return "members";

    }

    @RequestMapping(value="/add", method=RequestMethod.GET)
    public String addMember( @ModelAttribute("newMember") Member member) {

        return "addMember";

    }

    @RequestMapping(value="/add", method=RequestMethod.POST)
    public String addMember(@ModelAttribute("newMember") @ValidMember member, BindingResult result) {

        if(result.hasErrors()){

            return "addMember";

        }

        memberService.save(member);

        return "redirect:/members";

    }

}
```

Here is the relevant part of the Member Domain Class:

@Entity

public class Member {

 @Id

 @GeneratedValue(strategy = GenerationType.*AUTO*)

private Long *id*;

 @NotEmpty

 @Column(length = 20)

private String *firstName*;

 // @Size(min=3, max=50, message="{lastName.Size}")

 @EmptyOrSize(min = 3, max = 50, message = "{size.name.validation}")

private String *lastName*;

private String *nickName*;

 @NotNull

private Gender *gender*;

 @email

private String *email*;

 @NotEmpty

private String *phone*;

 @Temporal(TemporalType.*DATE*)

 @DateTimeFormat(pattern = "yyyy-MM-dd")

private Date *birthDate*;

 @valid

 @OneToOne(cascade = CascadeType.*ALL*)

 @JoinColumn(name = "*credential_id*")

 Credential *credential*;

Here is the Credentials Class:

```
@Entity
public class Credential {

    @NotEmpty
    @Id
    private String username;

    @Size(min=4, max=100, message = "{password.Size}")
    private String password;

    private Boolean enabled = Boolean.TRUE;

    @OneToOne(mappedBy = "credential", cascade = CascadeType.PERSIST)
    private Member member;

    @valid
    @OneToMany(fetch = FetchType.EAGER, cascade = CascadeType.ALL, orphanRemoval = true)
    @JoinColumn(name = "username")
    private List<Authority> authorities = new ArrayList<Authority>();

    @Transient
    private List<String> authorityList = new ArrayList<>();
```

ErrorMessage.properties

NotEmpty = {0} field must have a value

NotNull = {0} is required

size.name.validation = Size of the {0} must be between {2} and {1}

Email = {0} must have a valid syntax

password.Size = {0} must have a length between {2} and {1}

firstName = First Name

lastName = Last Name

nickname = Nick Name

birthdate = Birth Date

gender = Gender

email = Email

phone = Phone

4. **[15 points]** Messaging is basic to scalable enterprise architectures. We covered two messaging technologies, JMS & AMQP. Explain the fundamentals of messaging.

Be sure to cover: the types of messaging, the messaging architecture, and the differences between the two, JMS & AMQP and how they are implemented.

Be specific. Give examples. Diagrams are good; be sure to explain them.

Answer:

Messaging is a loosely coupled asynchronous and reliable communication between applications.

The main type of messaging are

1. point to point
2. publish –subscribe

point to point

Message is sent from one application to another application via a queue. Here there is only one consumer but we can have multiple producers.

Publish – subscribe

Message(publish) is sent from producer(publisher) to multiple consumers(subscribers) via topic. Here there is only one producer.

The publisher is not required to know information about the subscribers.

It is easily deployable, dynamic and flexible.

The main difference between JMS and AMQP

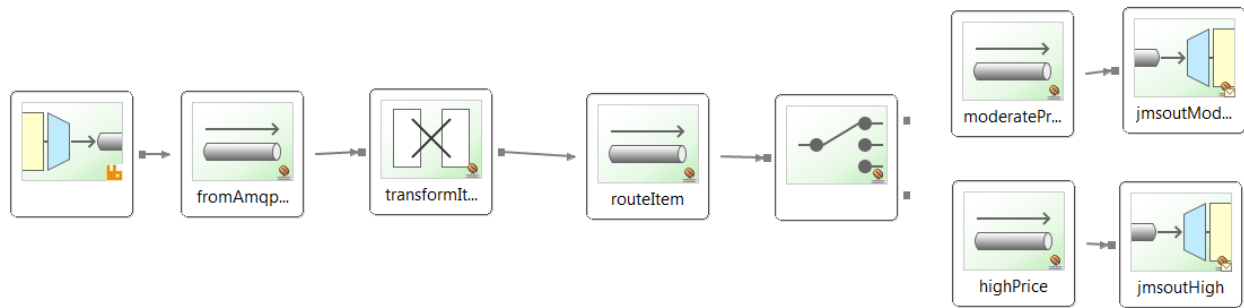
1. JMS has queue and topic but AMQP has only queue and exchange.
2. Unlike JMS, in AMQP message is not pushed directly to the queue, it is first pushed to the exchange.
3. JMS is an API and AMQP is a protocol. Therefore, JMS client can use AMQP as a protocol.
4. Unlike AMQP, JMS is not required to know how the message is formed or created.
5. AMQP supports multiple languages and it is a cross-platform messaging standard.
6. AMQP has a high quality messaging system and more secured than JMS.

5. [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.

Explain the fundamental aspects of Spring Integration. Why is it necessary & valuable? Describe the 3 main components. Drawing on the demo from class [Routing an order through the “enterprise”], give details on some of the EIP components.

Be specific. Give examples. Diagrams are good but be sure to explain them.

Here is a diagram that you should use to describe [some] components and an ESB type flow:



Answer:

Spring integration is a concise and clear approach of Enterprise Application Integration. It is a lightweight messaging system that allow integration capability on our spring application. As a messaging strategy it allows a transfer of information quickly and with a high level of decoupling between different components in our application.

Provide a simple model for implementing complex enterprise integration solutions.

Facilitate asynchronous, message-driven behavior within a Spring-based application.

Promote intuitive, incremental adoption for existing Spring users.

Main components are:

Message: contains information that is transferred between different part of an application or sent to an external system. It consists of Header and payload.

Header is a wrapper of map that contains meta-information

Payload is a pure java class that contains information.

Message Channel: it is a pipe that contains two end points and through which message travel.

Message Endpoint: it is an abstraction layer between application code and messaging framework. It is message consumers and message producers.

Components:

Channel adapter: Connects the application to an external system (unidirectional).

Gateway: Connects the application to an external system (bidirectional).

Service Activator: Can invoke an operation on a service object.

Transformer: Converts the content of a message.

Filter: Determines if a message can continue its way to the output channel.

Router: Decides to which channel the message will be sent.

Splitter: Splits the message in several parts.

Aggregator: Combines several messages into a single one.

6.[15 points] The Spring framework is the “example” architecture that we used in this course. It emphasizes good design, best practices and use of design patterns.

Explain the value of the framework. Things to consider:

N-Tier; Separation of Concerns; Different types of N-tier; Distributed capabilities;

The characteristics & value of a framework.

Be specific. Give examples. Diagrams are good but be sure to explain them.

Answer:

Spring Framework Is a lightweight architecture with there tire architecture mainly called presentation tier, business tier and persistent tier.

Spring framework make enterprise java application development as easy as possible by following good programming practice such as

1. POSO based programming
2. Separation of concern
3. Flexibility.

The advantages of spring framework are:

1. Spring make application development simple
2. DI give flexibility in bean wiring
3. Separates business logic with cross cutting concern
4. Adhere DRY principle
5. Increase reliability and reduce programming time.
6. Helps enforce best practice and rules
7. Standard based
8. Design pattern based.