Java as Product

* Java Standard Edition [ Java SE] - JDK // JAR
  + Console Application ---
  + Java Libraries
  + Desktop [ GUI ] Application [ Java Swing , Java FX ]
* Java Enterprise Edition [ Java EE] - Java EE SDK
  + Web Application / WAR
  + Enterprise Application //EAR
* Java Micro Edition [ Java ME ]
* Java Card - Store information in smart card chip.

**Java EE 7 Specification**

* **Donated to Eclipse for further enhancement.**

Jakarta EE

1. Web Profile
   1. Servlet
   2. JSP
   3. EJB Lite
   4. JAX-RS
2. Full Profile [ includes Web Profile]
   1. JAX-WS
   2. Messaging API
   3. Java Mail API

Java EE compatible Application Server will supply Java EE SDK

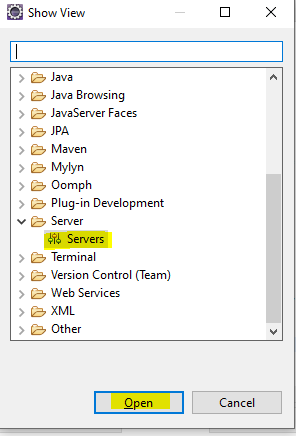
* Oracle WebLogic Server
* Redhat JBoss Server
* IBM Websphere
* Tom EE+ Application Server
* Tomcat Sever

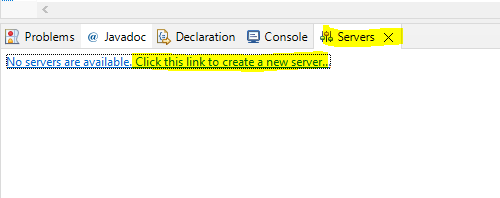
<https://www.apache.org/dyn/closer.cgi/tomee/tomee-8.0.15/apache-tomee-8.0.15-plus.zip>

* Download zip file
* Then extract it to a folder

In eclipse

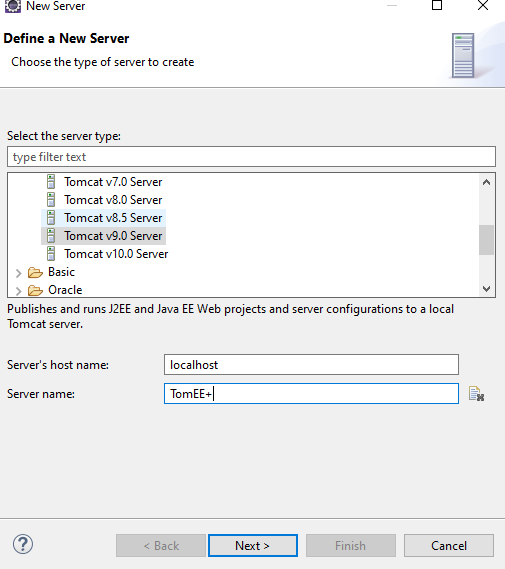
* Window – show view – Other – Server – Open



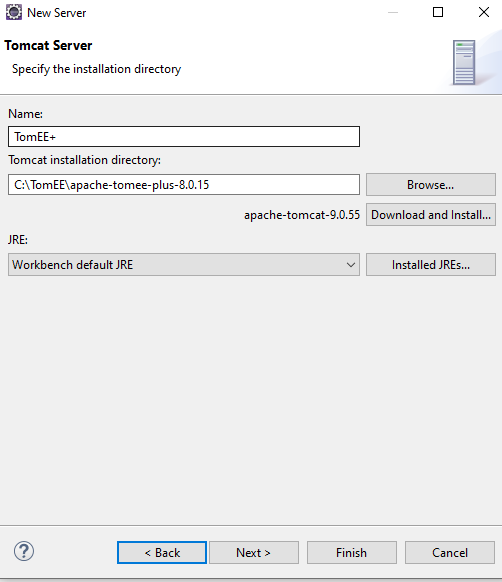
\

Click the link to create a new Server.

