3. General

1. Define using your own words the next concepts in 2 lines:**Q.**You have a live server behaving strange, you don’t know what is happening but the requests are not responding. You see the number of apache threads increase until the server crashes. How would you start investigating this issue?Test plan
2. the server may have been attacked by hackers with DDoS (Distribuited Denial of Service). I would verify the server logs to search the IPs sources that have requested to the website in the last 48 hours. If the IPs source belong to a TOR network, it may not be possible to identify the hosts that requested the website. I would also verify the web server connection and if the mpm\_worker\_module is active on apache.

**Link:**

<http://www.genericarticles.com/mediawiki/index.php?title=How_to_optimize_apache_web_server_for_maximum_concurrent_connections_or_increase_max_clients_in_apache>;

1. **Q.**Difference between Unit Test, Integration test and Acceptance test. Test case

**A. Unit Test**: Specify and test one point of the contract of single method of a class. This should have a very narrow and well defined scope. Complex dependencies and interactions to the outside world are stubbed or mocked.

**Integration test**: Test the correct inter-operation of multiple subsystems. There is a whole spectrum there, from testing integration between two classes, to testing integration with the production environment.

**Acceptance test**: Test that a feature or use case is correctly implemented. It is similar to an integration test, but focusing on the use case to provide rather than on the components involved.

**Links**: <http://www.typemock.com/unit-tests-integration-tests>; <http://www.testfeed.co.uk/integration-vs-acceptance-tests/>;

1. **Q.**What is a mock and how would you create them in PHP.
2. Mocking is primarily used in unit testing. An object being tested may have dependencies on other (complex) objects. To isolate the behavior of the object to be tested it is necessary to replace the other objects by mocks that simulate the behavior of the real objects. This is useful if the real objects are not able to be incorporated into the unit test. Summarizing, mocking is creating objects that simulate the behavior of real objects. In PHP language, the getMock ($className) method provided by PHPUnit can be used in a test to automatically generate an object that can act as a double test for the specified original class. This double test object can be used in every context in which an original class object is expected.
3. **Q.**You need to design a storage architecture for php sessions that provides high scalability, failover, and high performance. What would you do?
4. Create a Software Load Balacer with sticky Sessions and Central Session Store, which is more customizable and with HTTP Servers load balancing is typically combined with http accelerators or reverse proxy. Use a Horizontal Scaling (for App Server) and Vertical Partitioning with SAN (for Hardware Storage). Choose a DB which natively supports Master-Slave replication, use Master-Slave Async replication and write DAO layer to ensure. Use Horizontal and Vertical Patitioning of DB.

**Link:**

<http://www.slideshare.net/directi/building-a-scalable-architecture-for-web-apps>

1. **Q.**JSON vs XML. When / Why would you use instead of the other?
2. There is not a vast difference between JSON and XML in terms of simplicity, openness, and interoperability. Both of them use Unicode standards, what make these formats very easy to distribute to a wide range of users. With JSON, you are only limited to store classical data like text and numbers. However, XML allows you to store any data type you wish. Classical data does not require this extensibility and can benefit from the simplicity of JSON. The best tool for sharing traditional data is JASON because the data is stored in arrays and records, while XML stores data in trees. When sharing documents is necessary, then XML is the right tool to be used because JASON only offers options to transfer data without formatting.

**Link:**

<http://www.udemy.com/blog/json-vs-xml/>

1. **Q.**If you are using mysql command “explain”, what do you look at to know if you must optimize a query
2. With the help of [EXPLAIN](http://dev.mysql.com/doc/refman/5.0/en/explain.html), you can see where you should add indexes to tables so that the statement executes faster by using indexes to find rows. You can also use [EXPLAIN](http://dev.mysql.com/doc/refman/5.0/en/explain.html) to check whether the optimizer joins the tables in an optimal order. [EXPLAIN](http://dev.mysql.com/doc/refman/5.0/en/explain.html) can also be used to obtain information about the columns in a table.

