# FullStack.Cafe - Kill Your Tech Interview

## Q1: What Is Load Balancing? ☆

**Topics:** Software Architecture Load Balancing

### Answer:

**Load balancing** is simple technique for distributing workloads across multiple machines or clusters. The most common and simple load balancing algorithm is Round Robin. In this type of load balancing the request is divided in circular order ensuring all machines get equal number of requests and no single machine is overloaded or underloaded.

**The Purpose of load balancing is to**

- Optimize resource usage (avoid overload and under-load of any machines)
- Achieve Maximum Throughput
- Minimize response time

**Most common load balancing techniques in web based applications are**

1. Round robin
2. Session affinity or sticky session
3. IP Address affinity

## Q2: What is Entity Framework? ☆

**Topics:** Entity Framework

### Answer:

ADO.NET EF is an ORM (object-relational mapping) which creates a higher abstract object model over ADO.NET components. So rather than getting into dataset, datatables, command, and connection objects as shown in the below code, you work on higher level domain objects like customers, suppliers, etc.

## Q3: What is API Design? ☆

**Topics:** API Design

### Answer:

**API design** is the process of building an intermediary interface for a system to system connection to expose data to application users and developers.

The fundamental API design influences how well users can consume it and the general user experience. This development process does not allow a single approach. Instead, it combines a series of guidelines to meet initial expectations and continue to work consistently. Application programming interface designers closely follow industry best practices, design patterns, API design principles, and user needs to develop software that presents excellent functionality.

## Q4: What are different types of Web Services? ☆☆

**Topics:** API Design

### Answer:

There are two types of web services:

- **SOAP Web Services**: Runs on SOAP protocol and uses XML technology for sending data.
- **Restful Web Services**: It's an architectural style and runs on HTTP/HTTPS protocol almost all the time. REST is a stateless client-server architecture where web services are resources and can be identified by their URIs. Client applications can use HTTP GET/POST methods to invoke Restful web services.

## Q5: What is SOAP? ☆☆

**Topics:** SOA API Design

### Answer:

*SOAP* stands for Simple Object Access Protocol. SOAP is an XML based industry standard protocol for designing and developing web services. Since it's XML based, it's platform and language independent. So our server can be based on JAVA and client can be on .NET, PHP etc. and vice versa.

## Q6: What are advantages of REST web services? ☆☆

**Topics:** API Design REST & RESTful

### Answer:

Some of the advantages of REST web services are:

- Learning curve is easy since it works on HTTP protocol
- Supports multiple technologies for data transfer such as text, xml, json, image etc.
- No contract defined between server and client, so loosely coupled implementation.
- REST is a lightweight protocol
- REST methods can be tested easily over browser.

## Q7: Mention whether you can use GET request instead of PUT to create a resource? ☆☆

**Topics:** API Design

### Answer:

No, you are not supposed to use POST or GET. GET operations should only have view rights

## Q8: Mention what are resources in a REST architecture? ☆☆

**Topics:** API Design REST & RESTful

### Answer:

Resources are identified by logical URLs; it is the key element of a RESTful design. Unlike, SOAP web services in REST, you view the product data as a resource and this resource should contain all the required information.

## Q9: Mention some key characteristics of REST? ☆☆

**Topics:** API Design REST & RESTful

### Answer:

Some key characteristics of REST includes

- REST is stateless, therefore the SERVER has no state (or session data)
- With a well-applied REST API, the server could be restarted between two calls as every data is passed to the server
- Web service mostly uses POST method to make operations, whereas REST uses GET to access resources

## Q10: What are the core components of a HTTP Request? ☆☆

**Topics:** API Design

### Answer:

A HTTP Request has five major parts −

- **Verb** − Indicate HTTP methods such as GET, POST, DELETE, PUT etc.
- **URI** − Uniform Resource Identifier (URI) to identify the resource on server.
- **HTTP Version** − Indicate HTTP version, for example HTTP v1.1 .
- **Request Header** − Contains metadata for the HTTP Request message as key-value pairs. For example, client ( or browser) type, format supported by client, format of message body, cache settings etc.
- **Request Body** − Message content or Resource representation.