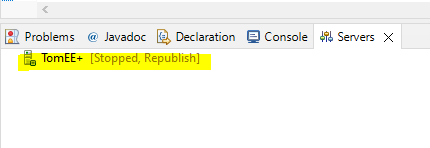
Select Apache Tomcat Sever V 9



Click next



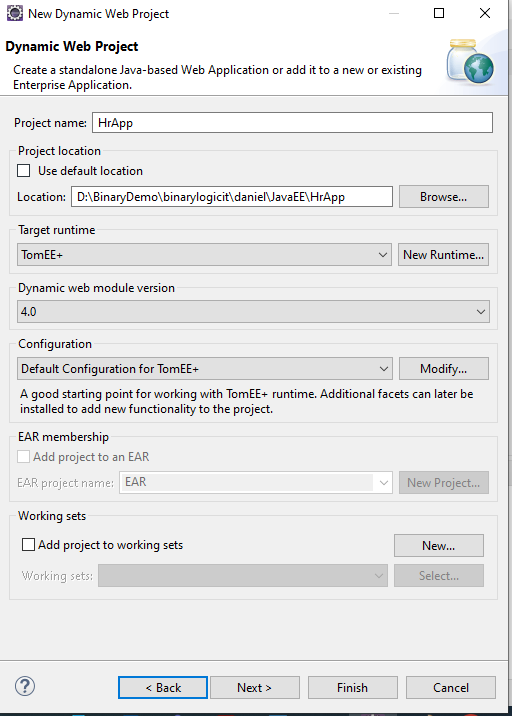
Next – Finish



Right click on TomEE+ and start

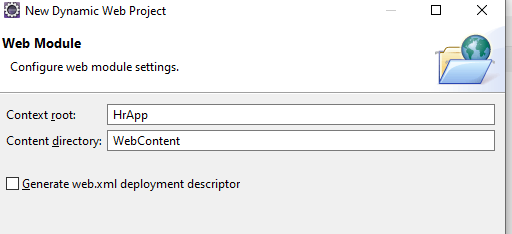
**Creating your First Web Application using Java EE**

* File – New – Other – Web – Dynamic Web Project

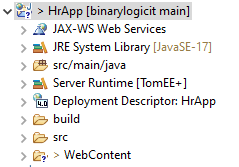


Next and Next Again

Content Director – Change to “WebContent”



