Please explain what the command below does.

# java -Xmx512m -cp my-app.jar ch.solenix.myapp.Program Hello World

java -Xmx512m command is setting maximum size, in bytes, of the memory allocation pool.

Append the letter M to indicate megabytes. And –cp command means class search path of my-app.jar and we are passing parameter Hello world to ch.solenix.myapp.Program.

# Design for Restful Api

# Accept and respond with JSON

REST APIs should accept JSON for request payload and also send responses to JSON. JSON is the standard for transferring data

Form data is good for sending data, especially if we want to send files. But for text and numbers, we don’t need form data to transfer those since—with most frameworks—we can transfer JSON by just getting the data from it directly on the client side. It’s by far the most straightforward to do so.

# Proper HTTP method for action:-

The action should be indicated by the HTTP request method that we’re making. The most common methods include GET, POST, PUT, and DELETE.

* GET retrieves resources.
* POST submits new data to the server.
* PUT updates existing data.
* DELETE removes data.

# Error Handling:-

When an error occurs, we should handle errors gracefully and return HTTP response codes that indicate what kind of error occurred

Common error HTTP status codes include:

* 400 Bad Request – This means that client-side input fails validation.
* 401 Unauthorized – This means the user isn’t not authorized to access a resource. It usually returns when the user isn’t authenticated.
* 403 Forbidden – This means the user is authenticated, but it’s not allowed to access a resource.
* 404 Not Found – This indicates that a resource is not found.
* 500 Internal server error – This is a generic server error. It probably shouldn’t be thrown explicitly.
* 502 [Bad Gateway](https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/502) – This indicates an invalid response from an upstream server.
* 503 Service Unavailable – This indicates that something unexpected happened on server side (It can be anything like server overload, some parts of the system failed, etc.).

# Allow filtering, sorting, and pagination:-

## Databases behind a REST API can get very large. Sometimes, there’s so much data that it shouldn’t be returned all at once because it’s way too slow or will bring down our systems. Therefore, we need ways to filter items.

## Filtering and pagination both increase performance by reducing the usage of server resources. As more data accumulates in the database, the more important these features become.

## Maintain good security practices:-

## Most communication between client and server should be private since we often send and receive private information. Therefore, using SSL/TLS for security is a must.

## Cache data to improve performance:-

We can add caching to return data from the local memory cache instead of querying the database to get the data every time we want to retrieve some data that users request. The good thing about caching is that users can get data faster.

Eg:- caching solutions like [Redis](https://redis.io/), in-memory caching

## Versioning our APIs

We should have different versions of API if we’re making any changes to them that may break clients. The versioning can be done according to semantic version (for example, 2.0.6 to indicate major version 2 and the sixth patch) like most apps do nowadays.

This way, we can gradually phase out old endpoints instead of forcing everyone to move to the new API at the same time. The v1 endpoint can stay active for people who don’t want to change, while the v2, with its shiny new features, can serve those who are ready to upgrade. This is especially important if our API is public. We should version them so that we won’t break third party apps that use our APIs.

Versioning is usually done with /v1/, /v2/, etc. added at the start of the API path.