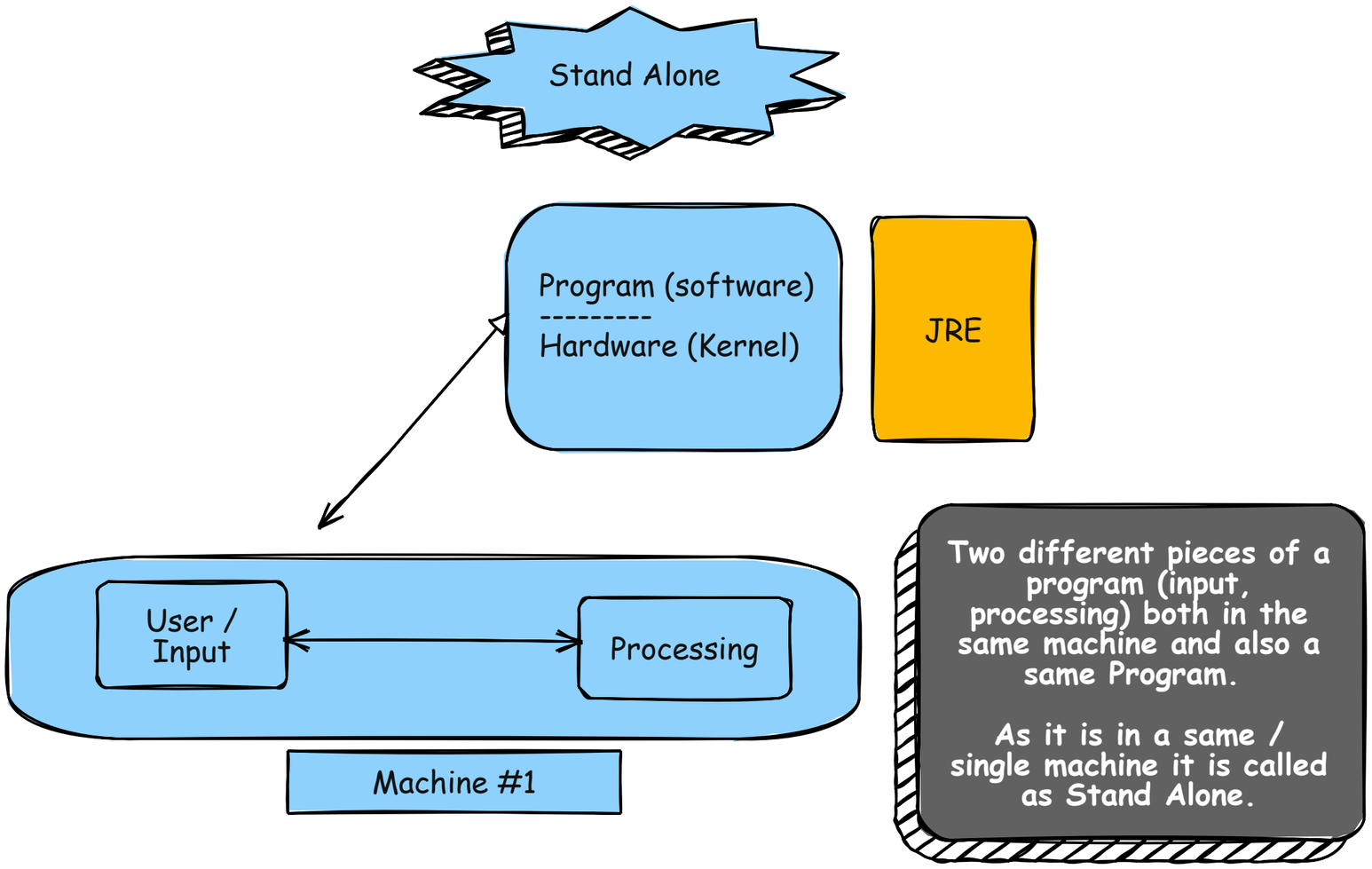
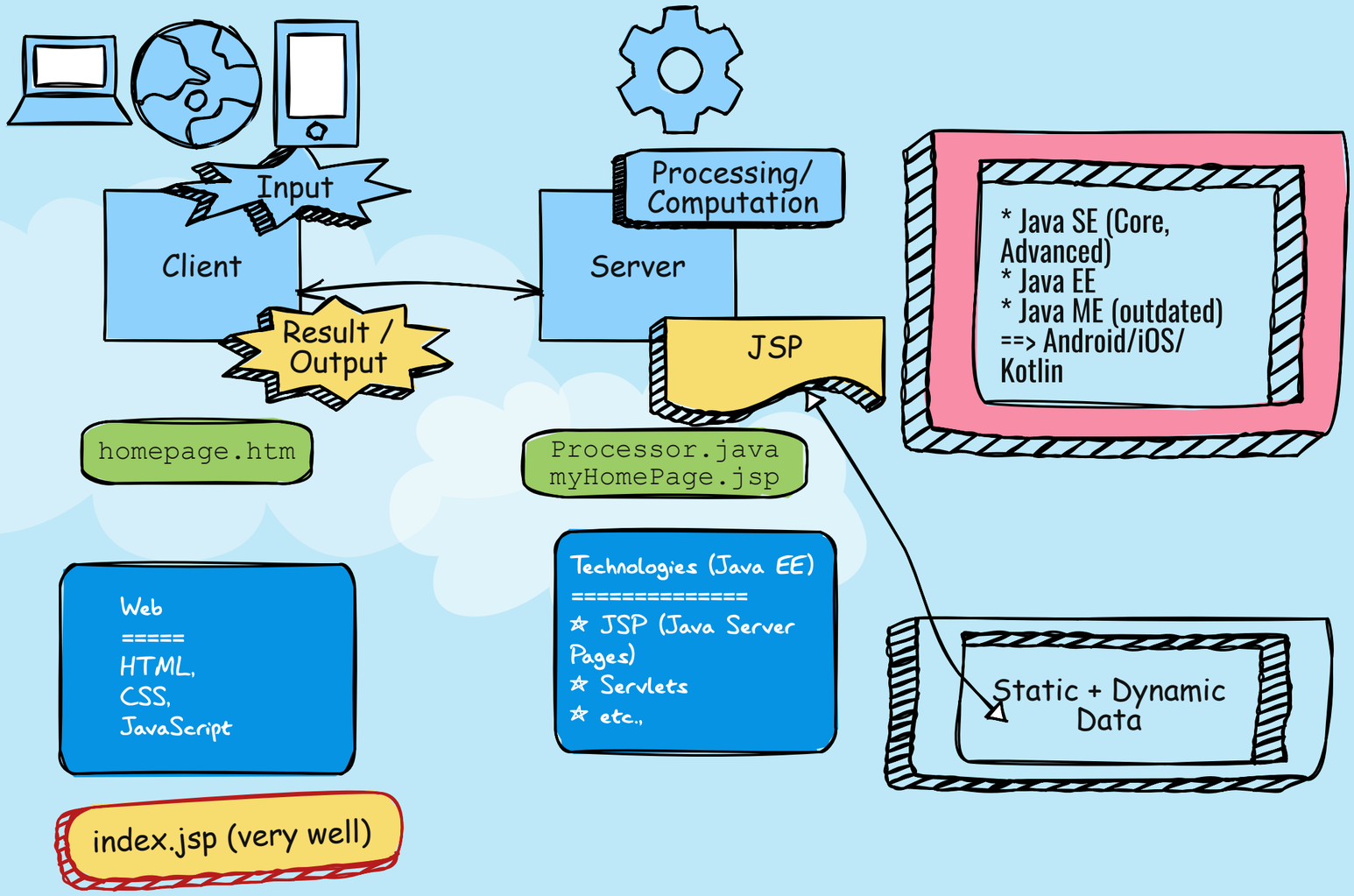
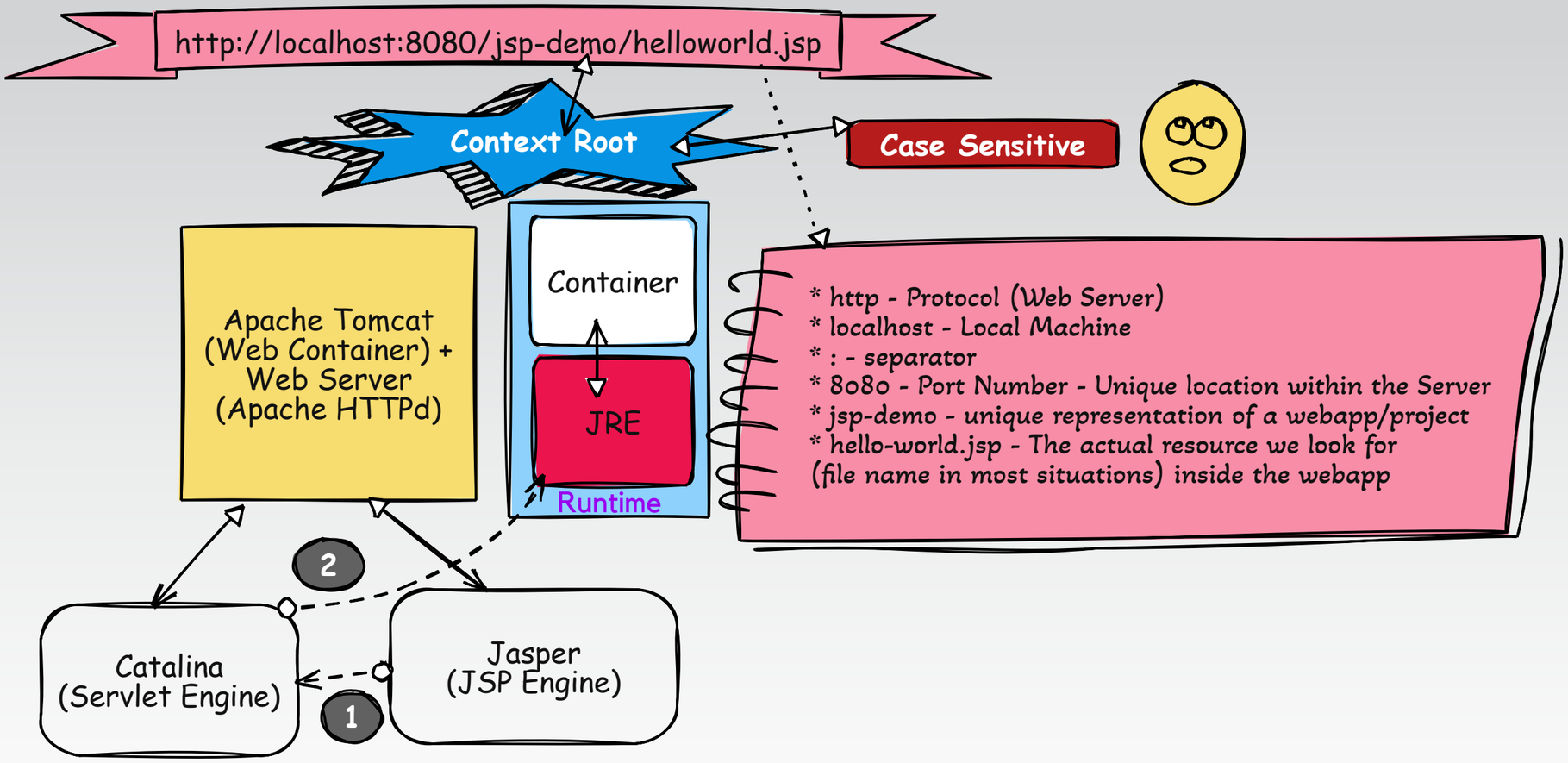
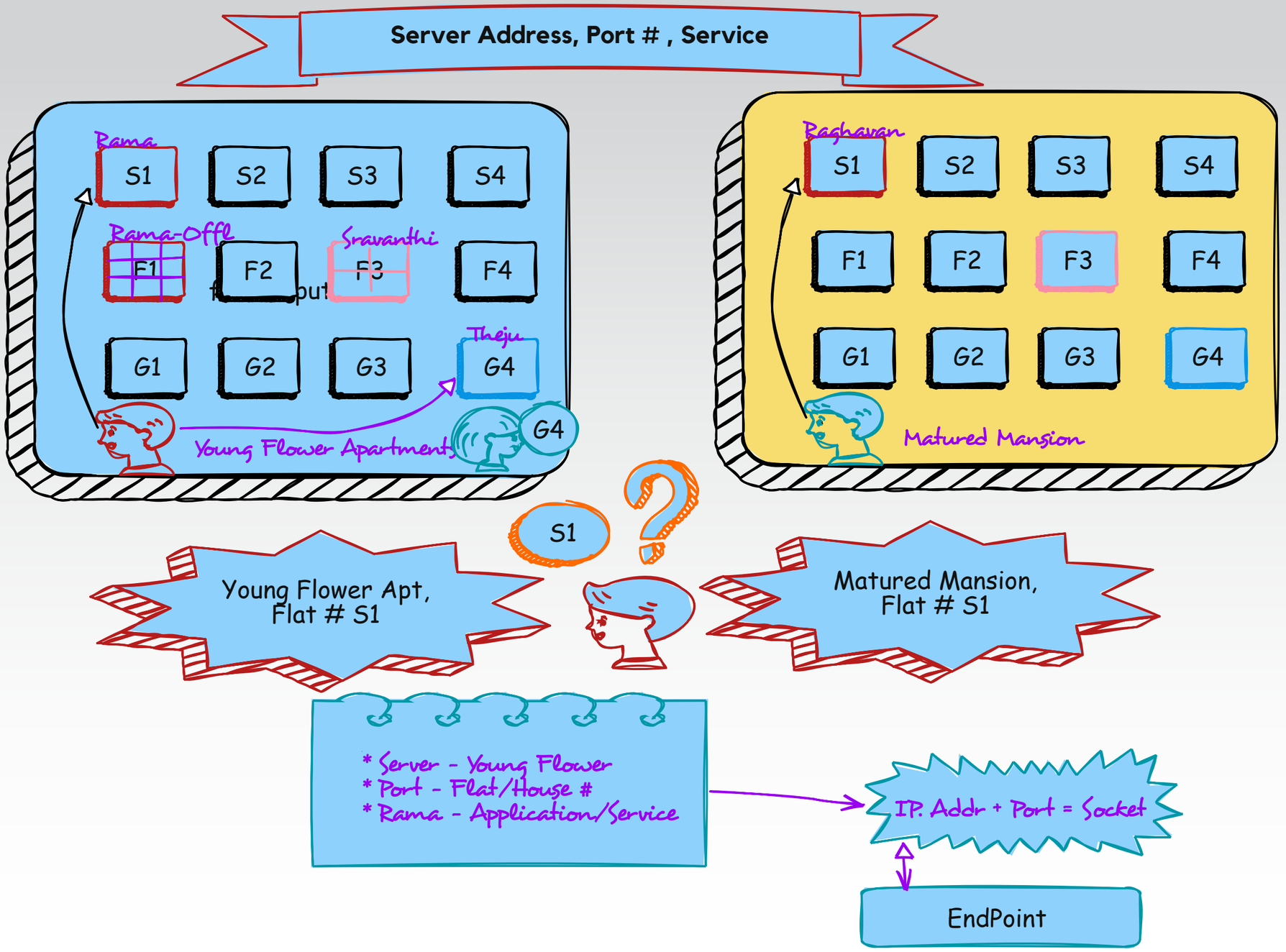

Stand Alone
Program (software)

Hardware (Kernel)
JRE
User / Input
Processing
Machine #1
Two different pieces of a program (input, processing) both in the same machine and also a same Program.
As it is in a same / single machine it is called as Stand Alone.


Input
Client
Result / Output
Server
Processing/ Computation
JSP
* Java SE (Core, Advanced)
* Java EE
* Java ME (outdated)
==> Android/iOS/ Kotlin
homepage.htm
Processor.java
myHomePage.jsp
Technologies (Java EE)
==========
☆ JSP (Java Server Pages)
☆ Servlets
☆ etc.,
Web
=====
HTML,
CSS,
JavaScript
Static + Dynamic Data
index.jsp (very well)

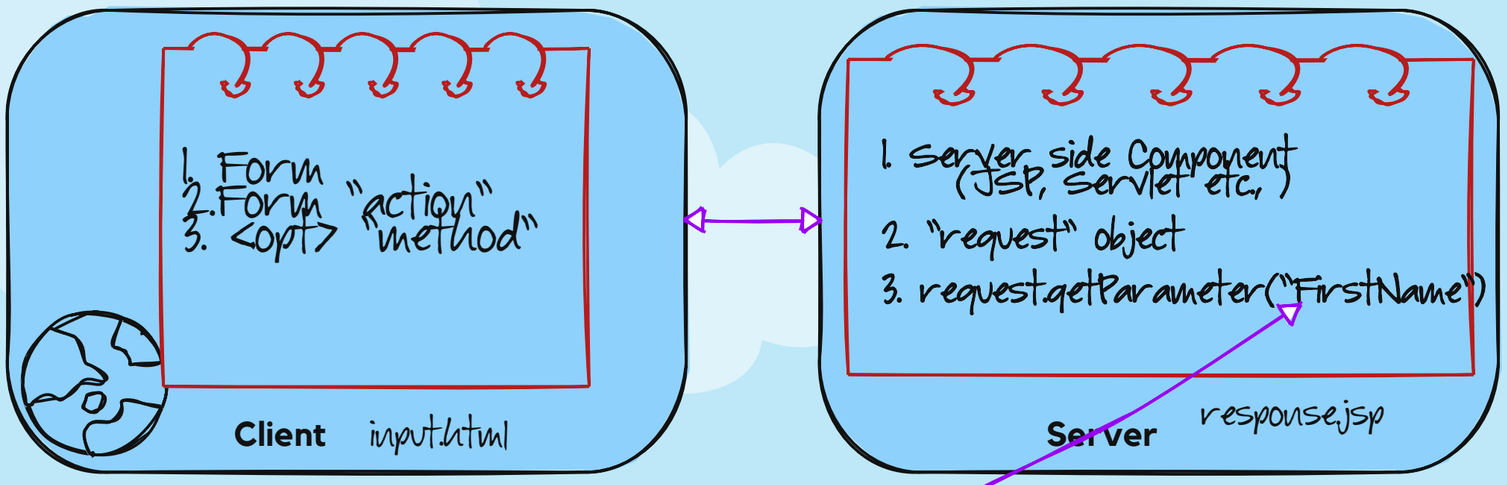

http://localhost:8080/jsp-demo/helloworld.jsp
Context Root
Case Sensitive
Apache Tomcat
(Web Container) +
Web Server
(Apache HTTPd)
Container
JRE
Runtime
* http - Protocol (Web Server)
* localhost - Local Machine
* : - separator
* 8080 - Port Number - Unique location within the Server
* jsp-demo - unique representation of a webapp/project
* hello-world.jsp - The actual resource we look for
(file name in most situations) inside the webapp
2
Catalina
(Servlet Engine)
Jasper
(JSP Engine)
1



Two different applications "jsp-demo", and "jsp-tutorials" are hosted on the Server "localhost" and exposed at the port number "8080".