Click Finish



M

OTHER JAVA CLASSES

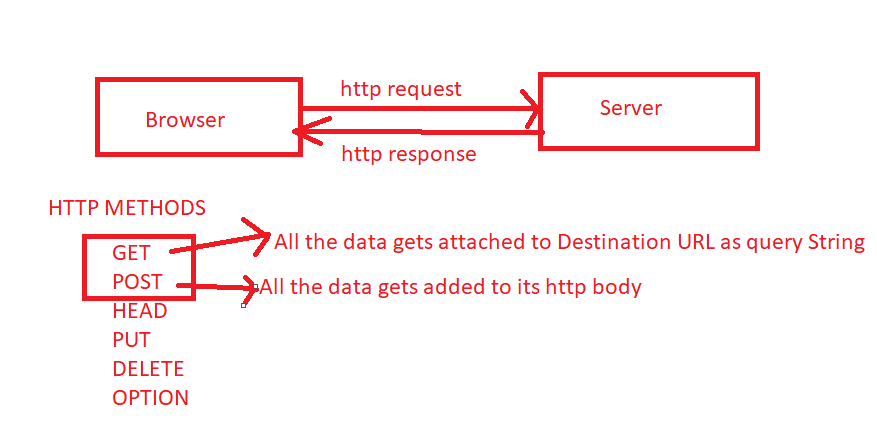
V : View

HTML, JSP

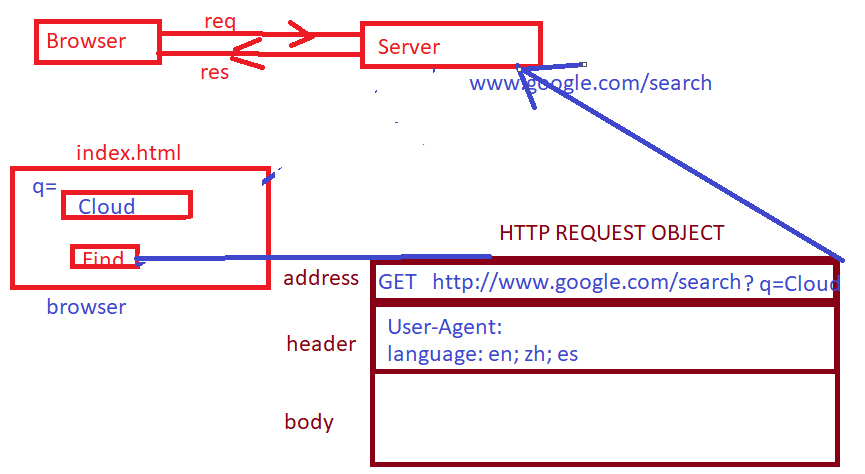
SERVLET

C : CONTROLLER

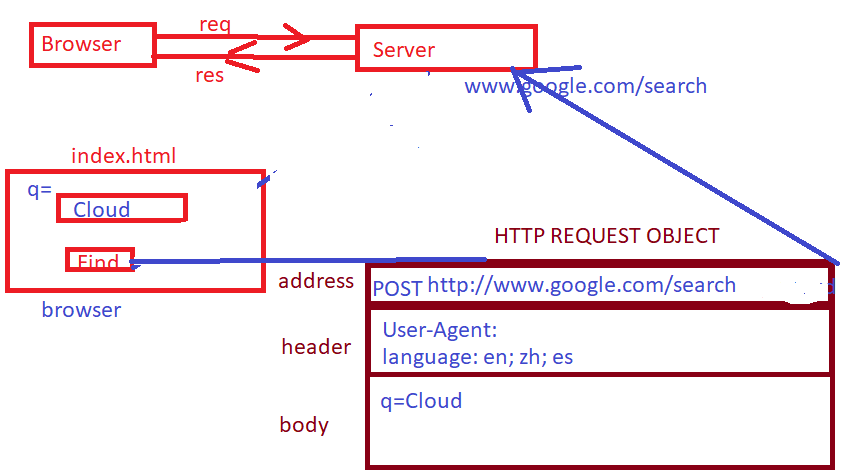
SERVLET

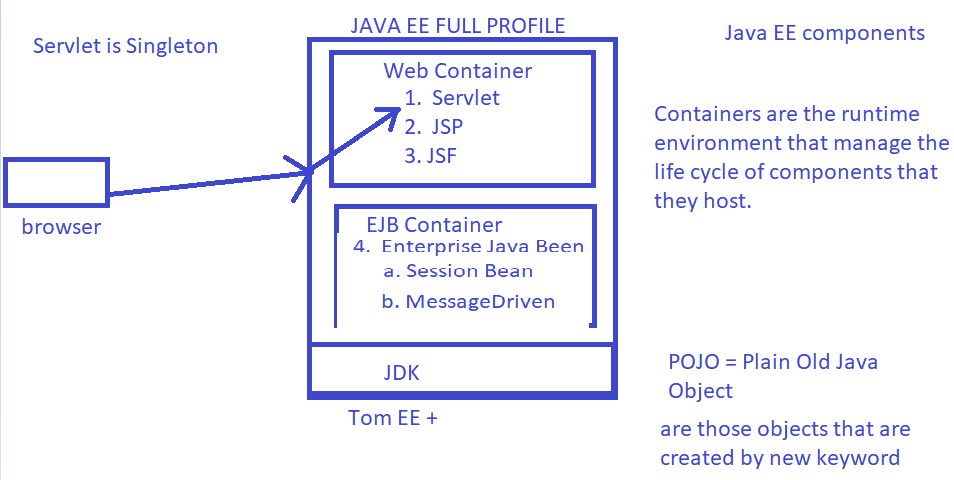


GET REQUEST



**POST**





Servlet Life Cycle

* Instantiation of Servlet is managed by Web Container.
* Only one instance / JVM is created – This means Servlet is Singleton.
* Instance if servlet is created at time when it was requested first.

//invoke only once after instantiation

public void init() { }

// invoke everytime it receives a request

service(ServletRequest req, ServletResponse res){

//doGet(HttpServletRequest req, HttpServletResponse res)

