



# NEXA DIGITAL SCHOOL

Java Avancée (JEE – JDBC – JSP)

# Java EE / Jakarta EE



Arnaud SEIGNEUR LAROUZÉE

# Présentation

# Présentation

Java EE a été créé en 1997 par la société Sun Microsystems

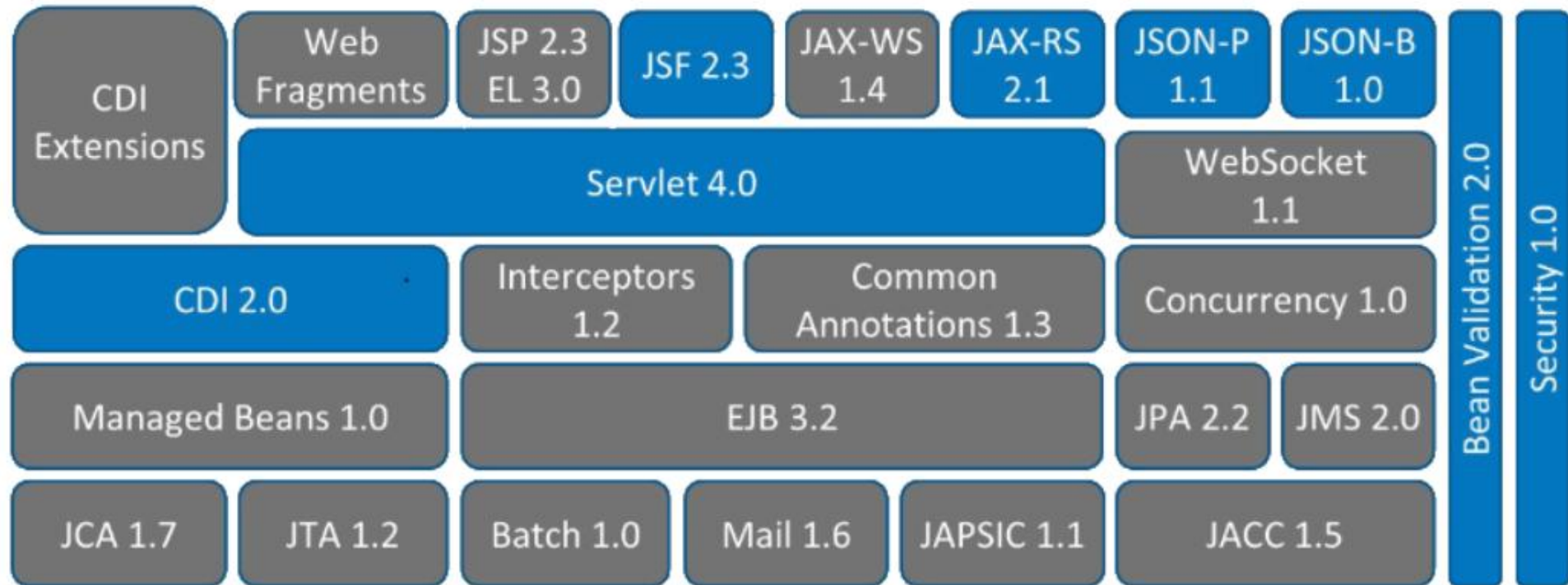
Ajout de nombreuses API au Framework initial Java SE, dans le but de permettre aux entreprises d'avoir une utilisation plus axée « coté serveur », afin de s'adapter au développement du Web.

Il est devenu Jakarta EE en 2018 grâce au choix de la communauté des développeurs suite à la passation du projet de la société Oracle à la Fondation Eclipse.

# Historique

- 1997 : création de [Java Entreprise édition \(Java EE\)](#) par Sun Microsystems
- Décembre 1999 : sortie de [Java EE 1.2](#)
- Septembre 2001 : sortie de [Java EE 1.3](#)
- Novembre 2003 : sortie de [Java EE 1.4](#)
- Mai 2006 : sortie de [Java EE 5](#)
- Avril 2009 : Sun Microsystems est racheté par Oracle Corporation
- Décembre 2009 : sortie [Java EE 6](#)
- Mai 2013 : sortie de [Java EE 7](#)
- Août 2017 : sortie de [Java EE 8](#)
- Octobre 2017 : Oracle confie le projet à [Fondation Oracle](#) qui lance [EE4J](#) pour poursuivre le développement de Java EE
- Janvier 2018 : suite au choix de la communauté [EE4J](#) devient [Jakarta EE](#)
- Décembre 2020 : sortie de [Jakarta EE 9](#)
- Mai 2021 : sortie de [Jakarta EE 9.1](#)
- Septembre 2022 : sortie de [Jakarta EE 10](#) (*version actuelle*)