The resources being looked at each application differ. In jsp-demo, it is "helloworld.jsp", and in. "jsp-tutorials", it is "index.jsp"

Form Submission

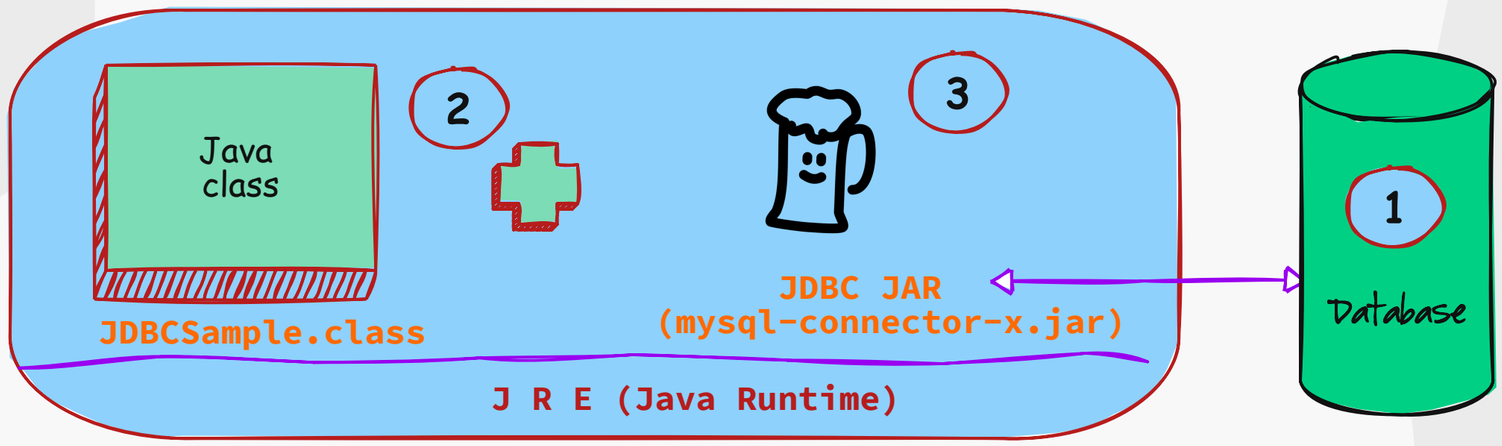



```
<form action="response.jsp"
    method="post">
  <input name="FirstName"../>
</form>
```

input.html

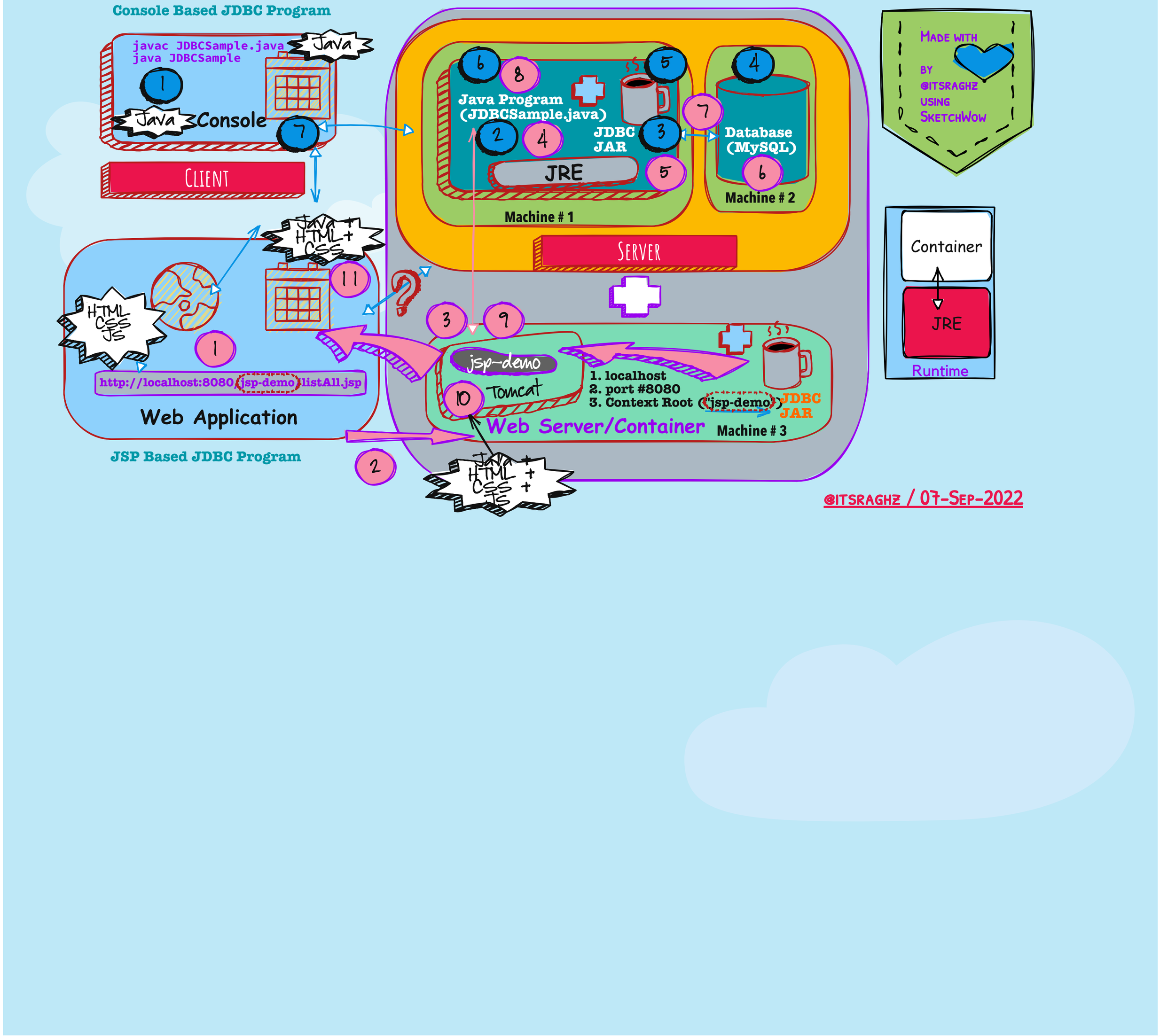
http://localhost:8080/jsp-demo/response.jsp

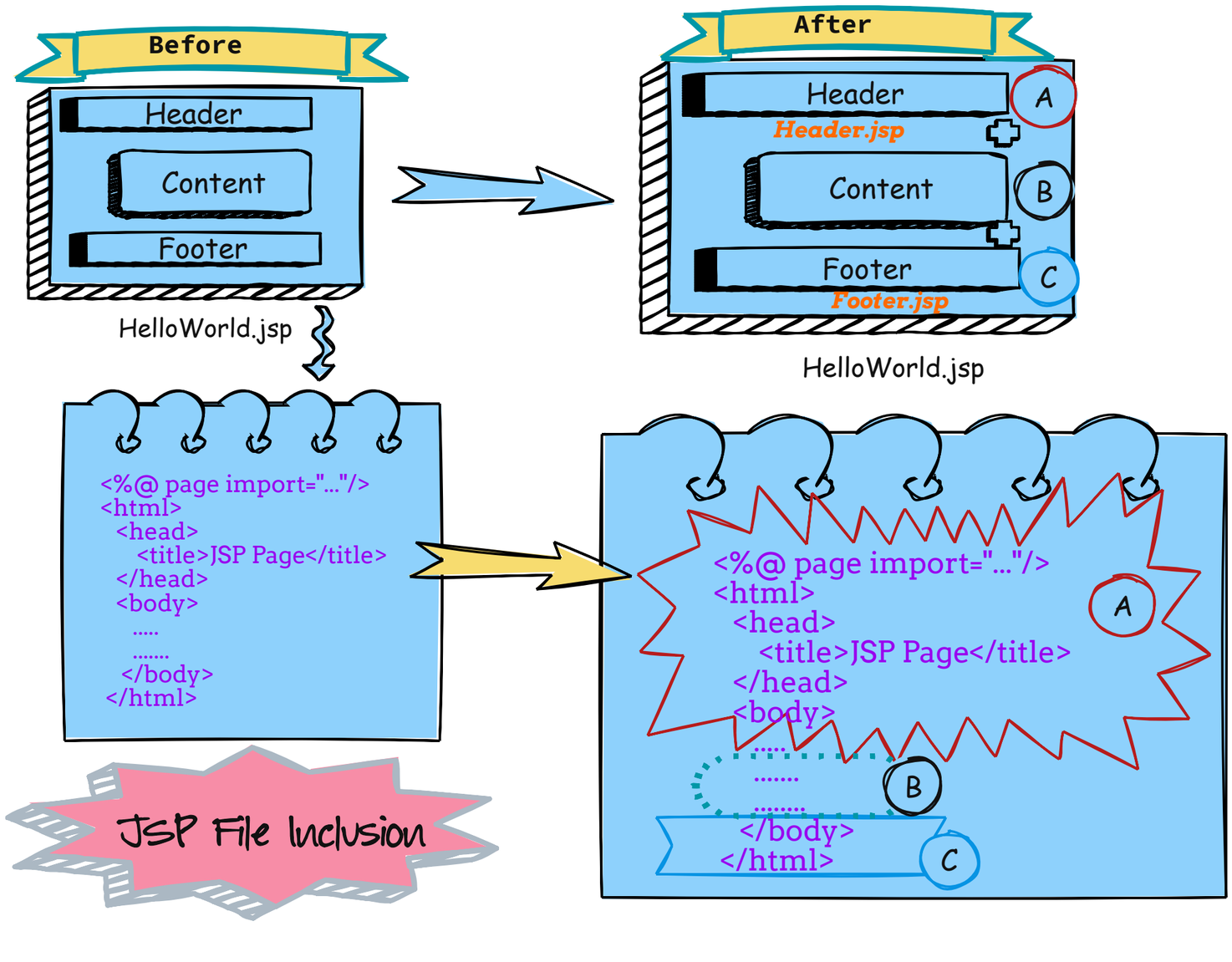
http://localhost:8080/jsp-demo/input.html

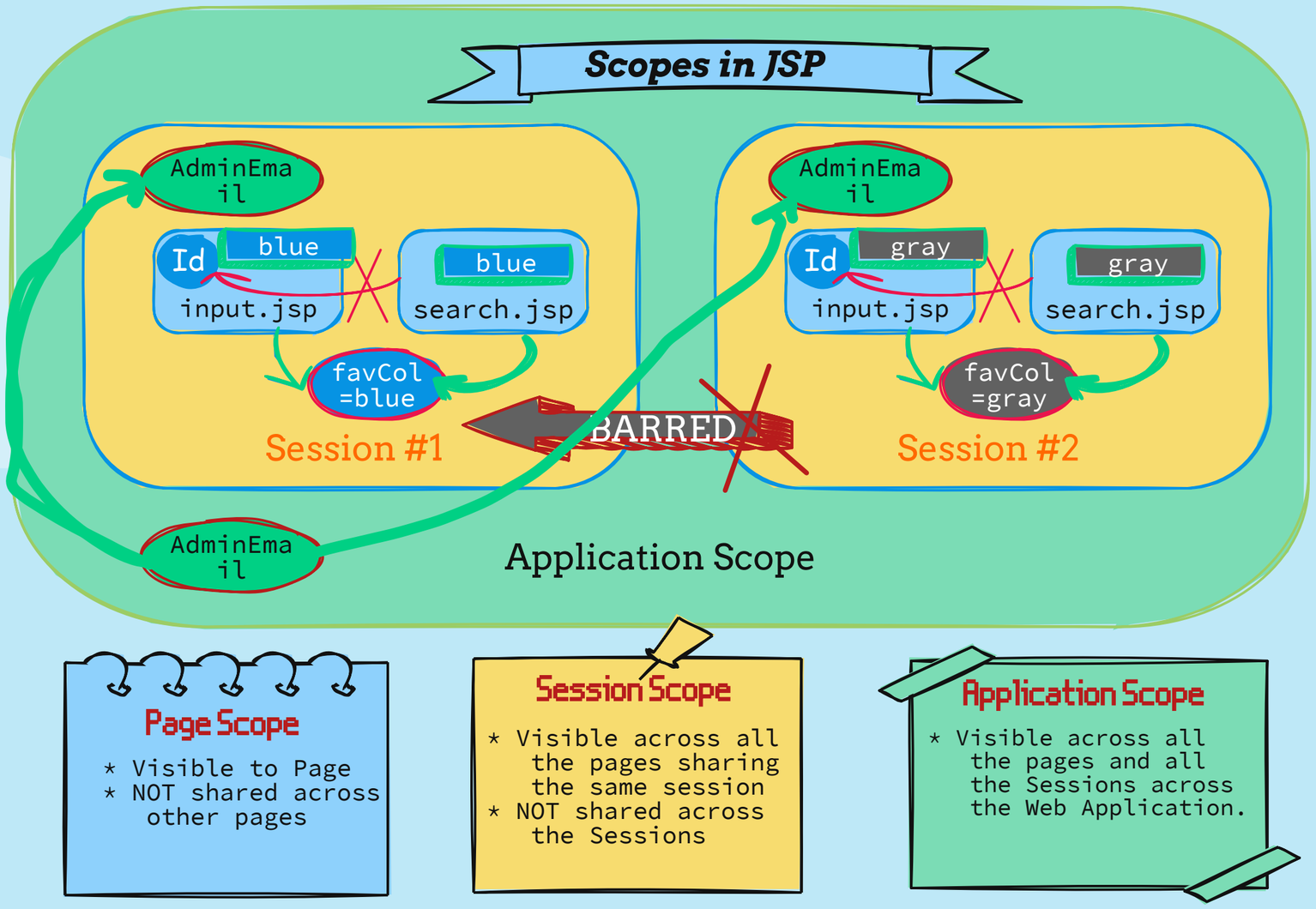


Adding the JDBC Jar into Eclipse Project

1. Right click project
2. Select Properties
3. Click the "Libraries" Tab
4. Select "Module Path"
5. Click "Add External JARs"
6. Choose the mysql-connector JAR file
7. Click Ok/Apply/Open
8. Click on "Order and Export" Tab
9. Ensure that this JAR is selected
10. Build the Project


Console Based JDBC Program
javac JDBCSample.java
java JDBCSample
Java
Java
Console
CLIENT
Java Program
(JDBCSample.java)
JDBC
JAR
JRE
Machine # 1
Database
(MySQL)
Machine # 2
SERVER
Java + HTML + CSS
HTML CSS JS
http://localhost:8080/jsp-demo/listAll.jsp
Web Application
JSP Based JDBC Program
jsp-demo
Tomcat
1. localhost
2. port #8080
3. Context Root ("jsp-demo")
JDBC JAR
Web Server/Container
Machine # 3
Java + HTML + CSS + JS
MADE WITH
BY
@ITSRAGHZ
USING
SKETCHWOW
Container
JRE
Runtime
@ITSRAGHZ / 07-SEP-2022


Before
Header
Content
Footer
HelloWorld.jsp
After
Header
A
Header.jsp
Content
B
Footer
C
Footer.jsp
HelloWorld.jsp
<%@ page import="..."/>
<html>
<head>
<title>JSP Page</title>
</head>
<body>
.....
.....
</body>
</html>
<%@ page import="..."/>
<html>
<head>
<title>JSP Page</title>
</head>
<body>
.....
.....
.....
</body>
</html>
A
B
C
JSP File Inclusion



Page Scope

* Visible to Page
* NOT shared across other pages

Session Scope

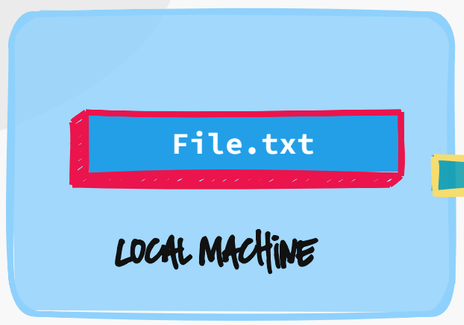
* Visible across all the pages sharing the same session
* NOT shared across the Sessions

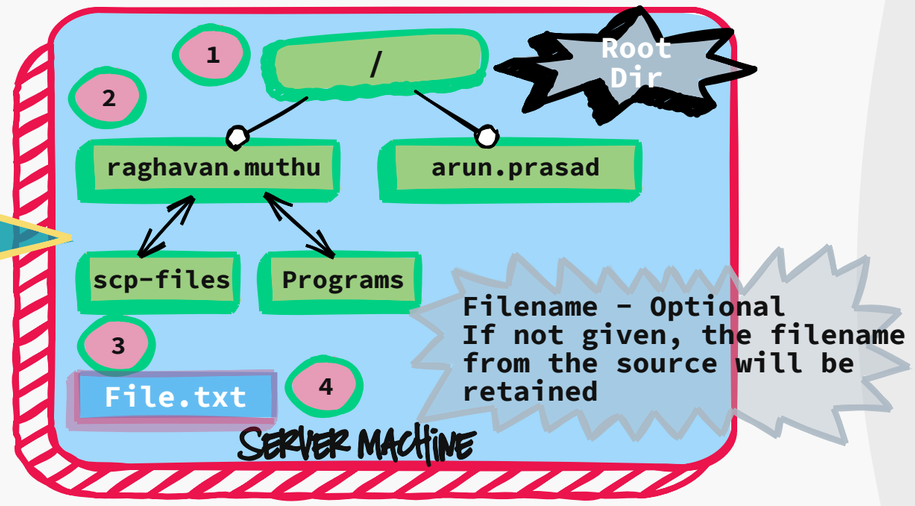
Application Scope

* Visible across all the pages and all the Sessions across the Web Application.

