Answers to Questions

Chapter 1

1. Yes, the abstractions provided by the framework are good. However, the way developers use them can easily lead to violations of the principles (separation of concerns, loose coupling, and so on).
2. There is no one number to fit all applications.
3. Not really. In general, the overhead of a few or even a dozen of additional abstraction layers is negligible compared to the overhead of using a framework itself.
4. Increasing the application’s throughput.
5. An abstraction that exposes implementation details.

Chapter 2

1. Active Record objects encapsulate database access (read and write operations), while with Data Mapper, domain objects have no knowledge of the underlying persistence mechanism.
2. Constraints should be used to define data consistency rules (or invariants). Validations are meant to provide meaningful feedback to users.
3. Duck typing is a method of identifying objects in code by checking if they respond to a given method instead of checking their type.
4. When the number of instantiations is large enough to affect the application’s performance.
5. Churn describes how often a given file has been modified.

Chapter 3

1. Serialization is the process of transforming an object to a format that can be moved to a different execution environment where the object can be reconstructed later. Active Job uses serialization to transfer job arguments between Rails executors (for example, a web server and a background jobs processor).
2. Plugins provide additional functionality, while adapters only act as interface translators.
3. A wrapper object combines both interface translation and implementation encapsulation.
4. Localizing the code under refactoring first and subsequently improving its quality in isolation.

Chapter 4

1. Unlike plugins, callbacks don’t have to implement a particular interface.
2. The more callbacks you have in models the slower become tests, especially when using factories.
3. Publishers, subscribers, and a message bus (or broker).
4. Rails concerns support dependency resolution for included modules and come with a convenient DSL to simplify extending the target object’s behavior.
5. Value objects represent simple values and distinguishable by these values. Usually (but not necessary), value objects are immutable.
6. Global objects introduce hidden dependencies between different layers (thus, increase coupling) and make understanding and testing code more complicated.

Chapter 5

1. The main consequences of having too few abstraction layers are having business-logic distributed between layers in unpredictable manner and having over-responsible abstractions.
2. The specification test states that if the tests for the object cover scenarios beyond its primary responsibilities, the corresponding functionality must be extracted into lower abstraction layers.
3. A callable object is any Ruby object that responds to #call. It’s hardly possible to provide a service object definition equally understandable by all Ruby developers. In most cases, we may say that a service object is an object implementing a single business operation.
4. Single responsibility, implementation encapsulation (no leaking), extensibility, testability.
5. Presentation, Application, Domain, and Infrastructure.

Chapter 6

1. A query object is an object responsible for building a query by using domain-level objects as input.
2. Active Record scope is a method, not an object (thus, no state, no isolation forms the model class, and so on).
3. An atomic scope is a scope that introduce an atomic modification to the query being built, that is, containing just a single expression.
4. A repository object may implement multiple methods to perform different queries, while query objects usually correspond to a single query. Also, in Active Record, query objects may accept Active Record relations as input and return relations as well to be chainable.

Chapter 7

1. Because form objects deal with user input from the Presentation Layer.
2. Input values typecasting and validation, performing side actions, parameters filtering.
3. Form objects may or may not be backed by a model.
4. Filter objects are responsible for filtering data, while form objects trigger business operations.
5. Filter objects use user-provided data as input while query objects use domain objects.

Chapter 8

1. Pros: we extract presentation-related logic from models.

Cons: helpers are just methods and couldn’t be turned into a proper abstraction (to increase maintainability).

1. An open presenter allows access to the underlying object methods not explicitly overridden by the presenter (that is, allows passing-through). A closed presenter’s interface is limited to what is defined on the presenter class.
2. Technically, they can. But you should better consider proper abstractions for HTML generations, such as view components.
3. The leaking decorator problem is the situation when a decorator is passed as an input to an object from a lower abstraction level and not expecting Presentation-level objects.
4. Serializer objects can be seen as specialized presenters: they prepare objects to be encoded into a wire format (for example, JSON).
5. Serializers describes how to represent an object under serialization. Technically, serializers usually responsible for the serialization itself (though it’s done at the lower, library level).

Chapter 9

1. Authentication answers the Who’s there? question, while authorization answers the Am I allowed to … ? question.
2. High risk of role explosion.
3. Authorization enforcement is an act of performing authorization. It must happen as soon as we know which action is to be performed and who wants to perform this action. Typically, it happens within the Presentation layer.
4. A policy object encapsulates a business rule or a set of rules describing which operation can be performed within a given context.
5. A view policy object is a specific kind of policy objects that determines the visibility of UI elements based on the current user’s permissions.
6. Common techniques are authorization preloading and caching.
7. Scoping-based authorization implies using only authorized scopes to verify access, that is, perform access checks at the same time as loading a resource. Pros: no data is loaded into memory if no access is granted (so, may be considered more secure).

Cons: potential performance overhead of performing complex queries frequently.

Chapter 10

1. Mailers belong to the Application layer since they don’t belong to either the Presentation or the Domain layer.
2. The Business layer contains objects representing business-logic rules and operations (usually, in a framework-agnostic way). The Services layer contains abstractions closer to implementation, usually external systems, but still representing application-level operations.
3. It’s an abstraction layer responsible for orchestrating user notifications.
4. A delivery object acts as a gateway to notification channels (backed by notifier objects) and follows the Action Mailer convention of having a delivery class per resource. A notification object encapsulates delivery mechanisms for a single notification.
5. Keeping notification settings in the User model increases the class responsibility (and make it one step closer to become a God object).

Chapter 11

1. Layouts, action templates, and partials.
2. Lack of interface (signatures) and state sharing with controllers. We can use strict locals and static analysis tools (linters, for example, erb\_lint) to build some confidence when using partials and templates.
3. Design system is a collection of reusable elements and guidelines for building a user interface.
4. View components are isolated and self-contained.
5. Helpers are just methods. Partials are markup templates. Components are Ruby objects (backed by HTML templates).

Chapter 12

1. In Rails 7.1, we can store configuration using YAML files and encrypted credentials. In addition, we can hard-code values in Ruby files and use environment variables.
2. Pros: simplicity, twelve-factor-ness.

Cons: flat structure, lack of organization, potential bloat (the ENV hell).

1. Secrets and settings.
2. Configuration sources belong to the infrastructure layer. Referring to configuration sources directly from upper architecture layers violates the layered architecture principles (and such violation always lead to decreased maintainability).
3. A configuration object is an object providing configuration information and encapsulating underlying configuration providers or data sources.

Chapter 13

1. The infrastructure layer spans the whole application, it stands outside of the layers stack.
2. Database adapters, background jobs adapters, mailing services, storage services, and so on
3. The abstraction distance between two objects is the number of intermediate abstractions between them.
4. Logging provides visibility to the events happening within the application.
5. Reporting exceptions must happen close to real time, while logs are usually used in retrospective analysis.
6. Instrumentation implies collecting and exposing vital characteristics of the application under consideration.
7. Performance is the main reason and motivation to extract low-level functionality from Rails applications to standalone services.