Enterprise Integration with Spring Certification Mock Exam

Smaller than the real [Enterprise Integration with Spring](#) exam (EIwS 1.x), this mock exam is composed of 25 questions. You have 44 minutes to complete it and a minimal score of 76% to reach (19 right questions).

Questions

Web Services

Question WS.1

What are the object-relational mapping technologies supported by Spring Web Services for XML marshaling? (unique answer)

1. JAXB
2. JiBX
3. Castor
4. All of them

Question WS.2

In terms of best practices, what is the approach promoted by Spring Web Services? (select one)

1. Generate XSD from parameter type and method signature
2. Contract-first approach
3. Code-first approach
4. All of them

Question WS.3

Which statements describe usage of the WebServiceTemplate?

1. Send Source Objects
2. Receive response message as Result
3. Handle incoming SOAP request
4. Marshal objects to XML before sending them

REST

Question REST.1

Which statements are NOT true about Spring MVC? (select one or many)

1. Implements the Java API for RESTful Web Services (JSR-311) standard
2. Supports both server and client side REST functionality
3. Handle content negotiation
4. Supports the *@PathParam* annotation

Question REST.2

According to the REST principle, what is the most appropriated HTTP request designed to update a client account?

1. POST /account/123
2. PUT /account/123
3. POST /account?id=123
4. PUT /account?id=123

Question REST.3

Which statement is FALSE given the following code compiled with debugging information? (select one)

```
@RequestMapping(value="/account/{accountId}", method=RequestMethod.DELETE)
public void remove(@PathVariable("accountId") Account accountId, Model model) {
```

1. Another *@RequestMapping* annotation has to decorate the *AccountController* class which belongs this method
2. The *"accoundId"* URI template variable specified in the *@PathVariable* annotation is mandatory
3. The *RequestMethod.DELETE* enumeration does not exist
4. All

Remoting

Question REM.1

Which remoting technologies the Spring Framework supports? (select one or many)

1. HTTP invoker
2. FTP
3. RMI
4. Burlap

Question REM.2

Using the Spring RMI supports, what statements is NOT true? (select one or many)

1. Exchanged classes do not have to implements the *Serializable* interface
2. RMI stubs are not mandatory
3. Binding in the RMI registry is automatically done by Spring

4. RMI *RemoteException* checked exceptions are wrapped to the Spring runtime *RmiRemotingException*

JMS

Question JMS.1

Which statement(s) is true regarding the Spring JMS Listener container? (select one or many)

1. Based on Message Driven Beans (MDB) of the EJB worlds
2. Based on the Message Driven POJOs model
3. Automatically send return value to the Reply-to queue
4. Could only be deployed into a Java EE application server

Question JMS.2

What are the properties that could be set to the *JmsTemplate*? (select two)

1. *MessageConverter*
2. *MessageListenerAdapter*
3. *DestinationResolver*
4. *QueueResolver*

Transaction

Question TX.1

By using the Spring jms namespace, wich statements are true regarding the *acknowledge="transacted"* attribute value of the JMS *<listener-container>* element? (select two)

1. JMS transaction has to be manually closed by the client
2. It is complementary to the *transaction-manager* attribute of the *<listener-container>* element
3. While receiving a JMS message, its JMS session is available to the *JmsTemplate*
4. JTA transaction manager does not support this configuration

Question TX.2

While using a local JMS transaction to receive a JMS message and a JDBC update is required to process the message, which statements are true?

1. Spring applies the best effort strategy by commiting JMS transaction then Database
2. Spring applies the best effort strategy by commiting Database then JMS
3. Transaction do not work with JMS, only with database
4. Application should handle duplicated messages

Spring Batch

Question BA.1

Where may be stored the Spring Batch metadata? (select one)

1. In memory
2. In database
3. All
4. None

Question BA.2

Which is NOT true about the Spring Batch Domain Language?

1. A *Job* is composed of one or more *Steps*
2. All *JobInstance* of the same job have the same *JobParameters*
3. All *JobInstance* of the same job have the same *JobExecution*
4. All the *JobExecution* of the same *JobInstance* have the same *JobParameters*

Question BA.3

What are the two key interfaces provided by Spring Batch to perform bulk reading and writing?