Secured Copy Protocol

Linux SCP Command





Filename - Optional
If not given, the filename from the source will be retained

/users/raghavan.muthu/File.txt

Data Ingredients

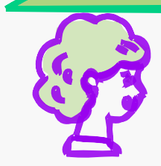
1. User - raghavan.muthu
2. Server - 10.121.1.241
3. target Location - /home/raghavan.muthu/scp-files

```
cp /dir/../source /dir/../target
```

```
scp /local/path/to/file user@a.b.c.d:/targetPath
```

```
scp /local/path/to/file user1@a.b.c.d:/home/user1/path
```

Path

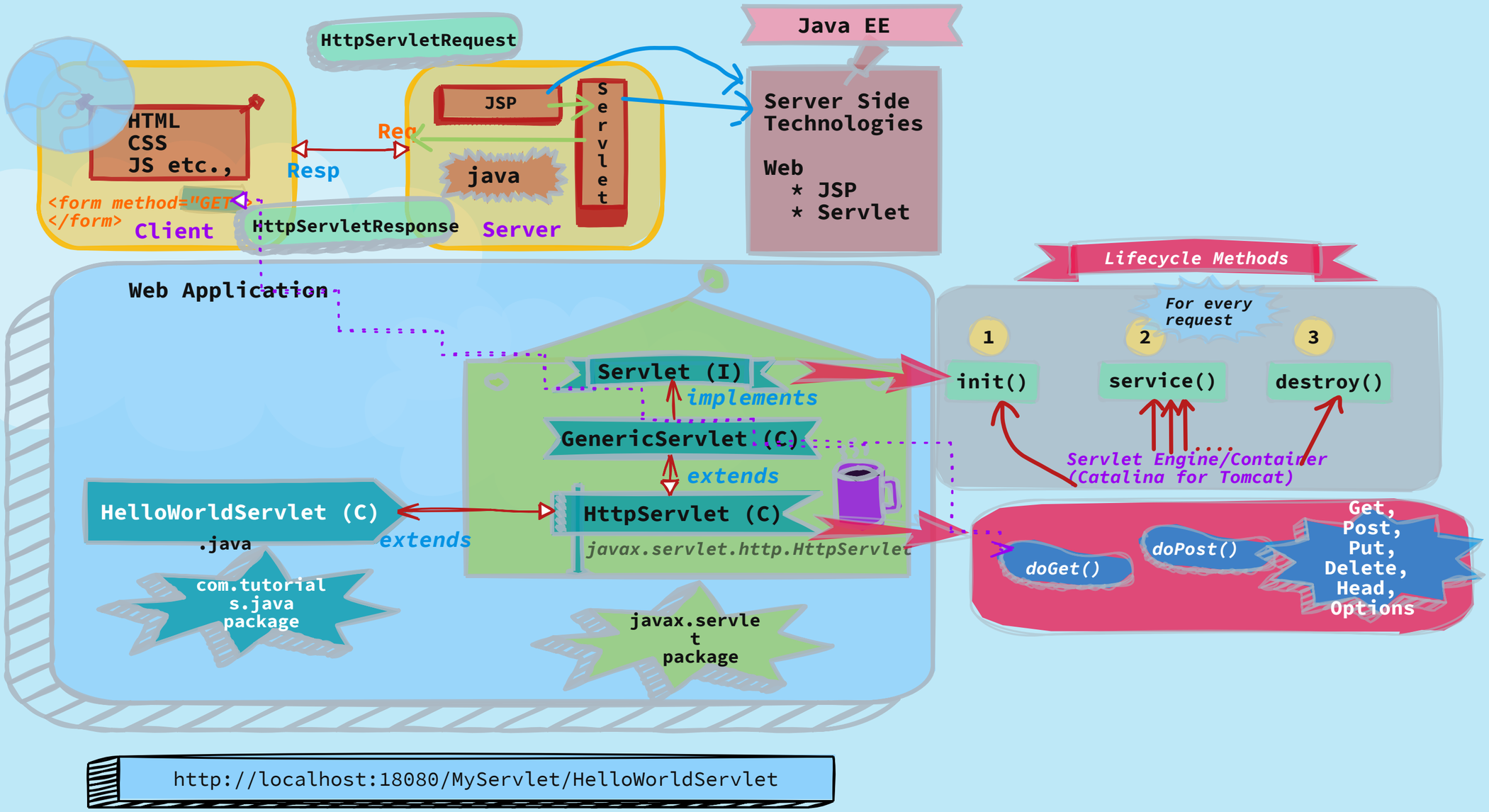


User Account
<userName>@<ServerIP>

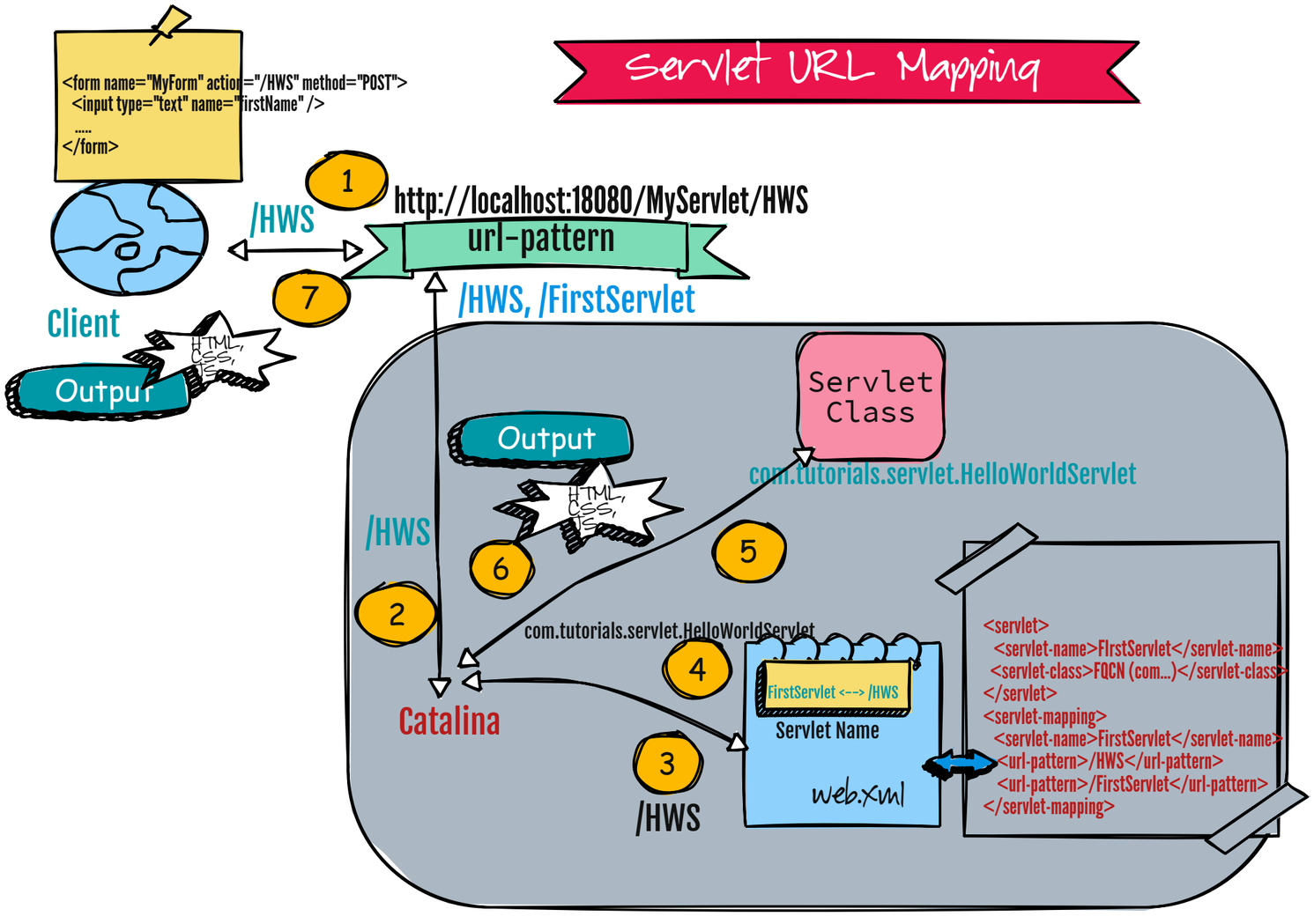
```
scp /local/path/to/file user1@a.b.c.d:/home/user2/path
```

```
scp /local/path/to/file user2@a.b.c.d:/home/user1/path
```

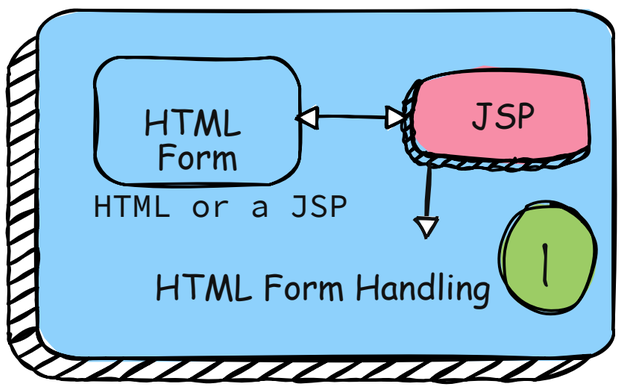
```
scp /local/path/to/file user2@a.b.c.d:/home/user3/path
```

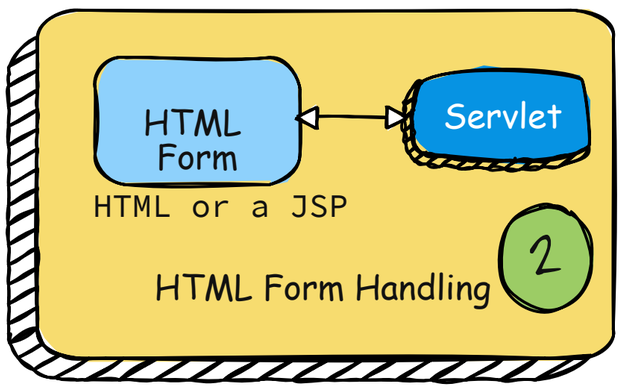

HttpServletRequest
Java EE
HTML
CSS
JS etc.,
JSP
Servlet
Server Side Technologies
Web
* JSP
* Servlet
Req
Resp
java
<form method="GET">
</form>
Client
HttpServletResponse
Server
Lifecycle Methods
Web Application
For every request
1
2
3
Servlet (I)
init()
service()
destroy()
implements
GenericServlet (C)
Servlet Engine/Container
(Catalina for Tomcat)
extends
HelloWorldServlet (C)
.java
HttpServlet (C)
extends
javax.servlet.http.HttpServlet
doGet()
doPost()
Get,
Post,
Put,
Delete,
Head,
Options
com.tutorials.java package
javax.servlet package
http://localhost:18080/MyServlet/HelloWorldServlet

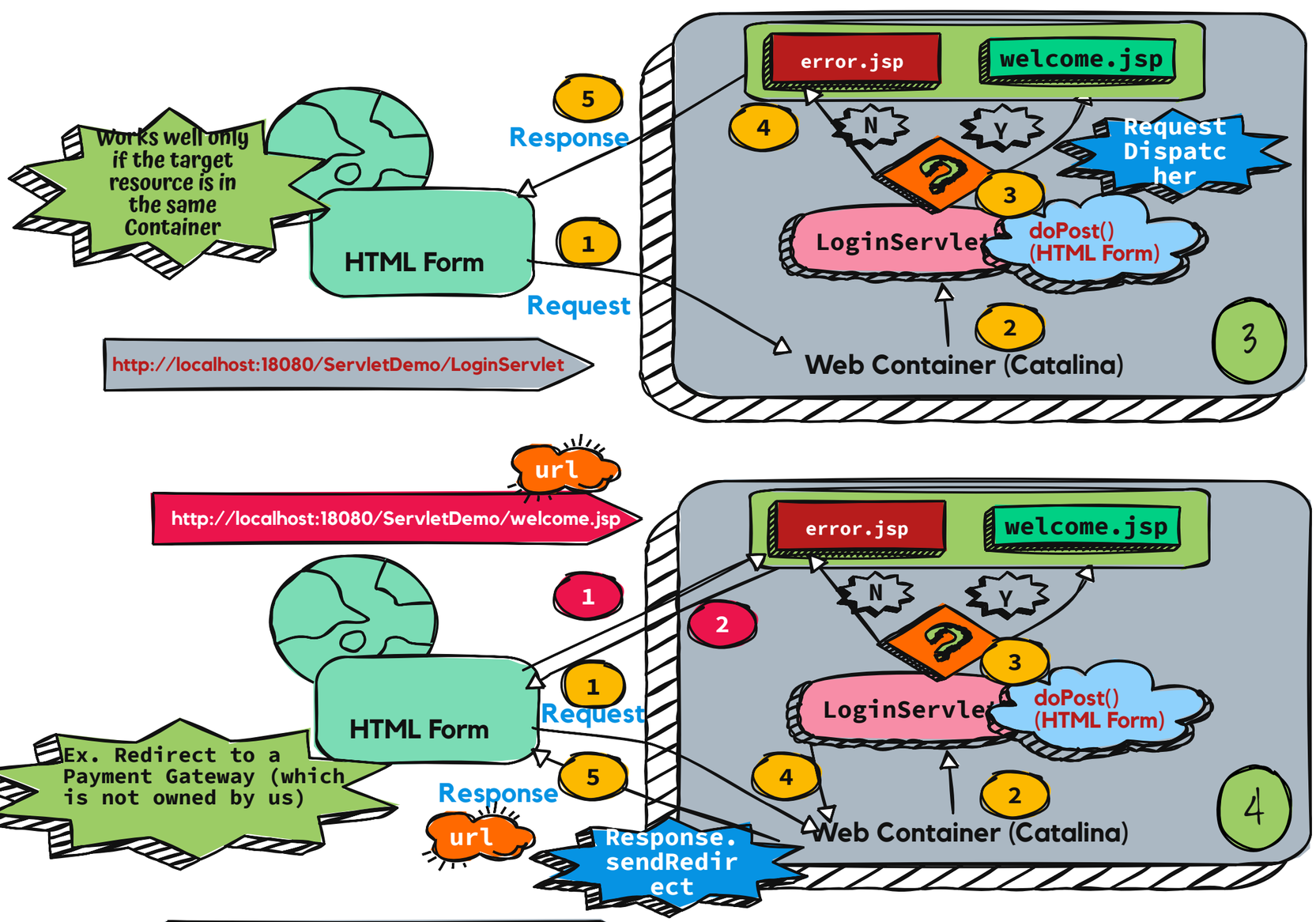
Servlet URL Mapping



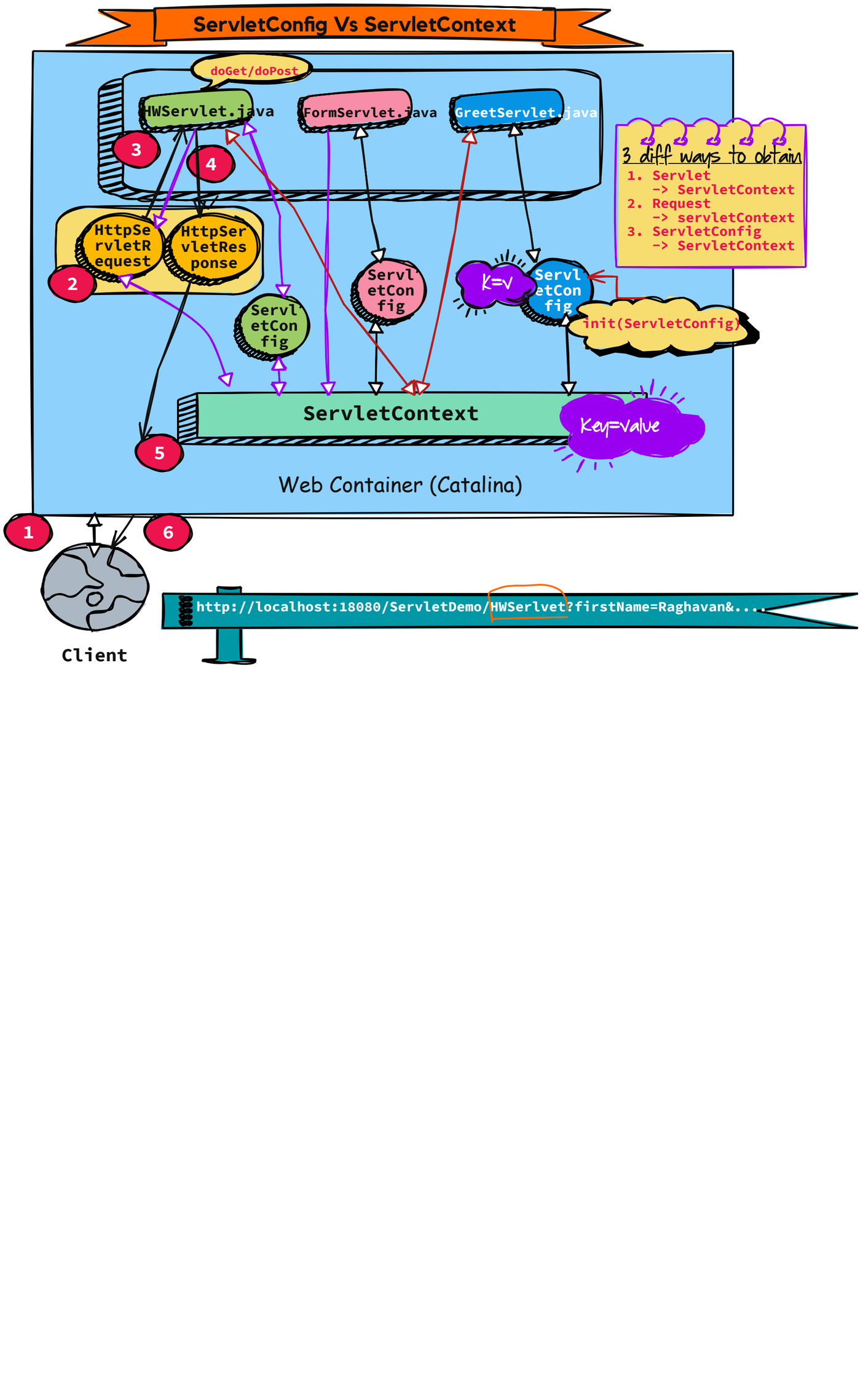
Different ways of Form Handling in Servlet/JSP

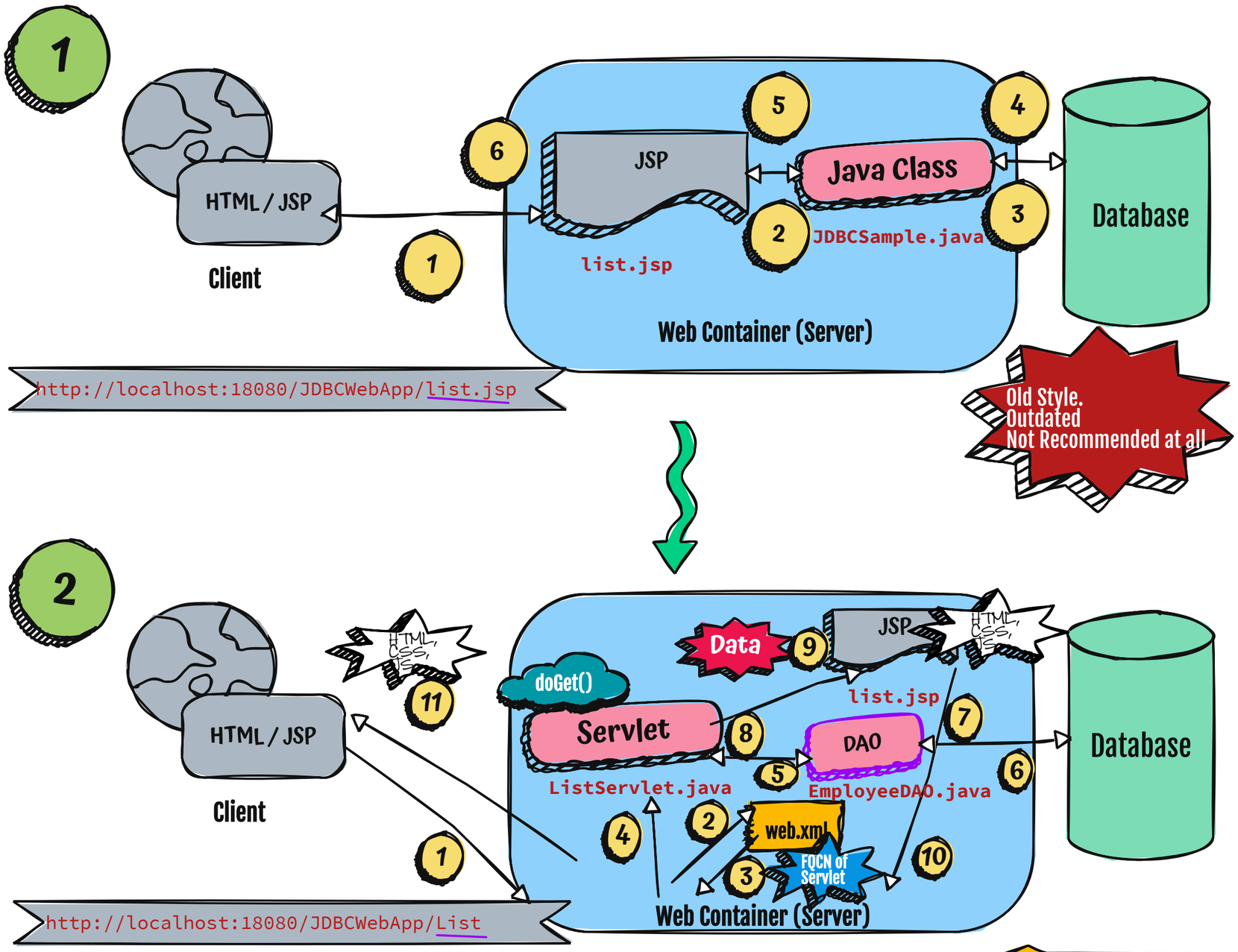






UrlRedirection - preferred when we want the browsers/ clients to redirect to an external domain/URL