# Composition



# Composition

Spécification complète de Jakarta EE 10 : [Jakarta.ee/specifications](https://jakarta.ee/specifications)

- Jakarta Bean validation: API de contraintes et de validation au niveau des déclarations (Hibernate / Spring)
- Jakarta Faces:JSF: Framework permettant de créer des interfaces utilisateurs pour des applications web
- Jakarta JSON Binding: Framework permettant de transformer de l'objet Java (POJO) en/depuis du JSON
- Jakarta JSON Processing: API pour analyser, générer, transformer et interroger des documents en JSON
- Jakarta Server Pages:JSP: API permettant de définir le moteur de génération de modèles pour applications Web
- Jakarta Servlet: une API coté serveur qui permet de gérer des requêtes HTTP
- Jakarta SOAP:JAX-RS: protocole d'échange d'information structurée dans l'implémentation de services web bâti sur XML
- Jakarta Persistence:JPA: API permettant aux développeurs d'organiser des données relationnelles dans des applications java (Hibernate )

# Installations pré requises

Pour ce cours j'utiliserai Tomcat 11 et l'IDE de la fondation Oracle « Eclipse » :

## Installation Tomcat 11

- DLL du fichier zip correspondant à l'OS utilisé ( ici 64-bit Windows zip)

[Tomcat download-11](#)

- Core:
  - [zip](#) (pgp, sha512)
  - [tar.gz](#) (pgp, sha512)
  - [Windows zip](#) (pgp, sha512)
  - [Windows Service Installer](#) (pgp, sha512)

## Installation de l'IDE Eclipse

- DLL du fichier d'installation d'Eclipse IDE

[Eclipse IDE for Enterprise Java and Web Developers](#)

### Download Links

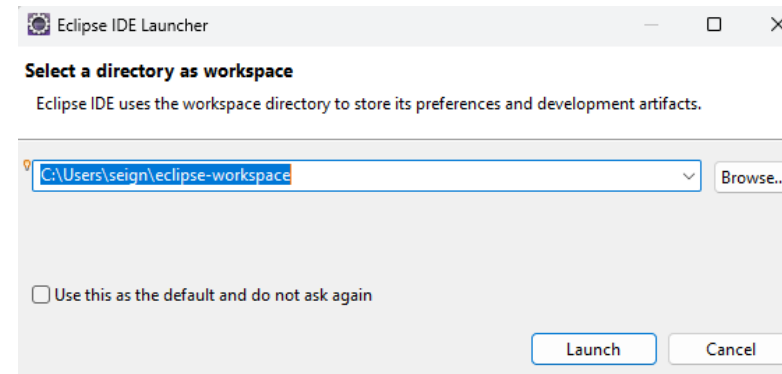
Windows  
x86\_64 | AArch64  
macOS  
x86\_64 | AArch64  
Linux  
x86\_64 | AArch64 | riscv64



# Installation Eclipse IDE

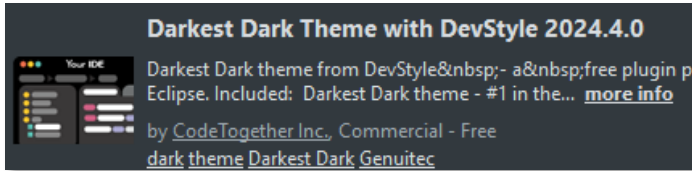
Une fois le fichier correspondant a votre OS téléchargé dézippez le puis sélectionnez Eclipse .Exe.

Lors de l'installation vous sera demandé de choisir l'emplacement de votre Workspace ( C'est le fichier qui regroupera tout vos fichier de Projets, il est donc TRÈS IMPORTANT de bien noter ou connaître son emplacement.



# Installation Eclipse IDE

Vous pouvez changer l'aspect de votre IDE en mode sombre cela dit il en existe un autre plus commun et visuellement acceptable ( pour moi) voici comment l'installer.

- Votre IDE ouvert allez sur **Help** (onglet tout à droite) puis sur **Eclipse Marketplace**
- Lancez le temps a Eclipse de charger tout ce qui doit l'être, une fois que la liste est chargée tapez dans la recherche **dark** puis **GO**
- Sélectionnez 
- Il vous sera proposé d'installer également CodeTogether permettant de coder à plusieurs sur le meme IDE je ne le conseil pas spécialement
- Une fois l'installation faite relancer l'IDE une pop up supplémentaire apparaîtra vous demandant quel thème vous souhaitez installer, sélectionnez-le.
- Finaliser le lancement de l'IDE, le pop-up n'apparaîtra plus lors des prochains lancements.

# Servlet

Technologie Java utilisée pour effectuer des traitements coté serveur en réponse aux requêtes provenant en général de poste clients distants.