## Q11: What is cached response? ☆☆

**Topics:** API Design

### Answer:

*Caching* refers to storing server response in client itself so that a client needs not to make server request for same resource again and again. A server response should have information about how a caching is to be done so that a client caches response for a period of time or never caches the server response.

## Q12: Define what is SOA ☆☆

**Topics:** SOA API Design

### Answer:

*A Service Oriented Architecture (SOA)* is basically defined as an architectural pattern consisting of services. Here application components provide services to the other components using communication protocol over the network. This communication involves data exchanging or some coordination activity between services.

Some of the key principles on which SOA is based are mentioned below

- The service contract should be standardized containing all the description of the services.
- There is loose coupling defining the less dependency between the web services and the client.
- It should follow Service Abstraction rule, which says the service should not expose the way functionality has been executed to the client application.
- Services should be reusable in order to work with various application types.
- Services should be stateless having the feature of discoverability.
- Services break big problems into little problems and allow diverse subscribers to use the services.

## Q13: What are the Advantages of Using ASP.NET Web API? ☆☆

**Topics:** ASP.NET Web API

### Answer:

Using ASP.NET Web API has a number of advantages, but core of the advantages are:

- It works the HTTP way using standard HTTP verbs like `GET`, `POST`, `PUT`, `DELETE`, etc. for all CRUD operations
- Complete support for routing
- Response generated in JSON or XML format using `MediaTypeFormatter`
- It has the ability to be hosted in IIS as well as self-host outside of IIS
- Supports Model binding and Validation
- Support for OData

## Q14: Which *status code* used for all *uncaught exceptions* by default? ☆☆

**Topics:** ASP.NET Web API

### Answer:

By default, most exceptions are translated into an HTTP response with status code **500**, **Internal Server Error.**

Consider:

```
[Route("CheckId/{id}")]
[HttpGet]
public IHttpActionResult CheckId(int id)
{
    if(id > 100)
    {
        throw new ArgumentOutOfRangeException();
    }
    return Ok(id);
}
```

For more control over the response, you can also construct the entire response message and include it with the `HttpResponseException`:

```
public Product GetProduct(int id)
{
    Product item = repository.Get(id);
    if (item == null)
    {
        var resp = new HttpResponseMessage(HttpStatusCode.NotFound)
        {
            Content = new StringContent(string.Format("No product with ID = {0}", id)),
            ReasonPhrase = "Product ID Not Found"
        };
        throw new HttpResponseException(resp);
    }
    return item;
}
```

## Q15: What exactly is OAuth (Open Authorization)? ☆☆

**Topics:** ASP.NET Web API

### Answer:

**OAuth** (Open Authorization) is an open standard for access granting/deligation protocol. It used as a way for Internet users to grant websites or applications access to their information on other websites but without giving them the passwords. It does not deal with authentication.

Basically there are three parties involved: oAuth Provider, oAuth Client and Owner.

- oAuth Client (Application Which wants to access your credential)
- oAuth Provider (eg. facebook, twitter...)
- Owner (the person with facebook,twitter.. account )

## Q16: Define Microservice Architecture ☆☆

**Topics:** Microservices Software Architecture

### Answer:

**Microservices**, aka ***Microservice Architecture***, is an architectural style that structures an application as a collection of small autonomous services, modeled around a **business domain.**

## Q17: Why use WebSocket over HTTP? ☆☆

**Topics:** WebSockets Software Architecture

### Answer:

A **WebSocket** is a continuous connection between client and server. That continuous connection allows the following:

1. Data can be sent from server to client at any time, without the client even requesting it. This is often called server-push and is very valuable for applications where the client needs to know fairly quickly when something happens on the server (like a new chat messages has been received or a new price has been udpated). A client cannot be pushed data over http. The client would have to regularly poll by making an http request every few seconds in order to get timely new data. Client polling is not efficient.

2. Data can be sent either way very efficiently. Because the connection is already established and a webSocket data frame is very efficiently organized, one can send data a lot more efficiently that via an HTTP request that necessarily contains headers, cookies, etc...