ServletConfig Vs ServletContext
doGet/doPost
HWServlet.java
FormServlet.java
GreetServlet.java
3
4
HttpSe rvletR equest
HttpSer vletRes ponse
2
Servl etCon fig
Servl etCon fig
K=V
Servl etCon fig
3 diff ways to obtain
1. Servlet
-> ServletContext
2. Request
-> servletContext
3. ServletConfig
-> ServletContext
init(ServletConfig)
ServletContext
Key=value
5
Web Container (Catalina)
1
6
http://localhost:18080/ServletDemo/HWSerlvet?firstName=Raghavan&....
Client



MVC Model

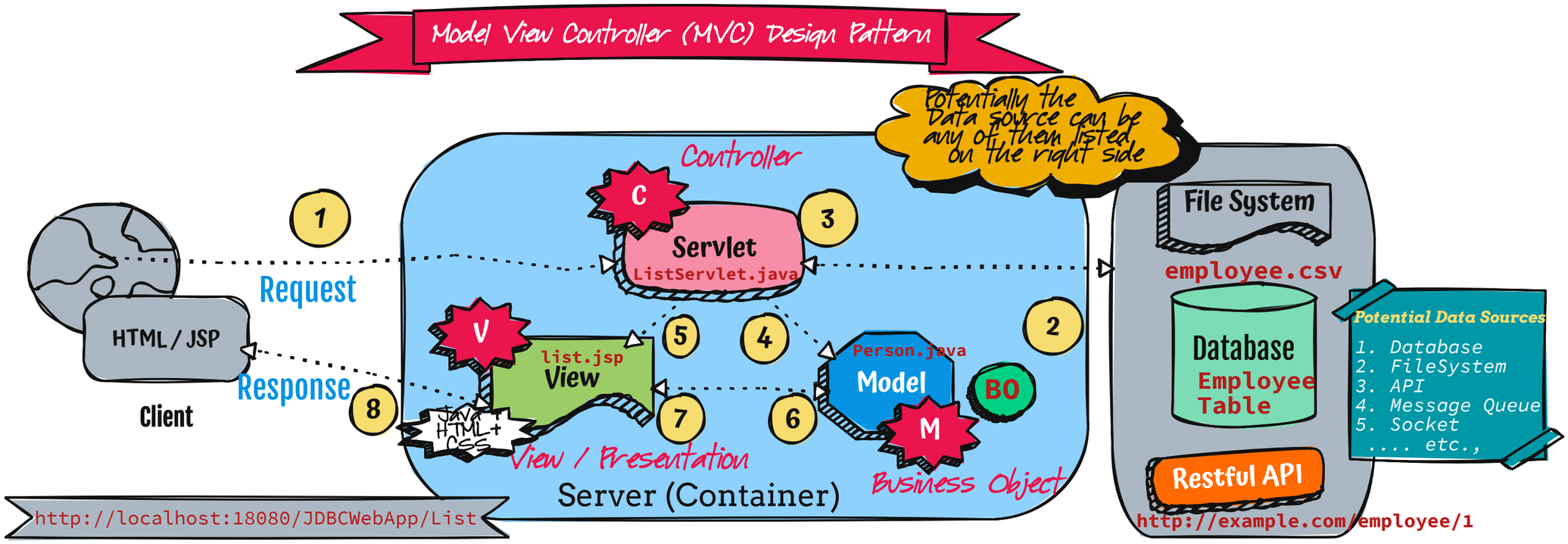
DAO - Data Access Object

* A design pattern that separates the Data Access Operations in a separate Java class
* EmployeeDAO - A DAO class deals with the Employee Table

Design Principle for a Class

A Class should handle only one responsibility

Single Responsibility Principle



Design Pattern - MVC

A Design patterns helps to have a proper separation of control in the Application flow.

* M (Model) - A Java class that stores the data
* V (View) - A JSP Page that displays the data by accessing / referring to the Model
* C (Controller) - A Java class (typically a Servlet in a Web Application) that controls the entire flow being a central component

C

* Central Component
* Controls the flow
* A Java Class
* A Servlet in Web app
* Invoked by the End Client usually

ListServlet.java

Music Director

V

* Used for Presentation or Display of the data
* Does not know/care the way data is obtained
* Invoked by Controller
* Refers the Model
* Shares the Final Output

list.jsp

Singer

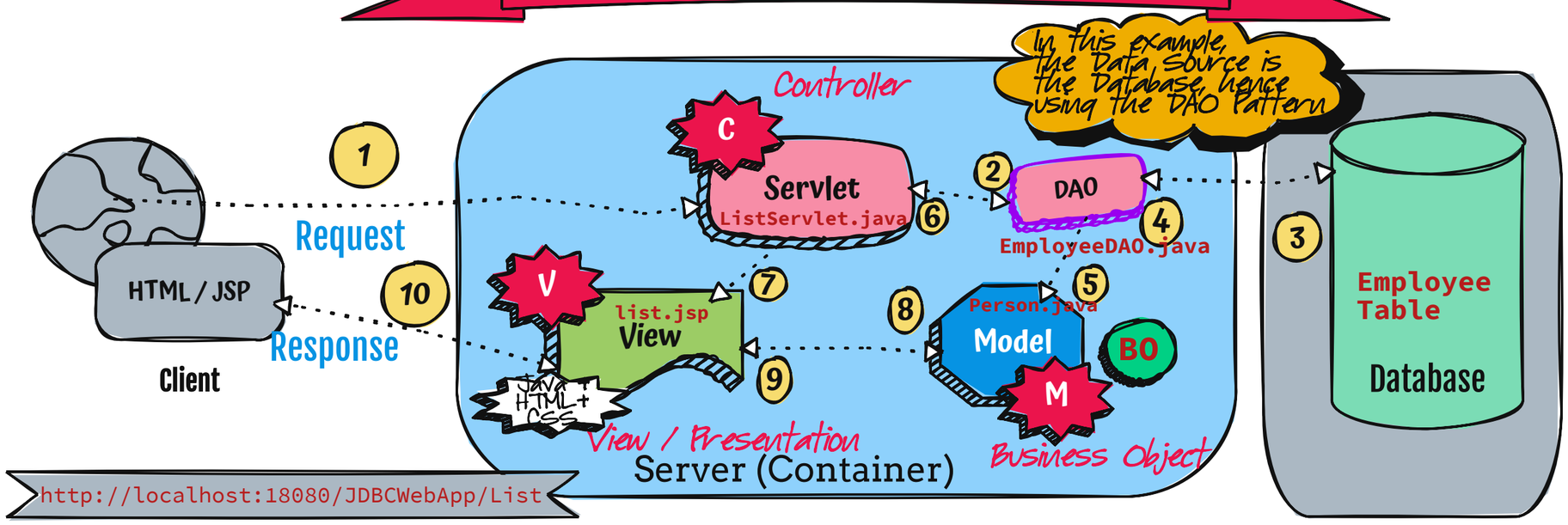
M

* A Java class to hold the business data
* Mostly filled from the Database Table
* Controller Updates
* View accesses it

Person.java

Lyricist

Model View Controller (MVC) + Data Access Object (DAO) Design Pattern



Design Pattern – MVC

A Design patterns helps to have a proper separation of control in the Application flow.

* M (Model) - A Java class that stores the data
* V (View) - A JSP Page that displays the data by accessing / referring to the Model
* C (Controller) - A Java class (typically a Servlet in a Web Application) that controls the entire flow being a central component

Music Director

C

* Central Component
* Controls the flow
* A Java Class
* A Servlet in Web app
* Invoked by the End Client usually

ListServlet.java

Singer

V

* Used for Presentation or Display of the data
* Does not know/care the way data is obtained
* Invoked by Controller
* Refers the Model
* Shares the Final Output

list.jsp

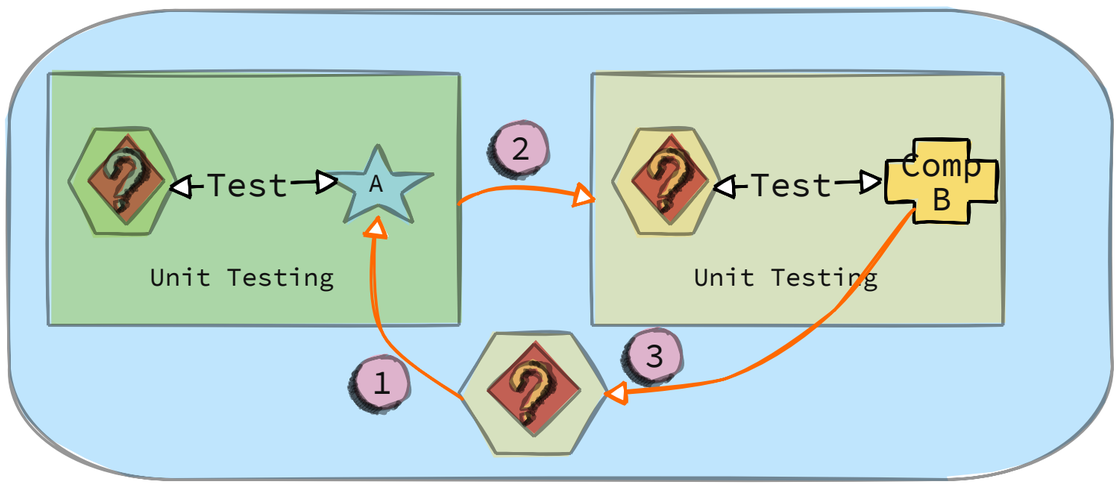
Lyricist

M

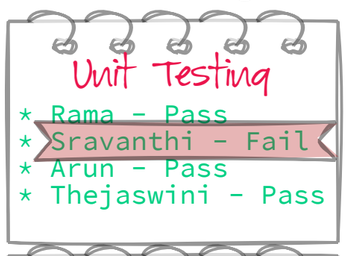
* A Java class to hold the business data
* Mostly filled from the Database Table
* Controller Updates
* View accesses it

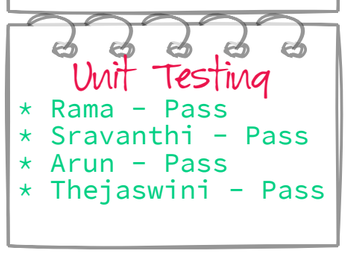
Person.java

Unit Vs Integration Testing



Integration Testing

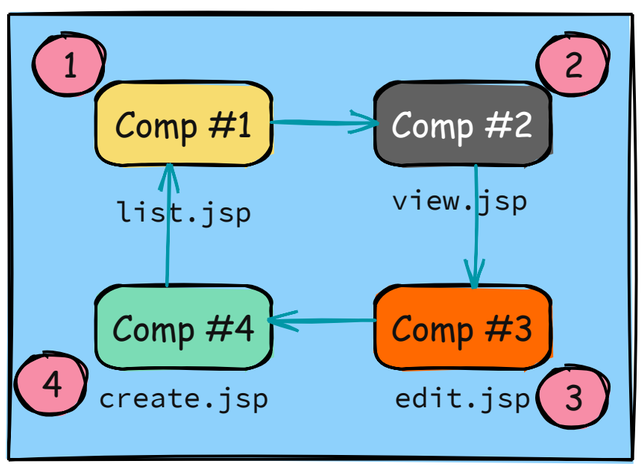




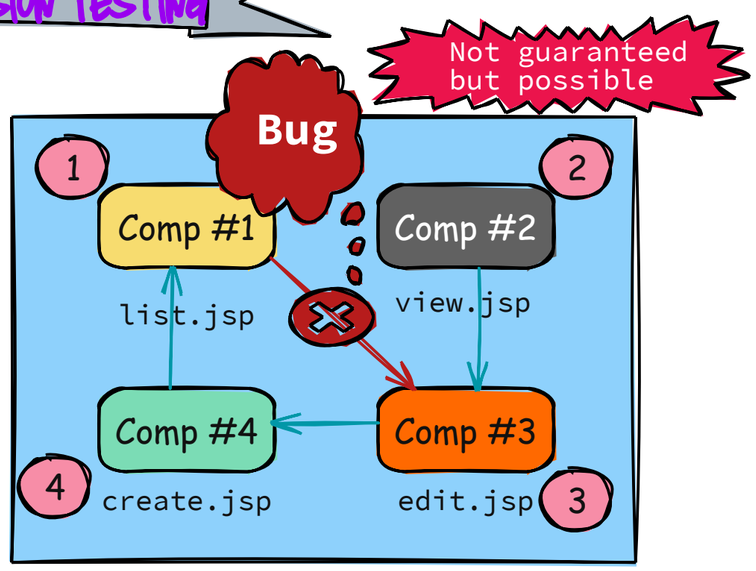
Unit Testing - performed at a component level

Integration Testing - performed at an Application level that combines all the components involved in the flow

SANITY VS REGRESSION TESTING



Happy Path Scenario
Sanity Testing



Regression Testing
Unusual / Mixed Path

Sanity -
Order of testing

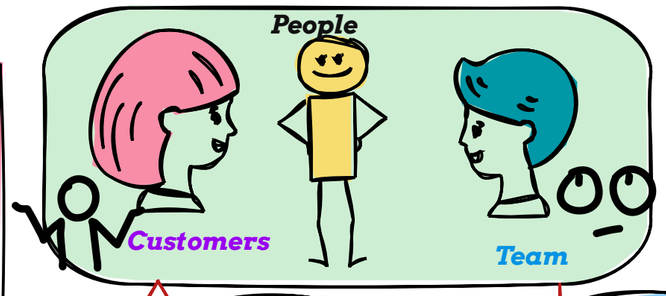
Sanity Vs Regression -
Usual or Natural Vs
Unusual or Mixed

Unit -
Quantity of Components

Unit Vs Integration -
Single Component Vs
Multiple Components

Software Development Life Cycle

SDLC Flow



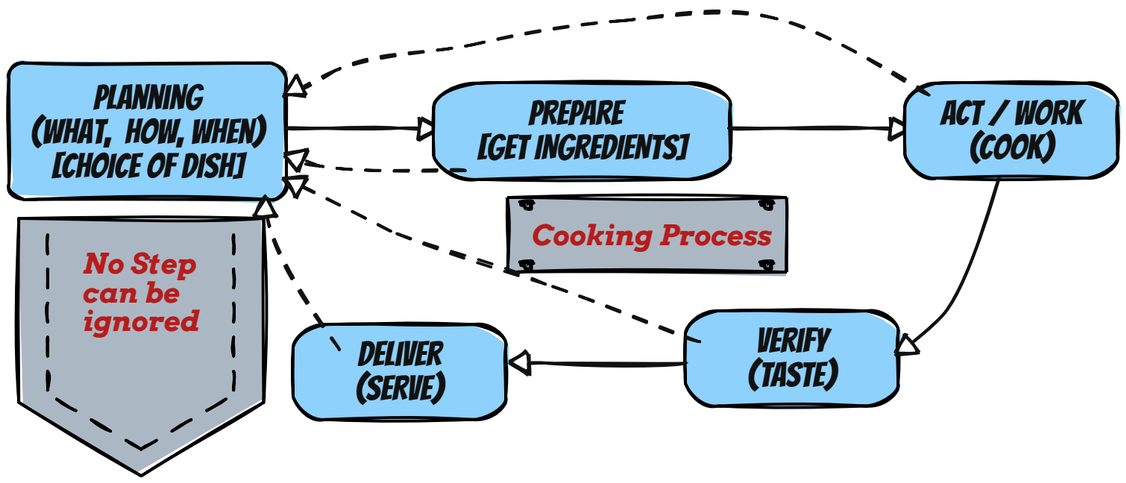
Employee Management
Student Management
NetBanking
Ticket Booking System
Library Management
Vehicle Tracking System
Food Delivery App
Online Shopping
Online Travel
Real Estate / Housing
Pharma / Consultancy
Matrimony etc.,

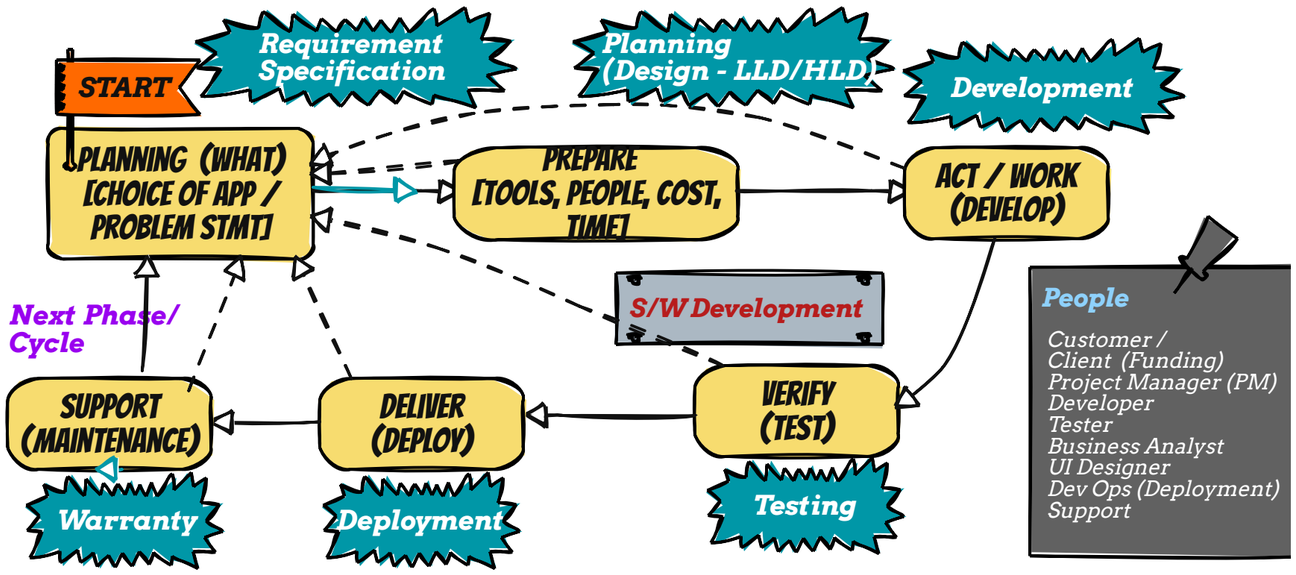
Java, MySQL
Angular, HTML
React,

Mobile, Responsive
iOS, Android, Windows

JIRA, Confluence, MS Word, Excel,
HP QC , DTS (Defect Tracking System)