1. ItemReader
2. FileReader
3. ItemWriter
4. FileWriter

Question BA.4

What is the method signature of a Spring Batch writer in charge of producing *Account*?

1. **void** write(Account account) **throws** Exception
2. **void** write(Collection<Account> accounts) **throws** Exception
3. **void** write(List<Account> accounts) **throws** Exception
4. **void** write(Set<Account> accounts) **throws** Exception

Question BA.5

In Spring Batch, which statements describe a *FieldSetMapper*?

1. May map the data obtained from a *LineTokenizer* into an object
2. May be used by a *FlatFileItemWriter*
3. Provides a mapping algorithm based on bean property paths
4. All

Spring Integration

Question SI.1

In Spring Integration, how to programmatically create a new String Message? (select two)

1. new GenericMessage("test");
2. new GenericMessage().withPayload(payload).withHeader("key", "value");
3. new GenericMessage("test").addHeader("key", "value");
4. MessageBuilder.withPayload("test").setHeader("key", "value").build();

Question SI.2

In Spring Integration, how to declare a channel queue with a capacity limit of 5 messages? (select one)

1. <channel id="myChannel" queue=”5”/>
2. <channel id=”myChannel” queue=”true” capacity=”5”/>
3. <channel id=”myChannel”><queue capacity=”5”></channel>
4. <channel id=”myChannel” queue=”true” size=”5”/>

Question SI.3

Which statement is true about <chain> in Spring Integration? (select two)

1. Chains may have splitters has sub-elements
2. Null may be used to filter message
3. Nested chains are not supported
4. All endpoints of the chain have to not return a void

Question SI.4

In Spring Integration, how to activate a passive endpoint? (select one or more)

1. With a trigger
2. WIth a poller
3. With a *task-executor*
4. With the *status="active"* attribute of the *channel* tag

Question SI.5

Is it possible that a message send to a pollable channel will be never consumed by and endpoint that is configured without any poller?

1. Yes
2. No

Question SI.6

What are default headers used by using the Message Splitter and the Message Aggregator provided by Spring Integration? (select two)

1. CORRELATION_ID
2. MESSAGE_GROUP
3. SEQUENCE_SIZE
4. SEQUENCE_INDEX

Question SI.7

In Spring Integration, which statements are true about Adapters? (select one or many)

1. They are bidirectional
2. May be either inbound or outbound
3. Could begin a message flow
4. Could end a message flow

Question SI.8

While using a DirectChannel in Spring Integration, which statements are true?

1. Transactional context is available on Consumer and Producer
2. Security context from Spring Security should only be access from Producer
3. Supports failover and load-balancing strategy
4. Could delegates to an instance of TaskExecutor to perform the Message dispatch

Response

Web Services

Question WS.1

Answer 4 is correct. The Object/XML Mapping module in the Spring Web Services distribution (called Spring OXM) supports JAXB 1 and 2, Castor XML, XMLBeans, XStream and JiBX,. And because it is a separate module, you can use it in non-Web services code as well.

Question WS.2

Answer 2 is correct. Spring Web Services aims to facilitate contract-first SOAP service Development. This approach provides a loose coupling between contract and implementation.
Opposed to CXF for instance, the code-first approach is not supported by Spring WS.
Generation XSD from java code could be done with external tools. It's not recommended.
Instead, prefer using XML request / reply samples to generate XSD.

Question WS.3

The correct answers are 1, 2 and 4. The WebServiceTemplate is used on the client-side.
Methods like sendSourceAndReceiveToResult or marshalSendAndReceive are available.
To handle incoming SOAP message, Spring WS requires exposing an Endpoint.

REST

Question REST.1

The correct answers are 1 and 4.
Opposed to frameworks like CXF or Jersey, Spring MVC does not implement the JAX-RS API. It is an alternative.
JAX-RS 1.0 only supports server side REST service. Spring MVC supports both and

provides a RestTemplate for client side.
The Spring MVC *ContentNegotiatingViewResolver* class selects the view that resembles the representation requested by the client then delegates to other view resolvers. It may use the *Accept* HTTP request header to list the media types that it understands.
The *@PathParam* is a JAX-RS annotation. The Spring MVC equivalent is *@PathVariable*.
Which statements are NOT true about Spring MVC?

