# Repetition Questions (dt. Wiederholungsfragen) Lesson 5

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

The lesson of the lecture today covered the following concepts (see fact sheets in script folder):

- Component dynamics
- Component container patterns and their realization in Spring

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

## Questions

#### Topic/Concept: Component Modeling Steps and Component Dynamics

1. Why/when is it useful to identify candidate components and go through the component modeling steps?
2. Who could be stakeholders of the Collaborators entry in a CRC card? Identify at least two.
3. List three notations and tools that can be used to specify and visualize component behavior (dynamics).
4. Why is it important to specify component behavior (dynamics)? Hint: use at least two quality attributes in your answer.
5. Which arc42 view îs used to document component behavior (dynamics)?
6. Do all CRC cards from component specification always have to be refined and complemented with Component Interaction Diagrams (CIDs)?

#### Topic/Concept: Component Container Patterns and Spring

1. How are two primary patterns called that are implemented by tier 2 component containers?
2. What is the difference between a library and a container framework (in terms of supported patterns)?
3. What is the difference between a managed container and an unmanaged one?
4. How does the Spring framework/container learn about the components and their dependencies?
5. What is the relationship between an analysis-level domain model (and/or candidate components) and classes annotated with `@Component`, `@Controller`, `@Entity` in Spring?
6. List at least two services that implement cross-cutting concerns or shared services (residing on presentation, business logic, or data access layer) in Spring.

## Answers

#### Topic/Concept: Component Responsibilities and ComponentDynamics

1. *For instance, on larger and complex projects and when having to work in new domains; the objective is not to forget anything in design and implementation and to reduce technical risk (e.g., avoid failures such as those presented as project stories in the lecture).*
2. *Architects and other designers creating CIDs; reviewers interested in runtime qualities such as performance, scalability and security (see lecture slide 9).*

[1] ../3-exercises-solutions/ZIO-AppArch-ExerciseWeek5.pdf

3. *UML sequence diagram, UML communication diagram, C4 dynamic diagram; also informal lists and state machines (see arc42 hints).*
4. *Performance, scalability, reliability hot spots can only be detected when looking at control flow (and even data flow) in an application.*
5. *Runtime View*
6. *No, only those for which a positive "build" decision is made, and only those that are particularly significant according to the checklist from week 1.*

**Topic/Concept: Component Container Patterns and Spring**

1. *Inversion of Control, Dependency Injection (Plugin and Service Locator also ok as answers)*
2. *Hollywood principle (Inversion of Control); Plugin also ok as answer*
3. *Managed containers comes as an application server with start/stop commands, etc.; the main routine is inside the framework. An unmanaged container is created by the programmer (who then hands over control to the framework/container). Both of them implement the two container patterns (Inversion of Control, Dependency Injection) and offer shared services such as security and transaction management.*
4. *Annotations implementing dependency injection concepts such as bean assembly and autowiring (in earlier versions, only XML configuration files could be used).*
5. *Refined versions of the candidate components from OOAD models and/or CRC cards that passed the buy vs. build decision are implemented and then placed in the BLL of the application*
6. *JAX-RS support in Spring MVC (HATEOAS), transactions, security, resource pooling, Spring Data JPA (similar to JEE)*