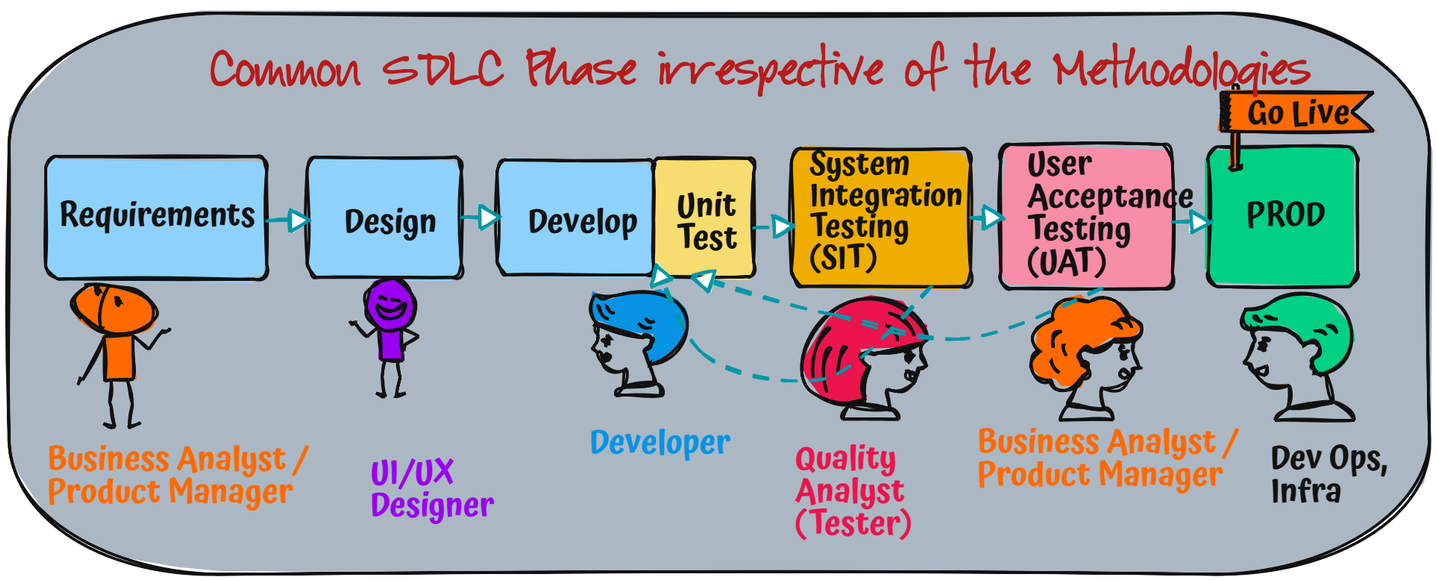
Cloud etc.,

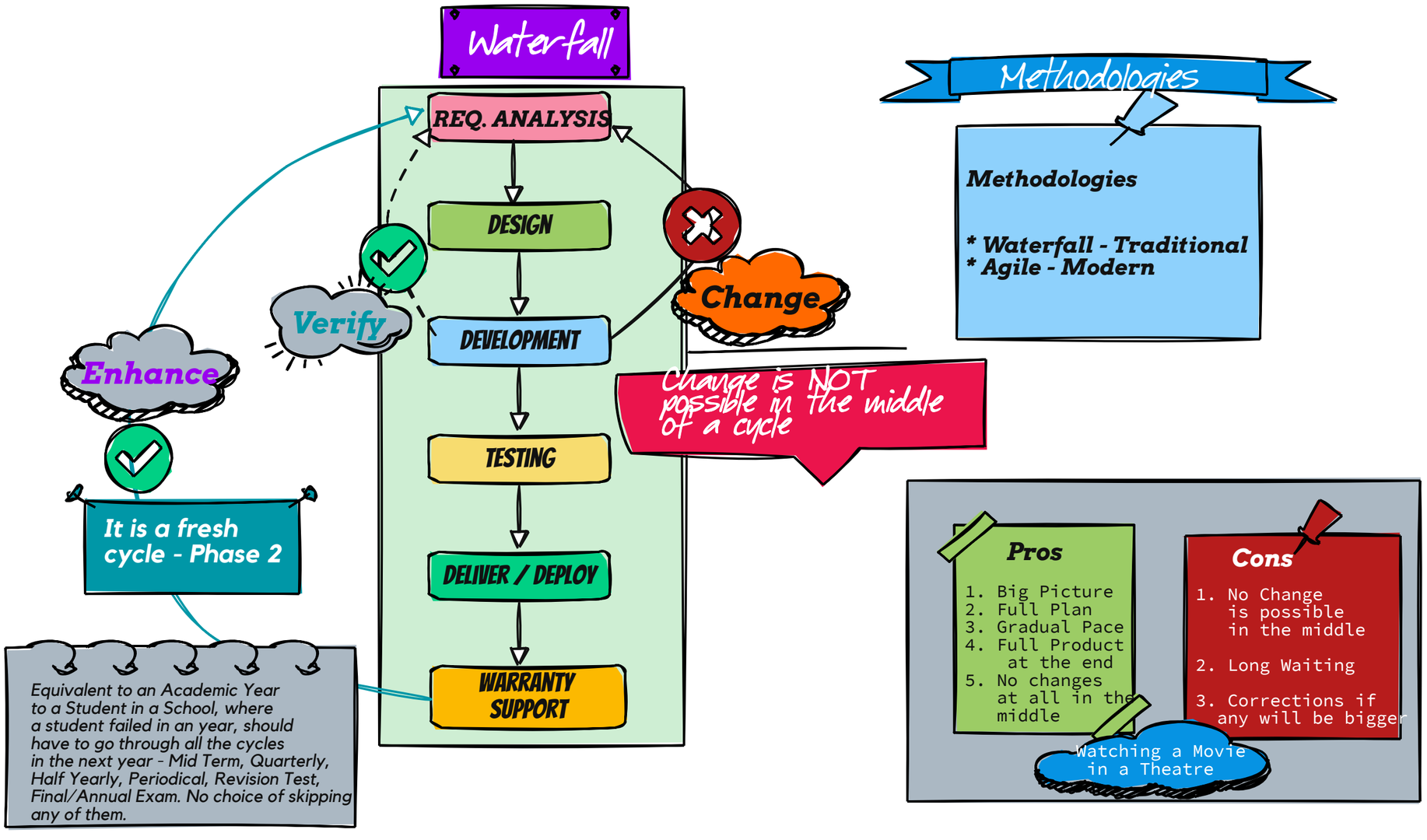


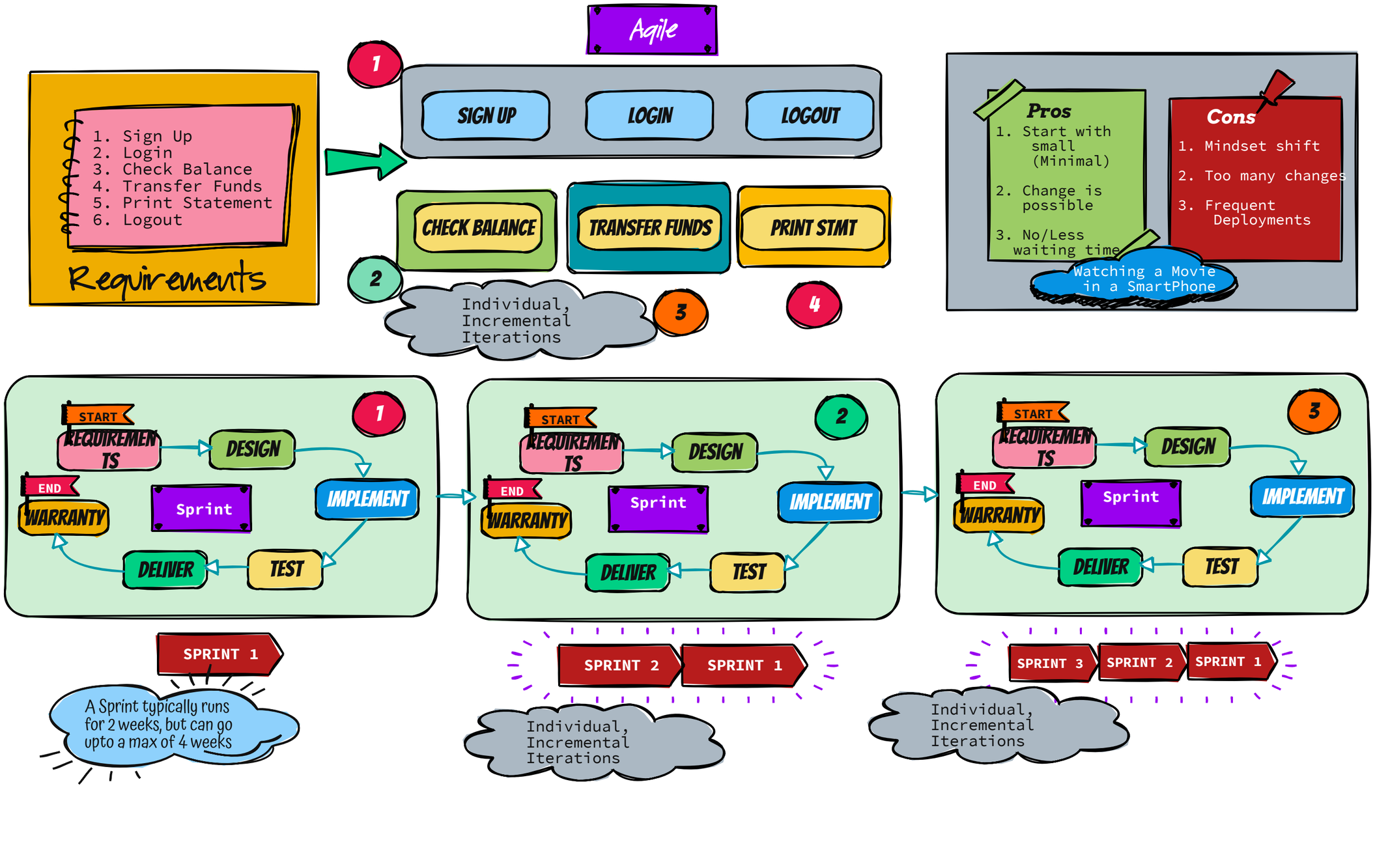


People

Customer / Client (Funding)
Project Manager (PM)
Developer
Tester
Business Analyst
UI Designer
Dev Ops (Deployment)
Support




Waterfall
REQ. ANALYSIS
DESIGN
DEVELOPMENT
TESTING
DELIVER / DEPLOY
WARRANTY SUPPORT
Verify
Change
Enhance
It is a fresh cycle - Phase 2
Change is NOT possible in the middle of a cycle
Equivalent to an Academic Year to a Student in a School, where a student failed in an year, should have to go through all the cycles in the next year - Mid Term, Quarterly, Half Yearly, Periodical, Revision Test, Final/Annual Exam. No choice of skipping any of them.
Methodologies
Methodologies
* Waterfall - Traditional
* Agile - Modern
Pros
1. Big Picture
2. Full Plan
3. Gradual Pace
4. Full Product at the end
5. No changes at all in the middle
Cons
1. No Change is possible in the middle
2. Long Waiting
3. Corrections if any will be bigger
Watching a Movie in a Theatre


Agile
1
SIGN UP
LOGIN
LOGOUT
CHECK BALANCE
TRANSFER FUNDS
PRINT STMT
2
3
4
Individual, Incremental Iterations
1. Sign Up
2. Login
3. Check Balance
4. Transfer Funds
5. Print Statement
6. Logout
Requirements
Pros
1. Start with small (Minimal)
2. Change is possible
3. No/Less waiting time
Cons
1. Mindset shift
2. Too many changes
3. Frequent Deployments
Watching a Movie in a SmartPhone
START
REQUIREMENTS
DESIGN
IMPLEMENT
TEST
DELIVER
END
WARRANTY
Sprint
SPRINT 1
A Sprint typically runs for 2 weeks, but can go upto a max of 4 weeks
SPRINT 2
SPRINT 3
Individual, Incremental Iterations

SQL - Primary, Foriegn Keys

Attributes

Id	Name	Age	Gender	Address
1	Ram	21	M	
2	Seetha	18	F	

Person — ENTITY

No | Bldg | Street | Area | City | PIN

Primary Key

Id.	Name.	Age.	Gend
1	Ram	21	M
2	Seetha	18	F

Person — ENTITY

Primary Key

Id.	Ref Id.	Type.	Address
1	1	Home	12, MG Road, Bangalore
2	1	Office	29, Residency Road, Bangalore
3	2	Home	4, Main Road, Trichy
4	2	Hotel	55, Beach Road, Goa

FK

Address — ENTITY

Normalization

- 1NF
- 2NF
- 3NF
- BCNF

1. 1st Normal Form
2. Separate the additional attributes from an entity and keep them a part of a separate entity
3. To prevent the Update/ Deletion Anamolies

Boyce Code Normal Form

Primary Key (PK)

1. Primary Key - Unique and NOT Null
2. Automatically Incremented
3. Mostly referred as ID colu (Identifier)
4. Short name - PK

Foreign Key (FK)

1. Foreign Key - dependent on the PK of another table
2. Can have duplicates (1:N)
3. Referred as XYZ_ID where XYZ refers the other table
4. Short name - FK
5. Can NOT be null

Referential Integrity - The term by which the data referred between two tables should be intact at all the times

VERSION CONRTOL SYSTEM (VCS)

9:00 AM
Snapshot #1
11:00 AM
Snapshot #2
5:00 PM
Snapshot #6

Version Control

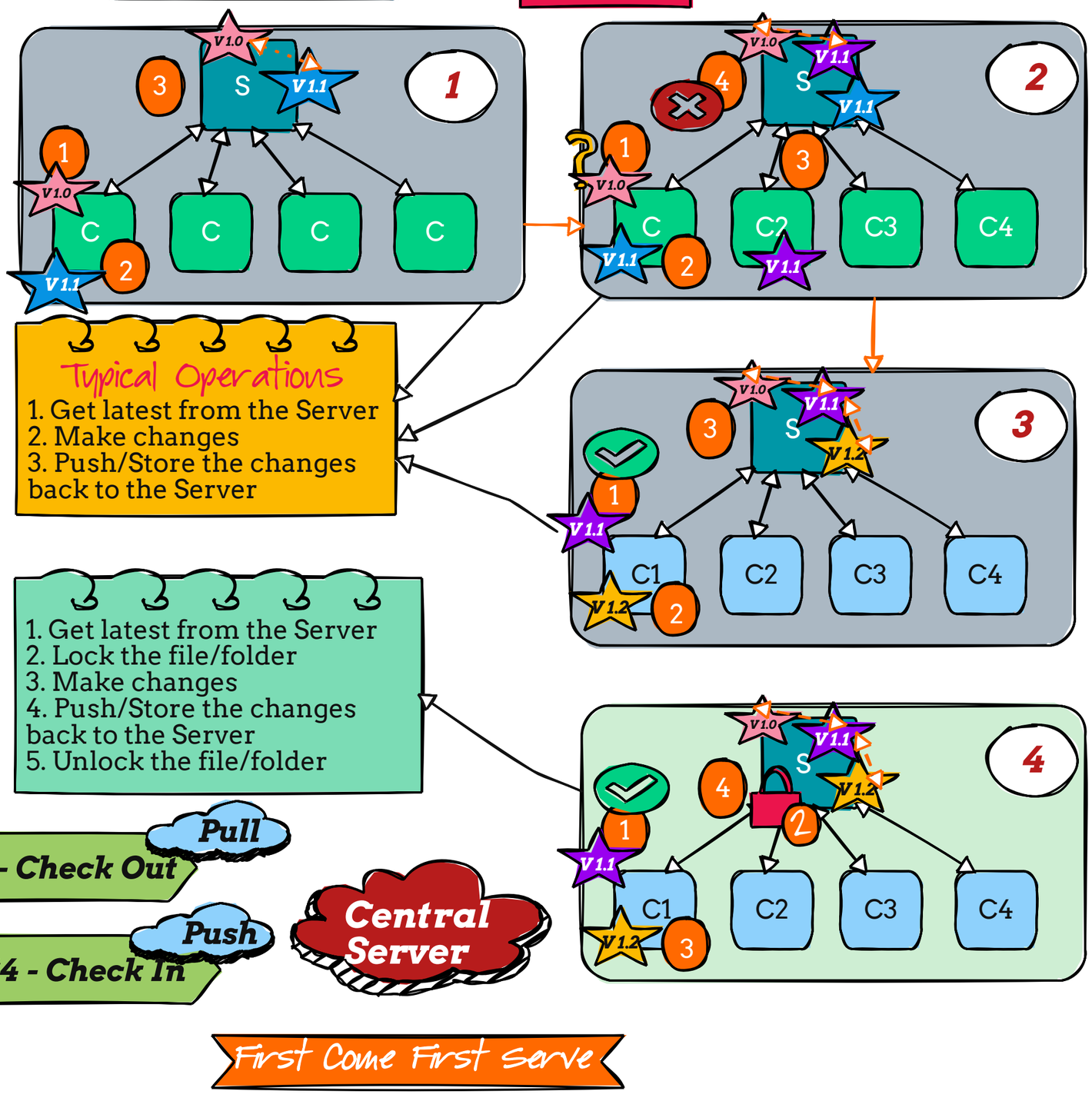
* Multiple versions
* COmparison (diff)
* Backup
* Quick and easy

Example - Bus Conductor monitoring all the passengers and maintain a status of tickets being issued at various times

Client Server Model

Traditional VCS

1. CVS
2. VSS
3. SVN
etc.,



GIT (DVCS)

Modern

D - Distributed
V - Version
C - Control
S - Systems

Typical Operations

1. Get latest from the Server
2. Make changes
3. Push/Store the changes back to the Server

1. Get latest from the Server
2. Lock the file/folder
3. Make changes
4. Push/Store the changes back to the Server
5. Unlock the file/folder

