Repetition Questions (dt. Wiederholungsfragen) Lesson 9

Topics and Concepts (Link to Lecture, via Fact Sheets)

The lesson of the lecture today covered the following concepts:

1. Service Layer (PoEAA) and SOA Principles and Patterns
2. RESTful HTTP

In the corresponding exercise[1], we worked with these concepts.

Questions

Topic/Concept: SOA Principles and Patterns

1. What is the difference between a component and a service according to M. Fowler?
2. Which PoEAA patterns are intended for service design?
3. How was SOA defined in the lecture (hint: take different viewpoints in your answer and mention principles and patterns)?
4. Why is a service contract important to have (hint: think about quality attributes and DDD concepts)?
5. What are the core responsibilities of an Enterprise Service Bus (ESB) according to the SOA patterns UML metamodel form the lecture?

Topic/Concept: REST

1. What is the difference between REST and HTTP?
2. Name at least two of the constraints that define REST as a style.
3. Which REST maturity levels were introduced? What is the main concept on each level? What does HATEOAS stand for?
4. How can the REST principles be implemented? Name at least two APIs and three RFCs.
5. Why are MIME types important in RESTful service design?
6. How can RESTful HTTP and WS-* (WSDL, SOAP) be compared (in terms of decisions to be made)?

Answers

Topic/Concept: SOA Principles and Patterns

1. *Components are a local information hiding/encapsulation concept, whereas services have a remote interface. Local components can implement remote services (and both services and components are usually implemented by multiple collaborating classes in OOP).*
2. *Service Layer, Remote Facade, Data Transfer Object (DTO)*
3. *See the three definitions on lecture slide 11, for instance IT architect: principles such as loose coupling (in four dimensions) and information hiding; patterns such as service contract, ESB, service registry.*
4. *Client and server might work for different firms and have different goals; a service contract can help achieve interoperability and express the Published Language (as defined in DDD).*

[1] ../3-exercises-solutions/ZIO-AppArch-ExerciseWeek9.pdf

5. *Routing, adaptation, transformation according to the UML diagram on lecture slide 13 (the latter responsibility can realize the ACL from DDD)*

Topic/Concept: REST

1. *REST is an architectural style (defined via constraints) and HTTP is a remoting protocol. HTTP also is the foundation for the Web, which in turn is the only full implementation of the REST style (which to some extent was reverse engineered from the architecture of the Web as a distributed document management system). See for instance restcookbook.com[2] in addition to lecture slides.*
2. *See slide 16 of the lecture: Client-server, Stateless, Cacheable, Uniform interface (which includes identification of resources; manipulation of resources through representations; self-descriptive messages; hypermedia as the engine of application state), Layered system, Code on demand (optional)*
3. *Four: POX, URIs, HTTP verbs, Hypermedia a.k.a. Hypertext as the Engine of Application State (HATEOAS)*
4. *HTTP, URIs, media/MIME types; also AtomPub and ASF (see lecture slides for RFC numbers)*
5. *Content negotiation and (custom) media types can support compatibility and bring flexibility. A DDD ACL can translate from one media type to another. Note that not only the "meta meta model" or message exchange format has to be defined (e.g., JSON vs. XML), but also the "meta model", i.e., the data structures to be transmitted (also known as Data Transfer Objects when created/accessed in the service provider/consumer programs).*
6. *Competition (to some extend, if WSDL/SOAP is seen as Web version of RPC as an integration style)*

[2] http://restcookbook.com/Miscellaneous/rest-and-http/