Question REST.2

The correct answer is 2.
The POST method should be used to create a new Account.
The PUT method is used to update an new Account or create a new one if id is provided by the client.
Using URL path instead of HTTP parameters is recommended.

Question REST.3

The correct answer is 4.
A *@RequestMapping* on the class level is not required. Without it, all paths are simply absolute, and not relative.
If the URI template variable name matches the method argument name you can omit the URI variable template to bind as long as the code is compiled with debugging information. Spring MVC will match the method argument name to the URI template variable name.
The *RequestMethod* enumerates GET, HEAD, POST, PUT, DELETE, OPTIONS and TRACE methods.

Remoting

Question REM.1

The correct answers are 1, 3 and 4.
Spring HTTP invokers use the standard Java serialization mechanism to expose services through HTTP.
FTP is not what Spring called a remoting technology.
Spring's support for RMI enables to expose business services through the RMI infrastructure.

Spring's support for Burlap enables to expose business services using the lightweight XML HTTP-based protocol provided by Caucho.

Question REM.2

The correct answers are 1 and 4.
Java serialization is still used thus exhanged class should continue to implement the *Serializable* interface.
No more stub: dynamic proxy is created through the *RmiProxyFactoryBean.*
The *RmiServiceExporter* binds exported service into the RMI registry.
RMI *RemoteException* checked exceptions are wrapped to the Spring runtime *RemoteAccessException*, not *RmiRemotingException.*

JMS

Question JMS.1

The correct answers are 2 and 3.
Spring JMS Message Listener enables asynchronous message reception with Message-Driven POJOs. In a fashion similar to a Message-Driven Bean (MDB) in the EJB world, the Message-Driven POJO (MDP) acts as a receiver for JMS messages.
The result of a MDP is sent to the Destination (if one exists) defined in the JMS *Reply-To* property of the original Message.
Spring JMS listener container may be used in a Java SE application.

Question JMS.2

The correct answers are 1 and 3.
The *MessageConcerter* interface enables to specify a converter between Java objects and JMS messages.
MessageListenerAdapter is used by JMS message listener container, not *JmsTemplate*.
The *DestinationResolver* interface enables to resolve JMS destinations.
The *QueueResolver* class or interface does not exist.

Transaction

Question TX.1

The correct answers are 3 and 4.
The first question refers to the *client* mode of the *acknowledge* attribute.
Specifying a *transaction-manager* is an alternative to the transacted acknowledge.
The transaction opened by a JMS message listener is available for the JmsTemplate. Internally, the JmsTemplate calls the ConnectionFactoryUtils.doGetTransactionalSession(…) method to reuse the current JMS session.
To use a JTA transaction manager, the transaction-manager attribute of the JMS *<listener-container>* element has to reference an external JTA *PlatformTransactionManager* (e.g. Spring's *JtaTransactionManager*). In this case, the acknowledge *attribute* is ignored.

Question TX.2

The correct answers are 2 and 4.
Spring tries to bring JMS and JDBC commits close together. JDBC commits appends before the JMS transaction. For external reasons to the application, the JDBC statements commits but the JMS transaction failed then rollback. Thus the JMS Message is left to the queue. In this scenario, a same message is processed twice. Idempotence should be handled by the application design.

Spring Batch

Question BA.1

The correct answer is 3.
Spring Batch used a JobRepository to persist its metadata. Spring Batch provides 2 build-in implementations: an In-Memory job repository and a Database job repository.

Question BA.2

The correct answers are 2 and 3.
A Job has one to many steps.
The contract between *Job*, *JobInstance* and *JobParameters* can be defined as: *JobInstance* = *Job* + *JobParameters*. So the same Job with different *JobParameters* gives another *JobInstance*.
JobExecution is contextual. For instance, a *JobInstance* may have 2 different *JobExecution*: the first in failure and the second in success.
JobExecution references a *JobParameters* through its *JobInstance*.