Pull
#1 - Check Out

Push
#4 - Check In

Central Server

First Come First serve

Git - DVCS

D - Distributed
V - Version
C - Control
S - Systems

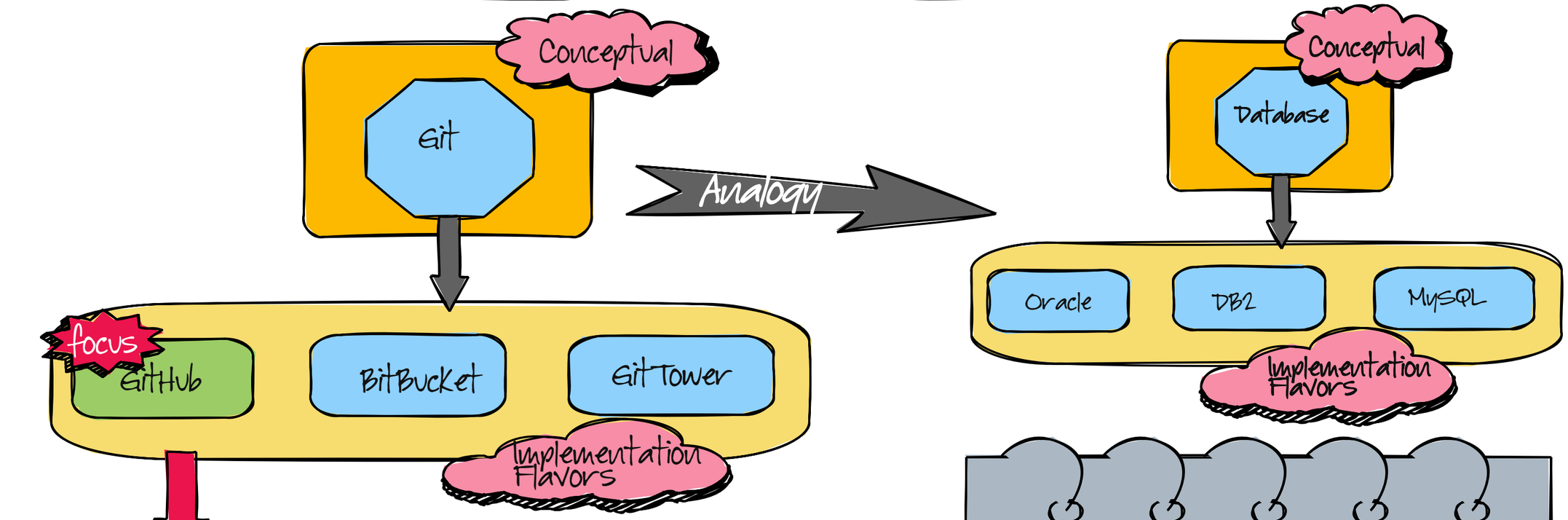
Made with ♥ Using SketchWow by itsraghz 09.Nov.2022

Legend
- Frequent
- In-frequent

C - Create
C - Check Out
C - Changes
C - Commit
C - Check In
M - Merge

MASTER (Central)
VCS
SERVER
Merge
5
Create
0
Remote
Push
4
Pull
1
Check Out
Check In
N/W
Local
Local (Dev)
Staging
2
Changes
3
Commit
CLIENT #1
CLIENT #2
CLIENT #3

Git & Flavors, Git In Action



STEP BY STEP ACTIONS

1. GitHub Account
2. Create a Repo - Central Copy
3. Init the Local Git with Name & Email of yours
4. Create SSH Keys for your m/c
5. Upload(Store) in the Github Acct
6. Get latest (Clone)
7. Make Changes
8. Commit (Staging)
9. Check in / Push (Remote)

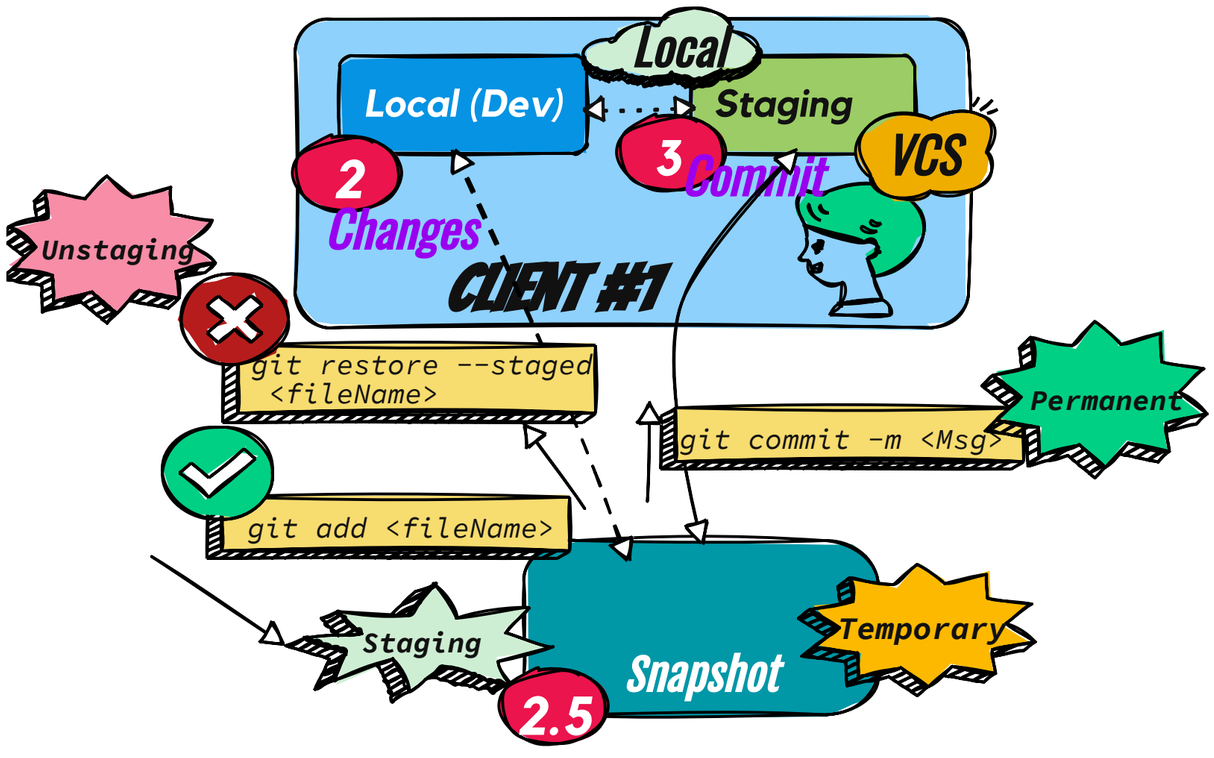
https://github.com/raghsonlinedotcom/001-git-demo

Git Bash for Windows (A cmd line utility)

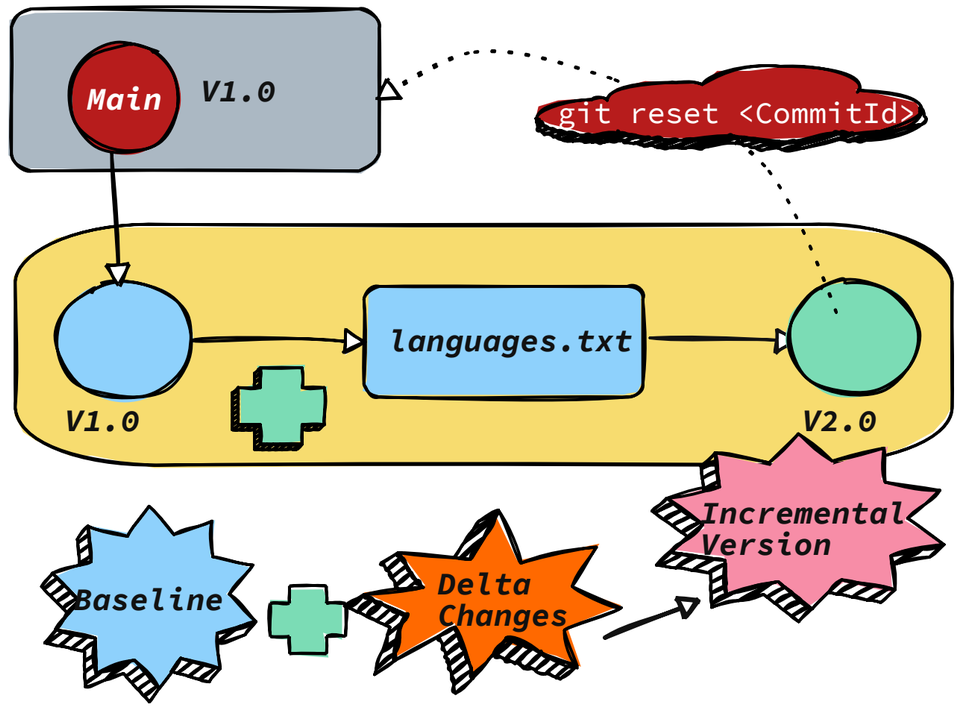
USEFUL GIT COMMANDS

1. git --version
2. git clone <remote-repo-url>
   Ex: git@github.com:raghsonlinedotcom/001-git-demo.git
3. git status
4. git branch
5. git add <fileName> / git add <file1> <file2> / git add .
6. git restore --staged <fileName>
7. git push
8. git log / git log --oneline

Git Local - Deep Dive



Git Incremental Delta



Java Version History

Made with by @itsraghz using SketchWow

Imperative Programming

1. Java = from J2SE
2. Only Major version
3. Name for each version

Oracle took over Java from Sun MicroSystems

1995	Jan 1996	Feb 1997	Dec 1998	May 2000	Feb 2002	Sep 2004	Dec 2006	2010	Jul 2011
0.1	1.0	1.1	1.2	1.3	1.4	Tiger 5.0	Mustang 6.0		Dolphin 7.0
JDK Beta	JDK 1.0 Java 1	JDK 1.1	J2SE 1.2	J2SE 1.3	Java 2	Java 5	Java 6		Java 7
	Inner classes Java Beans JDBC RMI, UniCode Seralization Reflection I18N	strictfp - keyword Swing GUI Collections Java IDL JVM + JIT Compiler	Hotspot JVM RMI + CORBA JNDI in Core JPDA Java Sound Synthetic Proxy	assert keyword Regular Exp. Exception Chaining IPV6 Support NIO (Non blocking) Java Web Start JAXP, JCE, JAAS	Generics VarArgs Static Imports Auto Boxing Enum Annotations	JDBC 4.0 Support JAXB with StAX Parser Minor Language API improvements	1. Underscore in the float values 2. Diamond Operator 3. Fork/Join changes 4. Try with Resources 5. AutoCloseable Interface 6. NIO (New IO)	

JCP - Java Community Process
JSR - Java Specification Request

JSR
JCP

JDK (Java Development Kit)

Functional Programming

6 Months Release Schedule

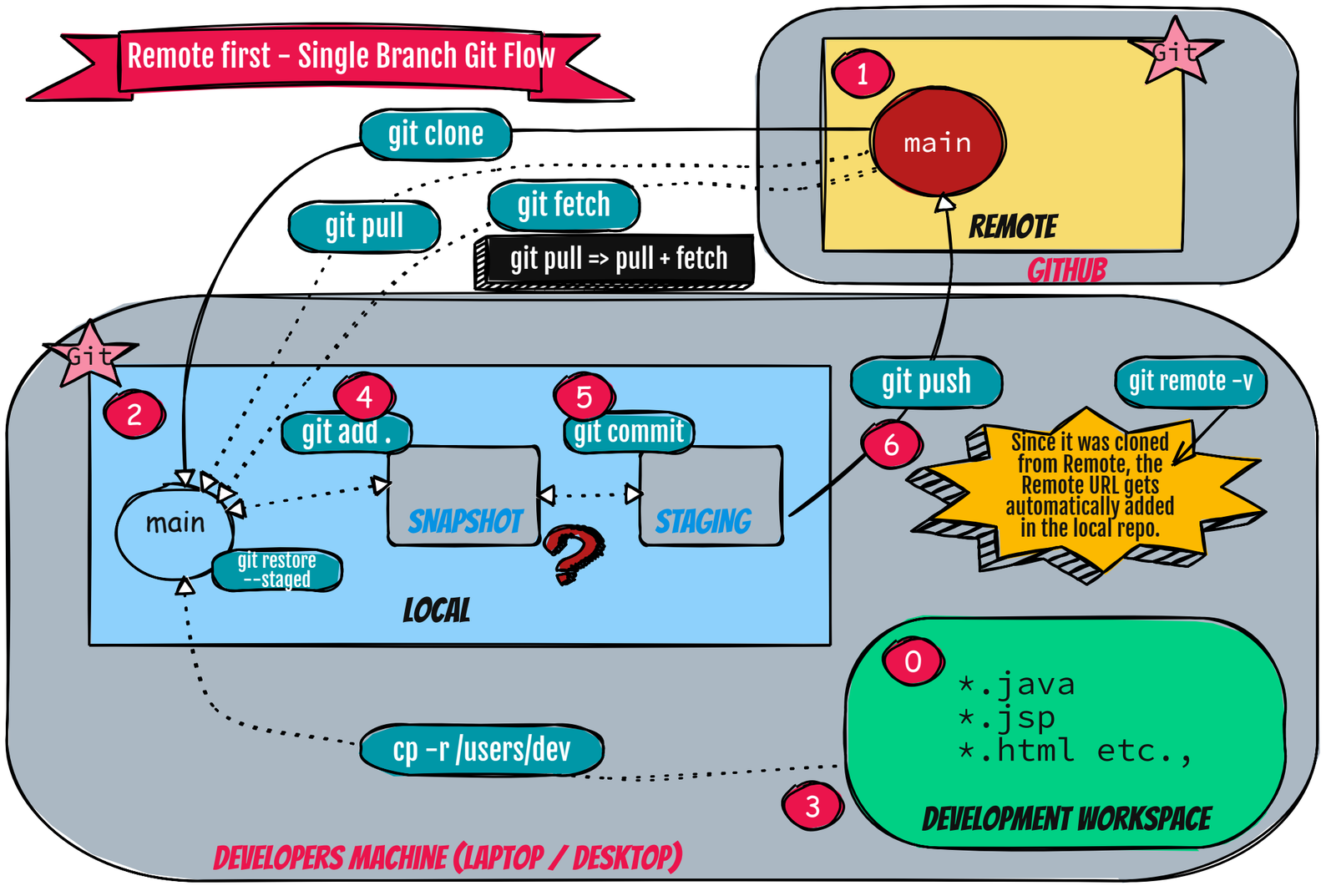
Mar 2014	Sep. 2017	Mar 2018	Sep 2018	Mar 2019	Sep 2019	Mar 2020	Sep 2020	Mar 2021	Sep 2021	Mar 2022	Sep 2022
8.0 LTS	9.0	10.0	11.0 LTS	12.0	13.0	14.0	15.0	16.0	17.0 LTS	18.0	19.0 LTS
Java 8	Java 9	Java 10	Java 11	Java 12	Java 13	Java 14	Java 15	Java 16	Java 17	Java 18	Java 19
1. Functional Programming -- Functional Interface -- Lambdas -- Streams -- Method References -- Date Time API changes	JPMS - Modular Programming REPL (jShell)	var keyword	HttpClient Flight Recorder Local Var to Lambda Parameters	JVM Constants API Enhanced Switch Expressions	Text Blocks - Preview	Records - Preview Safe NullPointers Pattern Matching for instanceOf - Preview	Sealed Classes - Preview Hidden Class Text Blocks Records - 2nd Preview	Vector API for computation - Incubator Records Pattern matching for instanceOf	Sealed Classes strictfp - keyword made absolete	UTF8 default Simple Web Server @snippet tag in Javadoc	Virtual Threads - Preview Record Patterns - Preview Pattern matching for switch - Preview