**Link:**

<http://dev.mysql.com/doc/refman/5.0/en/using-explain.html>

1. **Q.**What advantages provide the usage of traits?
2. Traits are a mechanism for code reuse in single inheritance languages such as PHP. A Trait is intended to reduce some limitations of single inheritance by enabling a developer to reuse sets of methods freely in several independent classes living in different class hierarchies. The semantics of the combination of Traits and classes is defined in a way which reduces complexity, and avoids the typical problems associated with multiple inheritance and Mixins. It is an addition to traditional inheritance and enables horizontal composition of behavior; that is, the application of class members without requiring inheritance.
3. **Q.**What is the benefit of using NameSpaces?
4. The NameSpaces allow ambiguous names / classes with same name to co-exists while being in two different NameSpaces. For example, Table class can be referring to a table in a persistent database and a HTML table. I can put namespaces to specifically use the exact table that I want, i.e. \Model\Table and \View\Table respectively.
5. **Q.**RESTful interface vs SOAP. Benefits of each option

**A.**

**REST**

* The RESTful Web services are completely stateless. This can be tested by restarting the server and checking if the interactions are able to survive.
* Restful services provide a good caching infrastructure over HTTP GET method (for most servers). This can improve the performance, if the data the Web service returns is not altered frequently and not dynamic in nature.
* The service producer and service consumer need to have a common understanding of the context as well as the content being passed along as there is no standard set of rules to describe the REST Web services interface.
* REST is particularly useful for restricted-profile [devices such as mobile](http://searchsoa.techtarget.com/answer/REST-or-SOAP-Which-offers-the-most-benefits-for-mobile-applications) and PDAs for which the overhead of additional parameters like headers and other SOAP elements are less.
* REST services are easy to integrate with the existing websites and are exposed with XML so the HTML pages can consume the same with ease. There is hardly any need to refactor the existing website architecture. This makes developers more productive and comfortable as they will not have to rewrite everything from scratch and just need to add on the existing functionality.
* REST-based implementation is simple compared to SOAP.

**SOAP**

* The Web Services Description Language (WSDL) contains and describes the common set of rules to define the messages, bindings, operations and location of the Web service. WSDL is a sort of formal contract to define the interface that the Web service offers.
* SOAP requires less plumbing code than REST services design, (i.e., transactions, security, coordination, addressing, trust, etc.) Most real-world applications are not simple and support complex operations, which require conversational state and contextual information to be maintained. With the SOAP approach, developers need not worry about writing this plumbing code into the application layer themselves.
* SOAP Web services (such as JAX-WS) are useful in handling asynchronous processing and invocation.
* SOAP supports several protocols and technologies, including WSDL, XSDs, SOAP, WS-Addressing

In a nutshell, when you're publishing a complex application program interface (API) to the outside world, SOAP will be more useful. But when something with a lower learning curve, and with lightweight and faster results and simple transactions (i.e., CRUD operations) is needed, my vote goes to REST.

1. **Q.**What is dependency infection about? What are some of the key benefits of using it?

**A.** Probably there is an error in the question (injection instead of infection). Dependency injection is a software design pattern that implements inversion of control and allows a program design to follow the dependency inversion principle and resolves conflict of dependence class using an Injection Costructor. The term was coined by Martin Fowler. The Dependency Injection component (in Symfony for example) allows you to standardize and centralize the way objects are constructed in your application. You need to use Dependency Injection for long-term projects. The developers normally don't plan their project, they just want to write code quickly to "get things done” and the result is that is impossible run Unit Test. Dependency Injection solves the problem.

**Advantages:**

* As dependency injection does not require any change in code behavior, it can be applied to legacy code as a refactoring. The result is more independent clients that are easier to unit test in isolation using stubs or mock objects that simulate other objects not under test. This ease of testing is often the first benefit noticed when using dependency injection.
* Dependency injection allows a client to remove all knowledge of a concrete implementation that it needs to use. This helps isolate the client from the impact of design changes and defects. It promotes reusability, testability and maintainability.
* Dependency injection can be used to externalize a system's configuration details into configuration files allowing the system to be reconfigured without recompilation. Separate configurations can be written for different situations that require different implementations of components. This includes, but is not limited to, testing.
* Reduction of boilerplate code in the application objects since all work to initialize or set up dependencies is handled by a provider component.
* Dependency injection allows concurrent or independent development. Two developers can independently develop classes that use each other, while only needing to know the interface, the classes will communicate through. Plugins are often developed by third party shops that never even talk to the developers who created the product that uses the plugins.

**Disadvantages:**

* Dependency injection can make codes difficult to trace (read) because it separates behavior from construction. This means developers must refer to more files to follow how a system performs.
* Dependency injection typically requires more lines of code to accomplish the same behavior legacy code would.
* Dependency injection diminishes [encapsulation](http://en.wikipedia.org/wiki/Encapsulation_%28object-oriented_programming%29) by requiring users of a system to know how it works and not merely what it does.
* Dependency injection increases [coupling](http://en.wikipedia.org/wiki/Coupling_%28computer_programming%29) by requiring the user of a subsystem to provide for the needs of that subsystem.