Question BA.3

The correct answers are 1 and 3.
ItemReader is the interface for providing data from many different types of input.
FileReader is a *java.io* convenience class for reading character files.
ItemWriter is the basic interface for generic output operations.
FileWriter is a *java.io* convenience class for writing character files.

Question BA.4

The correct answer is 3.
The signature should be compliant with the following *write* method signature of the *ItemWriter<T>* interface:
void write(List<? **extends** T> items) **throws** Exception;
The write method may handles more than a single Account.
A *Set* and a *Collection* does not match the previous signature.

Question BA.5

The correct answer is 1.
A *LineTokenizer* splits string obtained typically from a file into tokens embbeded into a *FieldSet*. The resulting *FieldSet* can then be passed to a *FieldSetMapper*.
A *FieldSetMapper* may be indirectly used by *FlatFileItemReader*, not a *FlatFileItemWriter*.
The *BeanWrapperFieldSetMapper* class is a *FieldSetMapper* implementation based on bean property paths. *FieldSetMapper* itself does not provide any algorithm.

Spring Integration

Question SI.1

The two correct answers are 1 and 4. The Message interface only defines retrieval methods for its payload and headers. No setters are available. A Message is immutable and thus Message cannot be modified after its initial creation (through constructor). The GenericMessage class provides 2 constructors: the first with a payload and the second with a payload and headers.
The MessageBuilder helper class is a more convenient way to construct Messages. Internally, its build() method calls the GenericMessage constructor.

Question SI.2

The correct answer is 3. To create a QueueChannel, the <queue/> sub-element has to be used. If no value is provided for the 'capacity' attribute on this <queue/> sub-element, the resulting queue is unbounded.

Question SI.3

Correct answers are 1 and 2.
Sub-elements of a chain are then filters, transformers, splitters, and service-activators.
Filter uses null to filter message.
Nested call to another chain from within a chain and then come back and continue execution within the original chain. The use of a <*gateway*> element is required.
The last element in the chain may not produce messages (ie. Service activator). Thus the *output-channel* attribute is optional.

Question SI.4

Correct answer is the number 2.
In Spring Integration, Pollable Channels are capable of buffering Messages within a queue. A consumer can only receive the Messages from such a channel if a poller is configured.
To enable concurrency, a task executor may be configured at the poller level.

Question SI.5

Correct answer is the number 2.
To configure the polling interval or cron expression for an individual endpoint, provide a 'poller' element with one of the scheduling attributes, such as 'fixed-rate' or 'cron'. If no poller is provided, then a single default poller must be registered within the apllication context. If no default poller is defined, the Spring application context failed at start with the following error message:
No poller has been defined for endpoint 'org.springframework.integration.config.ConsumerEndpointFactoryBean#0', and no default poller is available within the context.

Question SI.6

Correct answers are 1 and 3.
Correlation determines how messages are grouped for aggregation. In Spring Integration correlation is done by default based on the CORRELATION_ID message header. Messages with the same CORRELATION_ID will be grouped together.
The Spring Integration default implementation of a Release Strategy consults the SEQUENCE_NUMBER and SEQUENCE_SIZE headers of each arriving message to decide when a message group is complete and ready to be aggregated.
MESSAGE_GROUP and SEQUENCE_INDEX are not default Spring Integration headers.

Question SI.7

Correct answers are 2, 3 and 4.
Channel Adapters are unidirectional. They may be either inbound or outbound.. An inbound "Channel Adapter" endpoint connects a source system to a MessageChannel: it begins a message flow. An outbound "Channel Adapter" endpoint connects a MessageChannel to a target system : it ends a message flow.

Question SI.8

Correct answers are 1 and 3.

DirectChannel enables a single thread to perform the operations on "both sides" of the channel. So transaction and security contexts are propagated from producer to consumer through a thread-local.

The DirectChannel internally delegates to a Message Dispatcher to invoke its subscribed Message Handlers, and that dispatcher can have a load-balancing strategy. The load-balancer also works in combination with a boolean failover property.
The ExecutorChannel delegates to an instance of TaskExecutor to perform the dispatch. The send method typically not blocks.