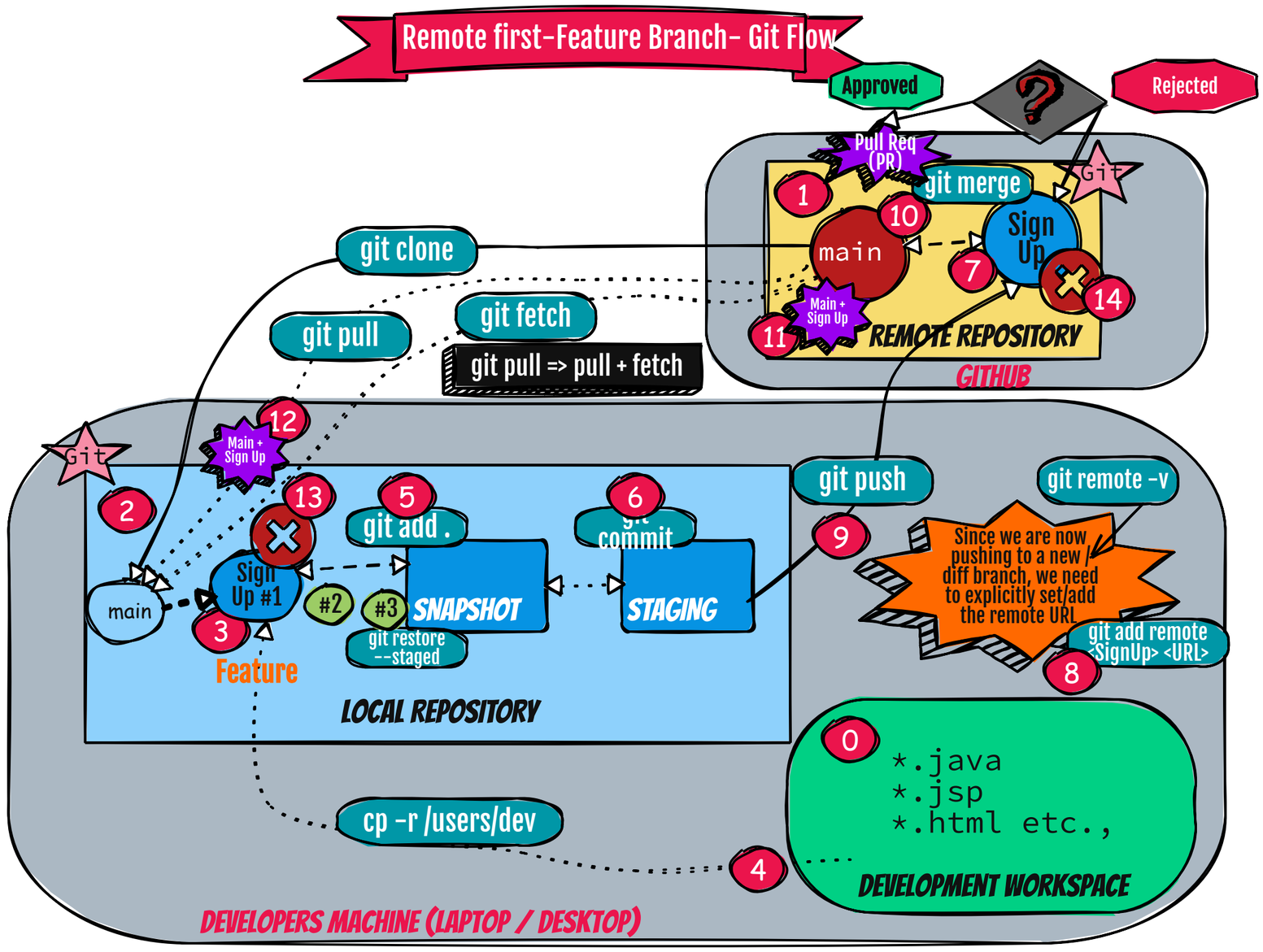
Version Names were dropped

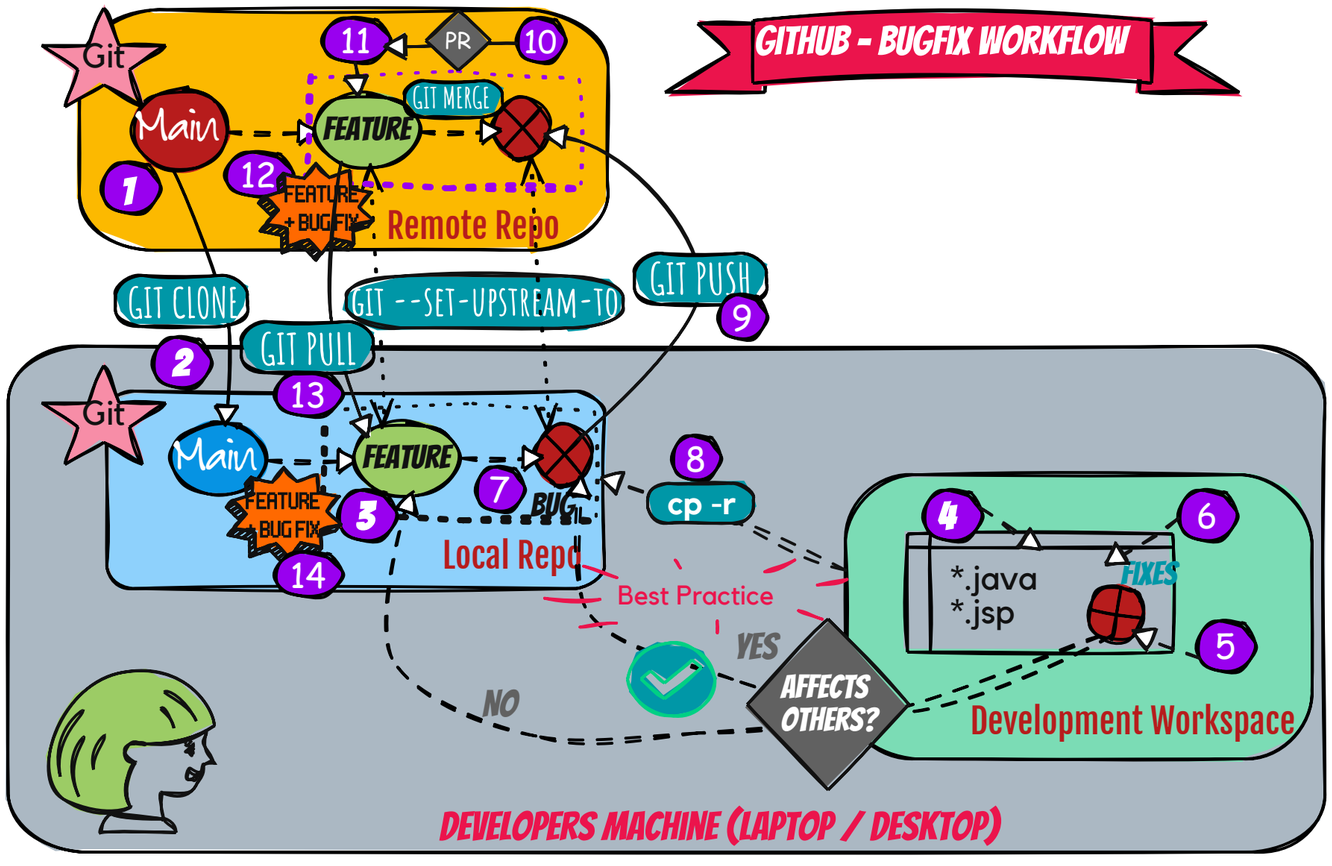
https://en.wikipedia.org/wiki/Java_version_history

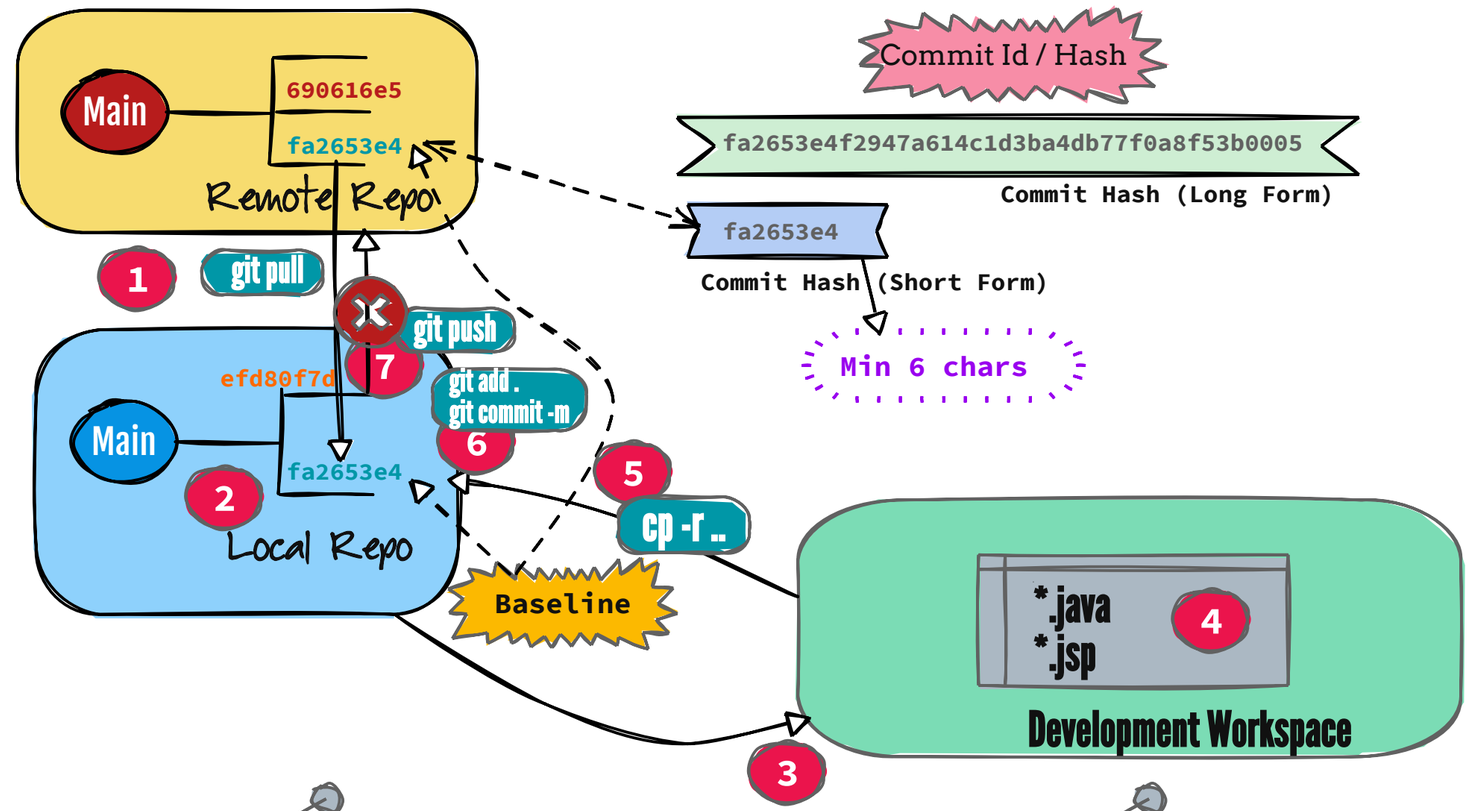
Stable & Long window for the EOL (End Of Life)

Long Term Support

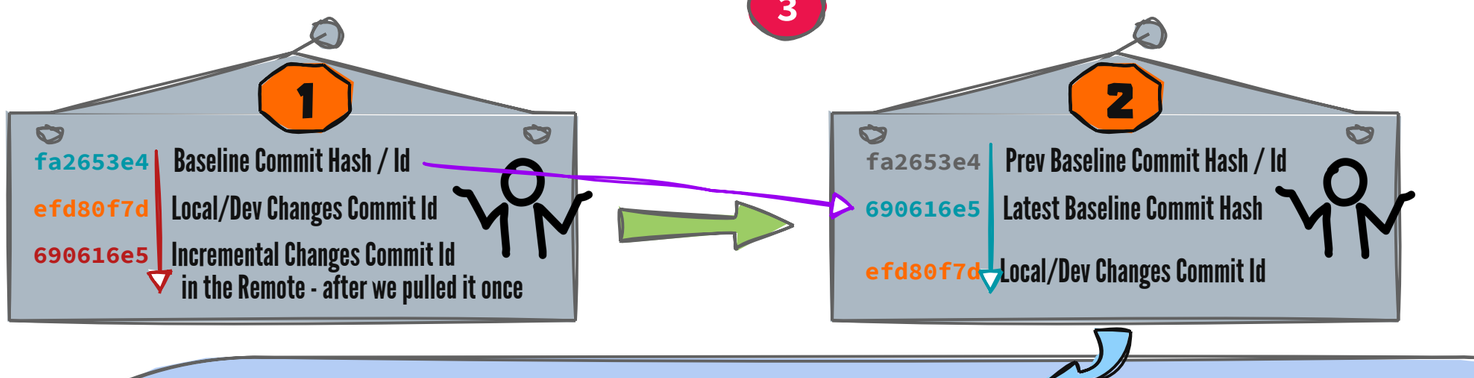

Remote first - Single Branch Git Flow
git clone
git pull
git fetch
git pull => pull + fetch
1
main
Git
REMOTE
GITHUB
Git
2
4
git add .
5
git commit
main
SNAPSHOT
STAGING
git restore --staged
LOCAL
git push
6
git remote -v
Since it was cloned from Remote, the Remote URL gets automatically added in the local repo.
0
*.java
*.jsp
*.html etc.,
DEVELOPMENT WORKSPACE
cp -r /users/dev
3
DEVELOPERS MACHINE (LAPTOP / DESKTOP)

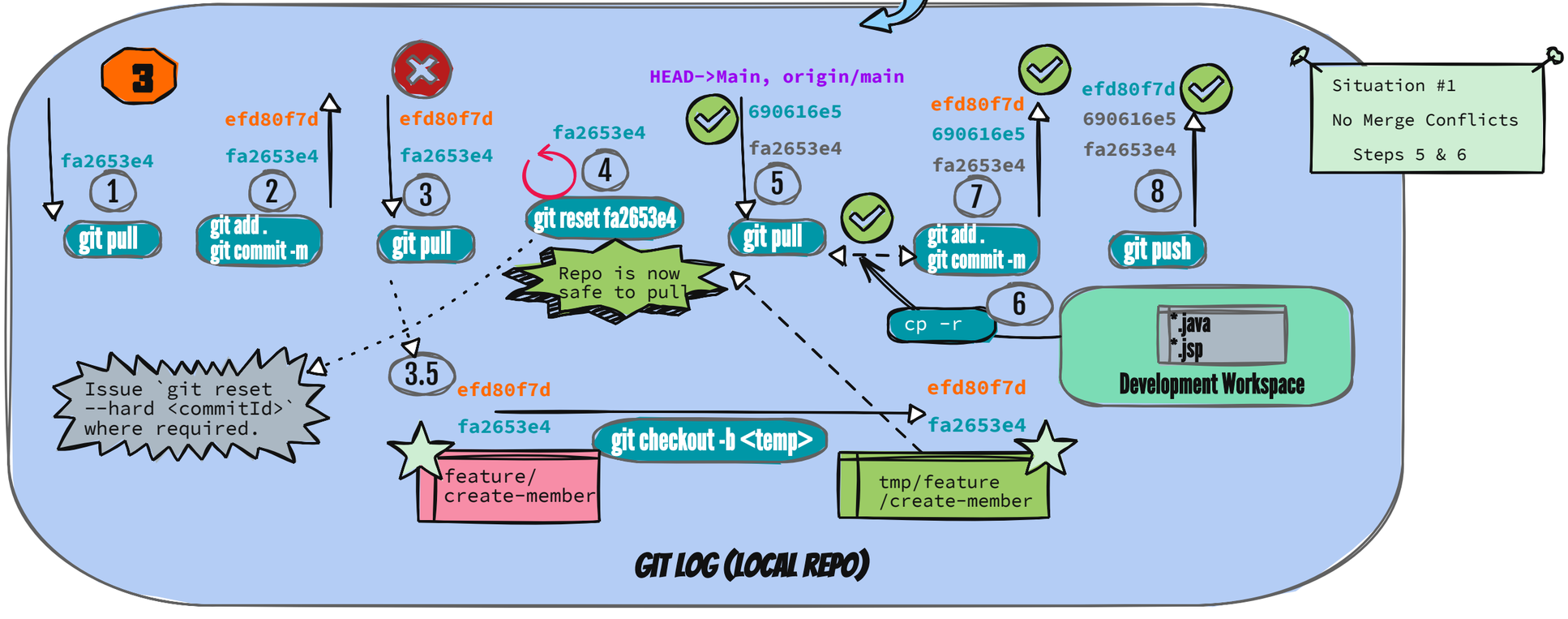

Remote first-Feature Branch- Git Flow
Approved
Rejected
Pull Req (PR)
git merge
Git
1
10
main
Sign Up
7
14
Main + Sign Up
11
REMOTE REPOSITORY
GITHUB
git clone
git pull
git fetch
git pull => pull + fetch
12
Main + Sign Up
Git
2
13
5
git add .
6
commit
git push
9
git remote -v
Since we are now pushing to a new / diff branch, we need to explicitly set/add the remote URL
main
Sign Up #1
#2
#3
SNAPSHOT
STAGING
3
Feature
git restore --staged
git add remote <SignUp> <URL>
8
LOCAL REPOSITORY
0
*.java
*.jsp
*.html etc.,
cp -r /users/dev
4
DEVELOPMENT WORKSPACE
DEVELOPERS MACHINE (LAPTOP / DESKTOP)

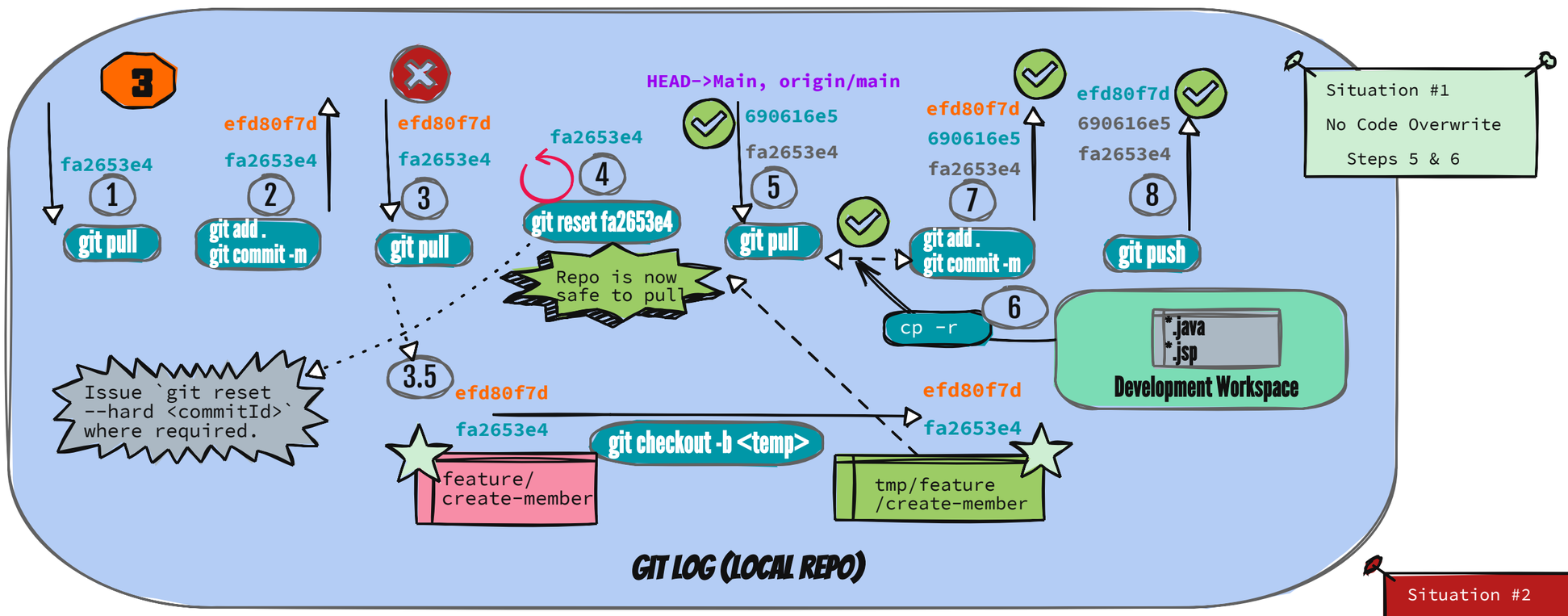

GITHUB - BUGFIX WORKFLOW
Git
Main
FEATURE
PR
GIT MERGE
FEATURE + BUGFIX
Remote Repo
GIT CLONE
GIT --SET-UPSTREAM-TO
GIT PUSH
GIT PULL
Local Repo
BUG FIX
cp -r
*.java
*.jsp
FIXES
Best Practice
YES
NO
AFFECTS OTHERS?
Development Workspace
DEVELOPERS MACHINE (LAPTOP / DESKTOP)