//doPost

}

// only once before servlet gets destroyed

destroy() {

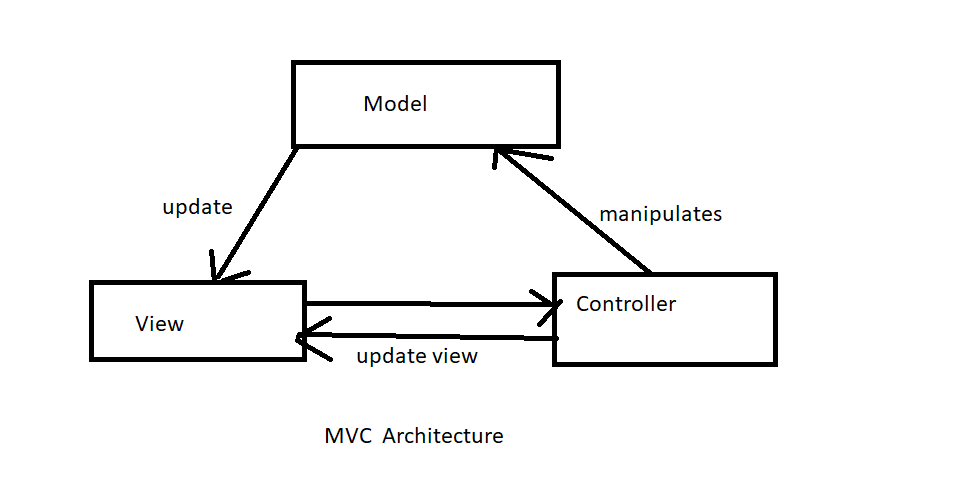
// it is invoked at the time servlet destruction.

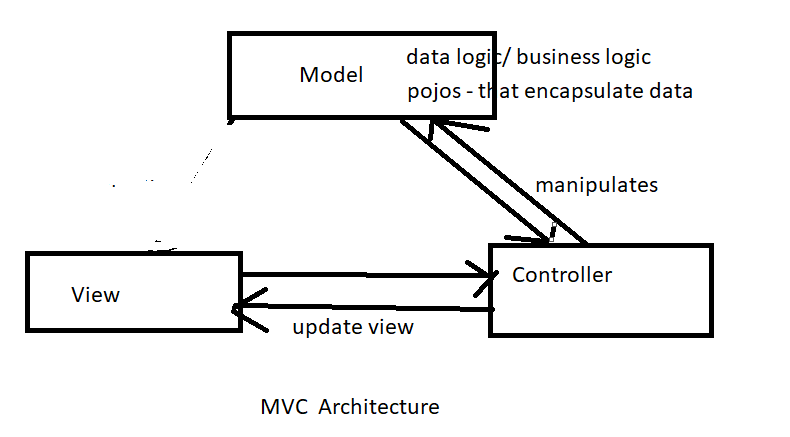
}

SDLC : Software Development Life Cycle

* Requirement Analysis.
* Design
* Coding
  + Unit Testing
* Testing [ QA ]
* Deployment

**Model – View – Controller Architecture**





**Model :** Handle data/business logic

**Controller :** It controls data flow into model object and update view whenever data change. It is responsible for handle request n response.

**View :** It displays the information from the model to the user.

**Java EE Web Application**

Model :

DAO, Pojos

Controller:

Servlet

View :

Servlet

JSP [ popular ]

**RequestDispatcher is used to redirect the call to other components[ view ] from controller.**

RequestDispatcher reqDispatcher=request.getRequestDispatcher("EmployeeUI");

reqDispatcher.forward(request, response);

**How the controller handover the data to View**

It is going to take the help of memory scoping to share the data.

* Request Scope : Explicit to single request – response cycle. Each new request will have its own copy of request & response. Data stored in 1st request will not visible to 2nd request.
* Session Scope : Explicit to User/Client Session – A data maintain in session will be visible to all request – response cycles for same client.
* Application Scope : Explicit to Application - A data maintain in application scope is visible to all users/clients.

**Java Server Page : JSP**

JSP is used to represent view / presentation logic for web application. This is server – side component and this can easily work with Java Objects.

* It is easier to maintain than servlet with respect to complexities that derived by containing embedded HTML
* It supports Expression Language and that helps a lot to simplify JSP page development.

JSP is the extension to servlet. In fact, it is servlet.

**Life Cycle of JSP page**

When use access JSP page first time.

<http://localhost:8080/HrApp/index.jsp>

1. Translation of JSP Page to Index\_Servlet.java by web container.
2. Compilation of .Java
3. ClassLoader loads the class [ .class ]
4. Instantiate an Object of this class // [ only one instance ]
5. init()
6. service()
   1. doGet
   2. doPost
7. destroy()

**Second Time onwards – all the request for JSP Page will start from Step 6**

**JSP provides implicit Objects**

1. **request**
2. **response**
3. **session**
4. **out**