1. Что такое JAXB, как аннотации JAXB можно применить в контексте SOAP-сервисов?

Java Architecture for XML Binding - позволяет Java классам ставить в соответствие XML-представления.
Для применения JAXB-аннотаций в SOAP-сервисом нужно, например, объявить `@XmlRootElement`, `@XmlAccessorOrder`, `@XmlAccessorType`, `@XmlType` перед классом, а `@XmlElement`, `@XmlAttribute` перед полем.

2. Предположим, что я разработал веб-сервис, но не смог предоставить WSDL разработчикам программного-клиента, а предоставил только XSD. Смогут ли они коммуницировать с моим веб-сервисом в случае, если им известна только XSD-схема?
Нет, т.к. XSD задает формат данных, а wsdl - функционал веб-сервиса. Мы сможем определить лишь формат данных, используемых в wsdl,

3. Является ли сгенерированный JAX-WS клиент thread safe или нет? Если нет, то как можно решить данную проблему, и коммуницировать с сервисов из нескольких потоков?

Сервисы сгенерированного клиента являются threadSafe. Однако их прокси не потокобезопасны. Чтобы сделать прокси потокобезопасными, можно использовать пул прокси, где каждому потоку предоставляется на время свой собственный прокси, которым владеет только данный поток. По окончанию работы с прокси, поток возвращает прокси в пул для дальнейшего использования другими потоками.

4. Предположим, что одним из нефункциональных требований к вашему сервису является поддержка throttling (умышленное ограничение количества одновременно выполняемых запросов; если поступает новый запрос, а в это время уже выполняется максимально разрешенное количество, то необходимо прервать выполнение запроса, например, путем выброса ThrottlingException (класс исключения нужно создать самостоятельно)). Как можно это реализовать? Реализуйте в одной из ваших работ.

Одним из способов реализации throttling с помощью Spring является создание реализации класса-наследника HandlerInterceptorAdapter и добавления в него логики проверки текущего количества обрабатываемых запросов.

Метод preHandle выполняется при получении запроса сервисом. Он увеличивает счетчик обрабатываемых запросов и проверять, не превысило ли число лимит. При достижении максимального количества, бросается ThrottlingException, а пользователю возвращается сообщение об ошибке с просьбой повторить позже.

Метод postHandle выполняется перед отправкой ответа сервисом. В нем счётчик уменьшается.

Ссылка на коммит с реализацией:
https://github.com/mopkoff/wst/commit/98e86f3d7f11556a31533b056a09cdb5787d2b59

5. Модифицируйте REST и SOAP сервисы так, чтобы для CREATE, DELETE, UPDATE операций требовалась Basic аутентификация. Соответственно внесите такие же изменения в клиентские приложения. Login / Password для простоты можно захардкодить.

Для реализации Basic Аутентификации средствами Spring, необходимо определить набор эндпоинтов, которые попадают под Security и тип менеджера аутентификации, где будут храниться роли.

Ссылка на коммит с реализацией серверной части:
https://github.com/mopkoff/wst/commit/658ecddbf486f1555c13acf165a32a9f916c047a

В запрос, который отправляется с клиента необходимо добавить заголовок **Authorization**, значением которого будут логин и пароль закодированные кодировкой Base64 с префиксом Basic.

Ссылка на коммит с реализацией клиентской части:

https://github.com/mopkoff/wst/commit/d913639d78bb15ae152a87a0b17f960e5c230ab1

6. Для чего нужны аннотации @Produces и @Consumes при реализации REST-сервиса?

Для задания Content-Type заголовка генерируемого сообщения при **@Produces** или задания ожидаемых значений в данном заголовки у приходящих сообщений при **@Consumes**

7. В чем принципиальное различие при реализации обработки ошибок в REST и SOAP-сервисах с точки зрения разработки сервиса, а также с точки зрения реализации клиента?

В сообщениях SOAP-протокола имеется специальный блок soap:Fault, который заполняется в случае, если в процессе обработки запроса произошла ошибка. Клиент, анализируя сообщение, содержащее soap:Fault, может узнать код ошибки, а также ее описание.

JAX-RS, предлагает разработчику достаточно гибкий механизм обработки ошибок. Но этот механизм не отягощен различными дополнительными правилами, и не накладывает никаких ограничений на состав и название методов.

Основная суть метода обработки ошибок в JAX-RS заключается в следующем: разработчик определяет свой обработчик собственного исключения и регистрирует в окружении JAX-RS. После успешной регистрации обработчика, при наступлении исключительной ситуации в коде ресурса достаточно лишь выбросить экземпляр зарегистрированного исключения с помощью ключевого слова throw. Все остальное JAX-RS сделает автоматически.

8. Предположим Вам нужно спроектировать public API для вашей системы. Что вы выберете, REST или SOAP? Какие будут основные критерии для выбора? Приведите ваши рассуждения по данному вопросу.

Критерии:

1. Количество необходимых методов
2. Быстродействие
3. Легковесность
4. Формат ответа
5. Сложность API
6. Быстроизменчивость API

Если наиболее важными критериями являются быстродействие, легковесность и различные форматы ответа, то выбор падет на REST, так как SOAP не такой легковесный как REST, а следовательно не такой быстрый. Также REST позволяет оформить ответ в различных форматах, в то время, как SOAP привязан к XML.

Если наиболее важным критерием является количество необходимых методов, а также предполагается, что API будет сложным и быстроизменяемым, то SOAP будет более полезным. Так как REST предоставляет всего несколько методов, в то время как в SOAP их количество может быть практически неограниченным. И в SOAP работать со сложным и быстроизменяемым API будет проще, так как сервисы SOAP самодокументируются.

9. В чем заключается смысл использования UDDI-реестров в промышленных SOA-системах? Можно ли обойтись без реестров?

В настоящее время, UDDI по большей части используется внутри компаний как платформа для регистрации и динамического нахождения сервисов друг другом. Большая часть возможностей UDDI в таком подходе, однако, не используется.

Да, можно обойтись каким-либо более простым подходом для решения данной задачи. Например, использование доменных имён и локального сервера таких доменных имён, в котором каждый сервис может обратиться к серверу и зарегистрировать себя.

10. Для чего используются такие компоненты SOA, как ESB? Можно ли обойтись без них?

ESB обеспечивает взаимосвязи между различными приложениями по различным протоколам взаимодействия. Т.е. сервисная шина является интеграционным программным обеспечением, и на предприятиях используется для объединения всех работающих систем, сервисов и приложений в единое информационное пространство. ESB в SOA применяется только для объединения слабосвязанных систем. Использование ESB значительно улучшает взаимодействие различных систем, но вполне можно обойтись без ESB при построении SOA.