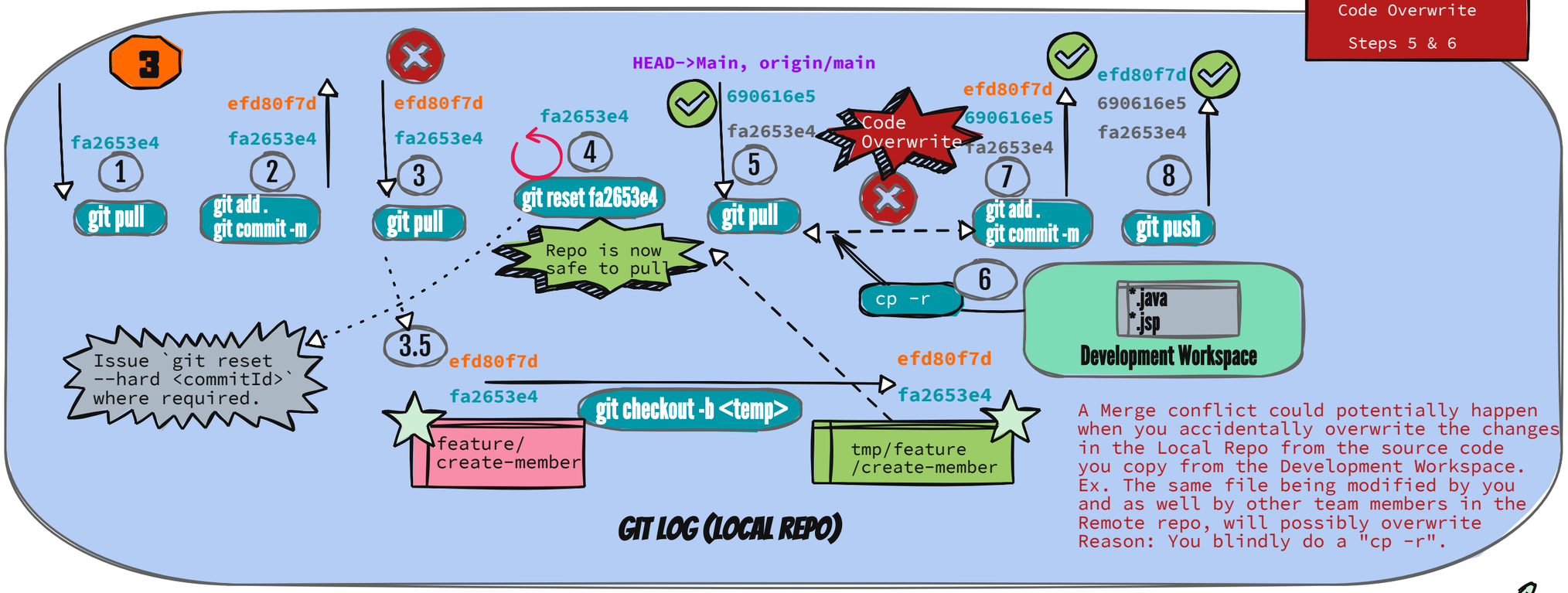

Main
690616e5
fa2653e4
Remote Repo
1
git pull
git push
7
efd80f7d
git add .
git commit -m
6
Main
fa2653e4
2
Local Repo
5
cp -r ..
Baseline
*.java
*.jsp
4
Development Workspace
3

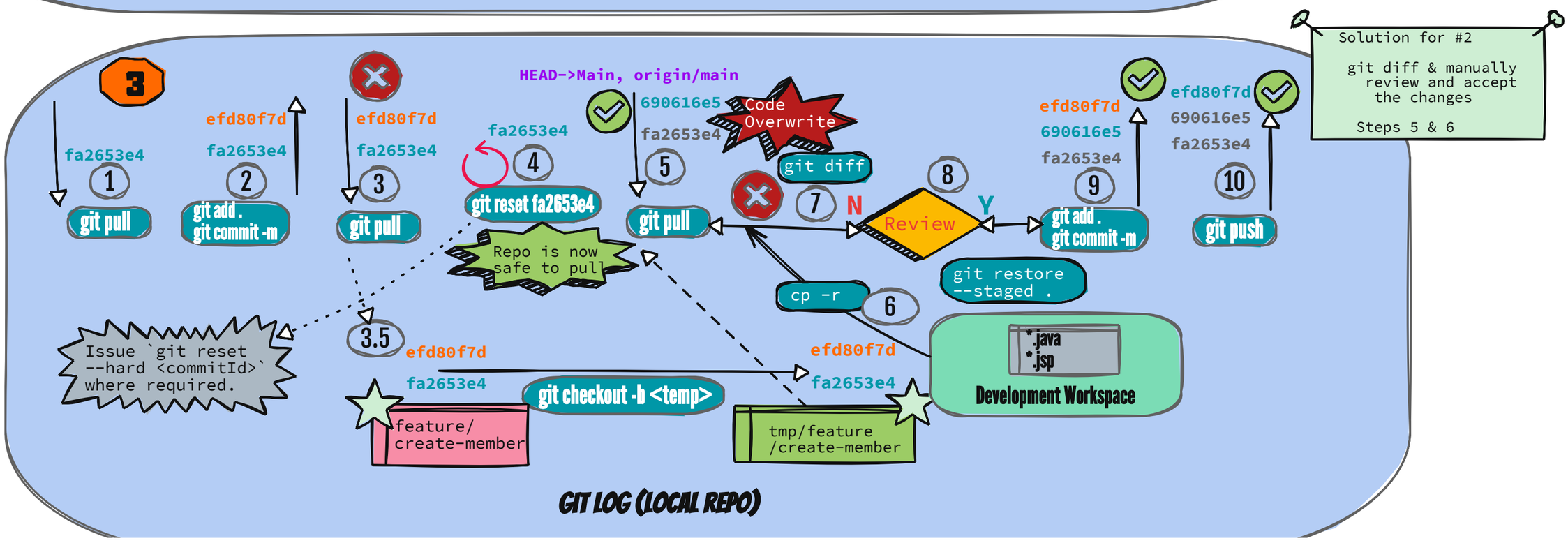
Commit Id / Hash
fa2653e4f2947a614c1d3ba4db77f0a8f53b0005
Commit Hash (Long Form)
fa2653e4
Commit Hash (Short Form)
Min 6 chars


1
fa2653e4
Baseline Commit Hash / Id
efd80f7d
Local/Dev Changes Commit Id
690616e5
Incremental Changes Commit Id
in the Remote - after we pulled it once
2
fa2653e4
Prev Baseline Commit Hash / Id
690616e5
Latest Baseline Commit Hash
efd80f7d
Local/Dev Changes Commit Id

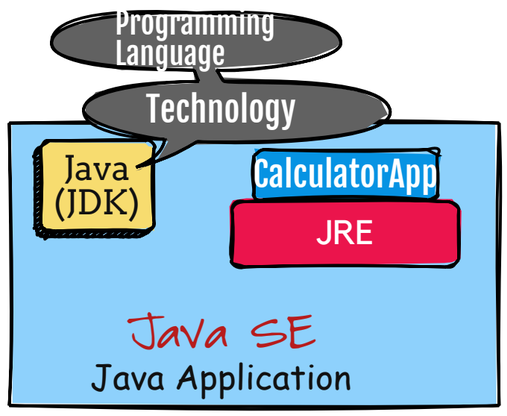

3
fa2653e4
1
git pull
efd80f7d
fa2653e4
2
git add .
git commit -m
efd80f7d
fa2653e4
3
git pull
fa2653e4
4
git reset fa2653e4
Repo is now
safe to pull
HEAD->Main, origin/main
690616e5
fa2653e4
5
git pull
efd80f7d
690616e5
fa2653e4
7
git add .
git commit -m
efd80f7d
690616e5
fa2653e4
8
git push
Situation #1
No Merge Conflicts
Steps 5 & 6
cp -r
6
*.java
*.jsp
Development Workspace
Issue `git reset
--hard <commitId>`
where required.
3.5
efd80f7d
fa2653e4
feature/
create-member
git checkout -b <temp>
efd80f7d
fa2653e4
tmp/feature
/create-member
GIT LOG (LOCAL REPO)

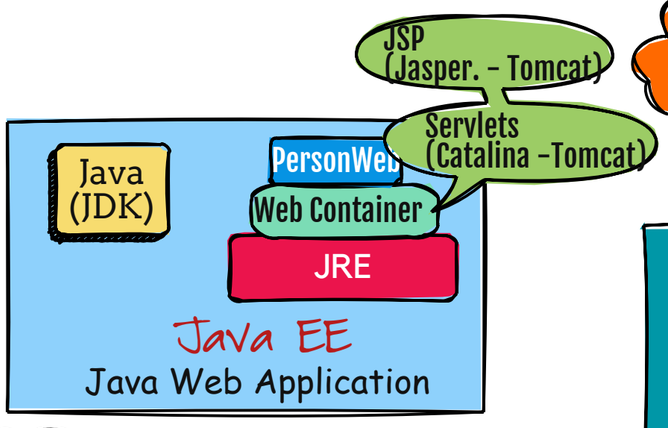

3
fa2653e4
1
git pull
efd80f7d
fa2653e4
2
git add .
git commit -m
efd80f7d
fa2653e4
3
git pull
fa2653e4
4
git reset fa2653e4
Repo is now safe to pull
HEAD->Main, origin/main
690616e5
fa2653e4
5
git pull
efd80f7d
690616e5
fa2653e4
7
git add .
git commit -m
efd80f7d
690616e5
fa2653e4
8
git push
Situation #1
No Code Overwrite
Steps 5 & 6
cp -r
6
*.java
*.jsp
Development Workspace
Issue `git reset --hard <commitId>` where required.
3.5
efd80f7d
fa2653e4
feature/
create-member
git checkout -b <temp>
efd80f7d
fa2653e4
tmp/feature
/create-member
GIT LOG (LOCAL REPO)

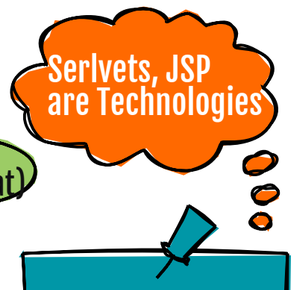

3
fa2653e4
1
git pull
efd80f7d
fa2653e4
2
git add .
git commit -m
efd80f7d
fa2653e4
3
git pull
fa2653e4
4
git reset fa2653e4
Repo is now safe to pull
HEAD->Main, origin/main
690616e5
fa2653e4
5
git pull
Code Overwrite
efd80f7d
690616e5
fa2653e4
7
git add .
git commit -m
efd80f7d
690616e5
fa2653e4
8
git push
Situation #2
Code Overwrite
Steps 5 & 6
cp -r
6
*.java
*.jsp
Development Workspace
Issue `git reset --hard <commitId>` where required.
3.5
efd80f7d
fa2653e4
feature/
create-member
git checkout -b <temp>
efd80f7d
fa2653e4
tmp/feature
/create-member
GIT LOG (LOCAL REPO)
A Merge conflict could potentially happen when you accidentally overwrite the changes in the Local Repo from the source code you copy from the Development Workspace.
Ex. The same file being modified by you and as well by other team members in the Remote repo, will possibly overwrite
Reason: You blindly do a "cp -r".


3
fa2653e4
1
git pull
efd80f7d
fa2653e4
2
git add .
git commit -m
efd80f7d
fa2653e4
3
git pull
fa2653e4
4
git reset fa2653e4
Repo is now safe to pull
HEAD->Main, origin/main
690616e5
fa2653e4
5
git pull
Code Overwrite
git diff
7
N
Review
Y
8
efd80f7d
690616e5
fa2653e4
9
git add .
git commit -m
efd80f7d
690616e5
fa2653e4
10
git push
Solution for #2
git diff & manually review and accept the changes
Steps 5 & 6
git restore --staged .
cp -r
6
*.java
*.jsp
Development Workspace
Issue `git reset --hard <commitId>` where required.
3.5
efd80f7d
fa2653e4
feature/
create-member
git checkout -b <temp>
efd80f7d
fa2653e4
tmp/feature
/create-member
GIT LOG (LOCAL REPO)

Technology Vs Framework

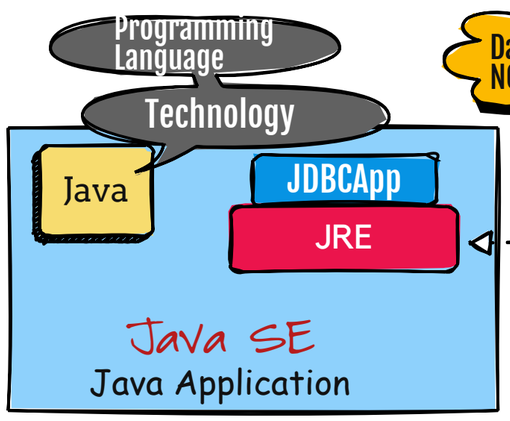


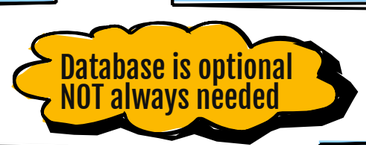


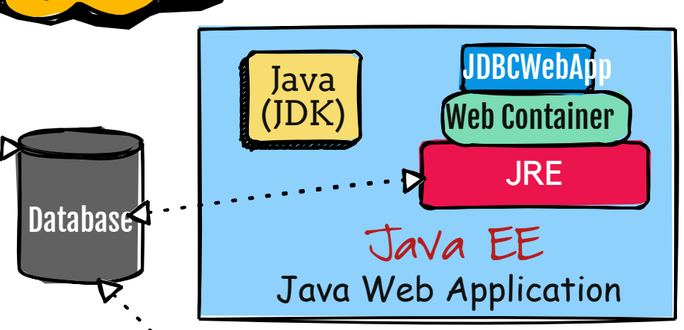


Technology

helps you with the underlying infra with which you prepare the Application



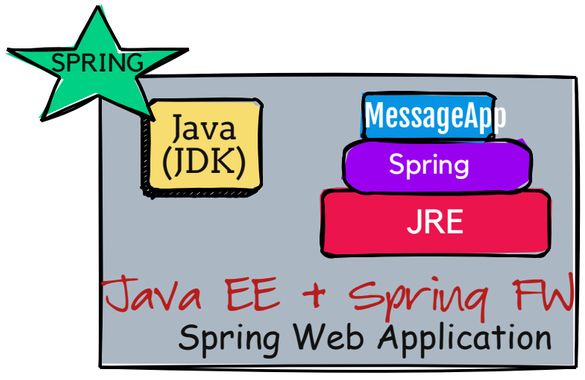


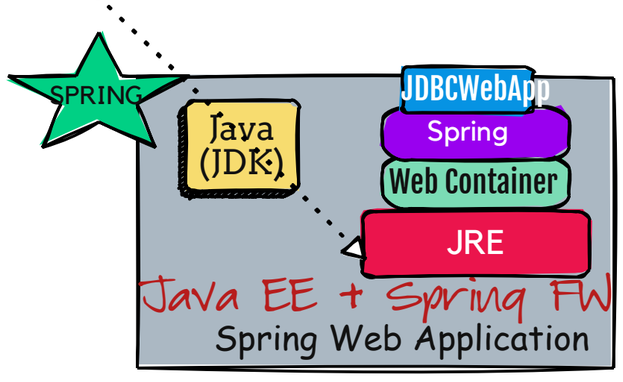


Framework

Like a Ready to Eat food - it is half way prepared for you to take it over.

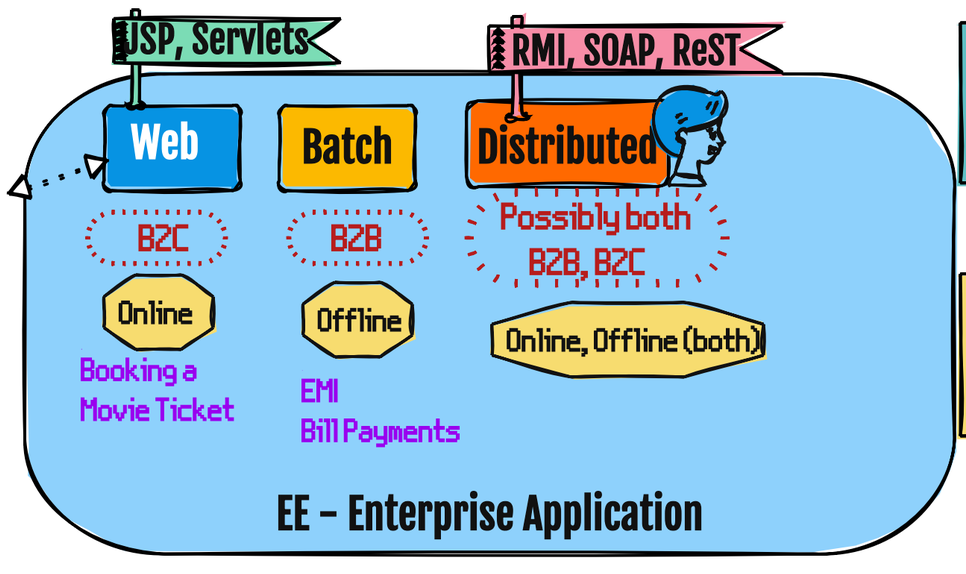
Common Works are taken care by the FW providers.





Frameworks

* Struts
* Tapesty
* Spring
* Play etc.,



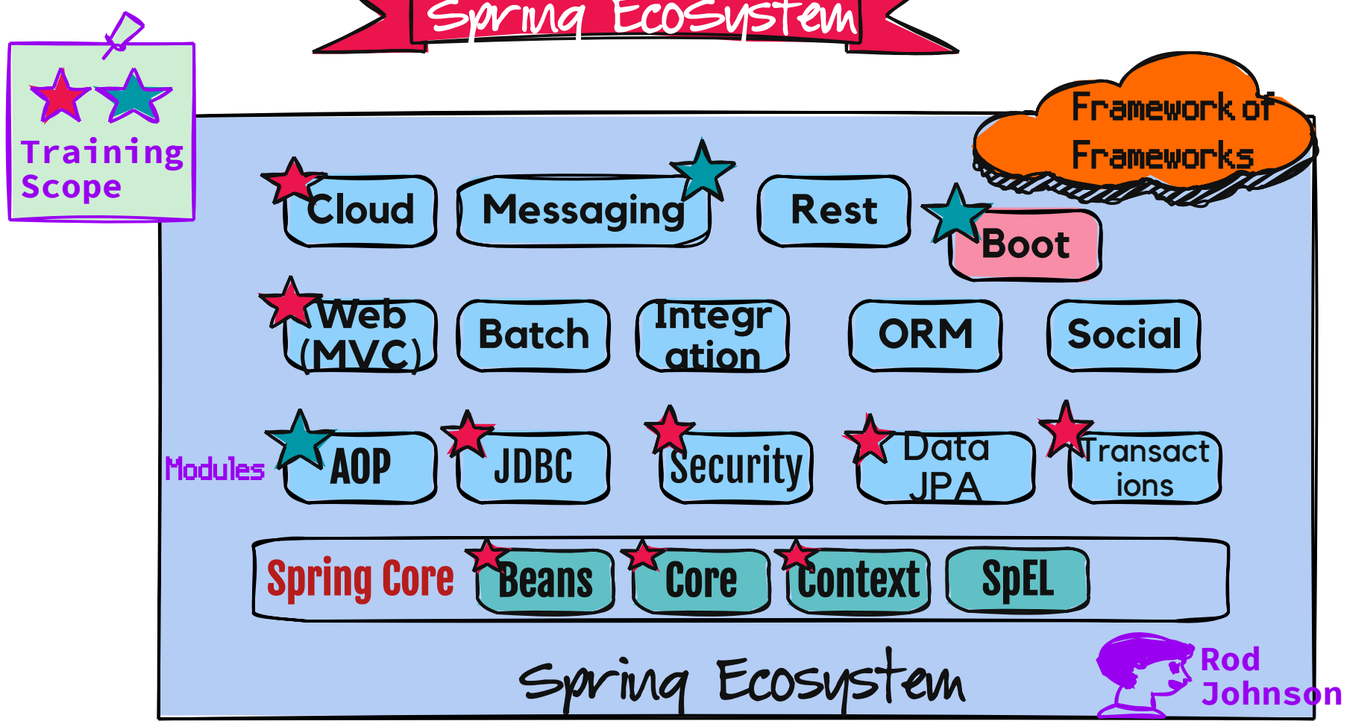
Modes

B2B - Business to Business
B2C - Business to Consumer

Nature

Online - Real time (F2F)
Offline / Batch (Out of Peak hours)

Spring Ecosystem



IoC & DI Principle