## Q18: What is Domain Driven Design? ☆☆

**Topics:** Software Architecture DDD

### Answer:

**Domain Driven Design** is a methodology and process prescription for the development of complex systems whose focus is mapping activities, tasks, events, and data within a problem domain into the technology artifacts of a solution domain.

It is all about trying to make your software a model of a real-world system or process.

## Q19: What is *Dependency Injection*? ☆☆

**Topics:** Design Patterns Dependency Injection

### Answer:

**Dependency injection** makes it easy to create loosely coupled components, which typically means that components consume functionality defined by interfaces without having any first-hand knowledge of which implementation classes are being used.

*Dependency injection* makes it easier to change the behaviour of an application by changing the components that implement the interfaces that define application features. It also results in components that are easier to isolate for unit testing.

## Q20: What do you understand by *Clean Architecture* approach? ☆☆

**Topics:** Clean Architecture Software Architecture

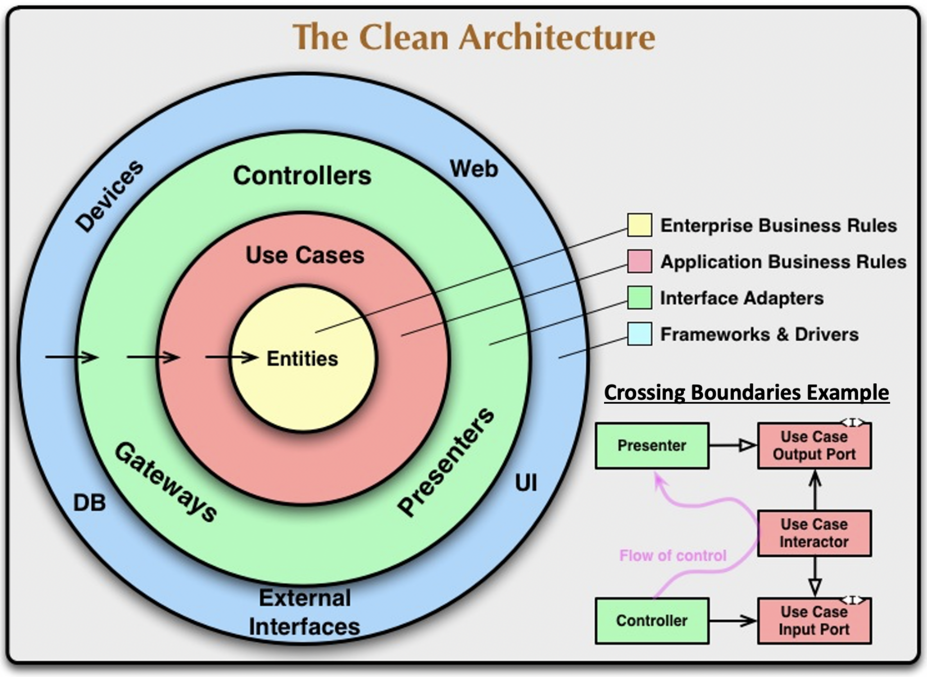
### Answer:

The majority of architectural approaches (like Onion or Hexagonal Architecture) have this common set of characteristics:

The architectural approach shall be:

- **Independent of any tech frameworks**
- **Testable**: business rules are testable without the UI, the DB, the web server, and any other external element.
- **Independent of the UI:** The UI can change easily with no impact on the business rules. The UI can easily be replaced.
- **Independent of the database**
- **Independent of external agents:** The business rules don't know anything about interfaces to the outside world.

**The Clean Architecture** is an attempt at integrating all these architectures into a single actionable idea:


The Clean Architecture
Devices
Controllers
Web
Use Cases
Entities
Gateways
Presenters
DB
UI
External Interfaces
Enterprise Business Rules
Application Business Rules
Interface Adapters
Frameworks & Drivers
Crossing Boundaries Example
Presenter
Use Case Output Port
Use Case Interactor
Flow of control
Controller
Use Case Input Port
<I>
<I>