Bien que les Servlets puissent répondre à n'importe quel type de requête, elles sont généralement employées pour répondre à des requêtes de type HTTP et qui permettent de retourner dynamiquement des pages HTML.

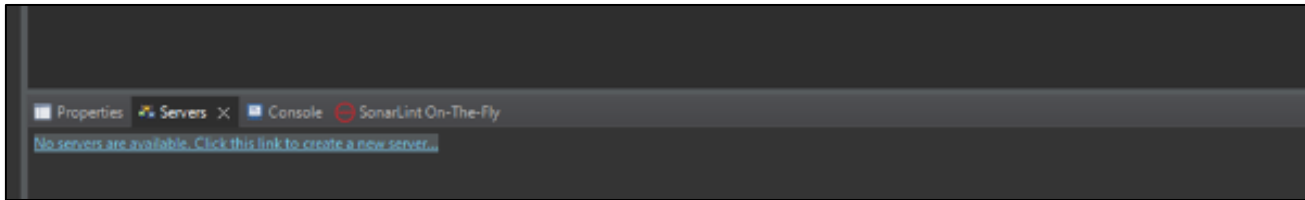
<https://jakarta.ee/specifications/servlet/4.0/apidocs/>

Cette technologie fonctionne en binôme avec des « conteneurs » comme Jetty ou Tomcat.

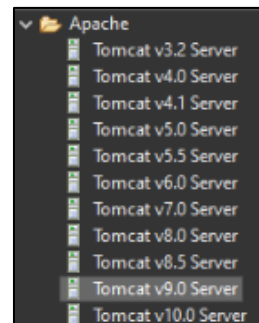
# Installation Tomcat

Pour installer Tomcat avec Eclipse, il faut :

1. Lancez Eclipse
2. Dans l'onglet Servers à gauche de la console cliquez sur le lien de création d'un nouveau serveur

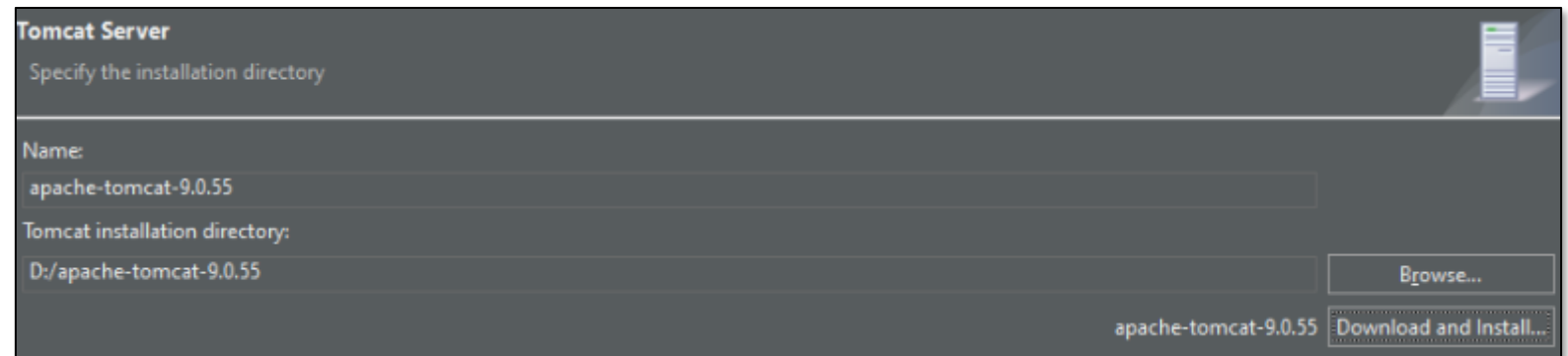


3. Cliquez sur le dossier Apache => sélectionnez la version Tomcat désirée  
(Ici Tomcat 9.0 Server)

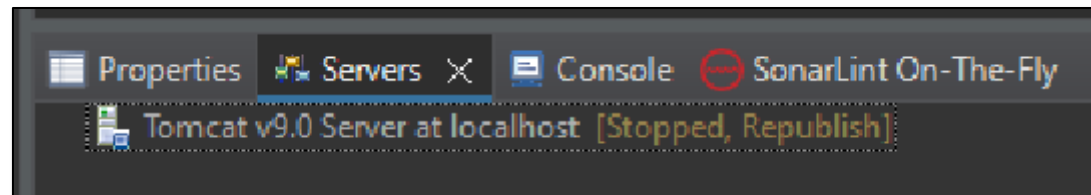


# Installation Tomcat

4. Cliquez sur Next puis sélectionner le dossier d'installation



5. Cliquez sur Next et enfin Finish
6. Vous devriez voir apparaître ceci dans l'onglet serveur



# Request / Response

Un Servlet fonctionne sur la base d'envois (request) et de réception (response) de HTTP

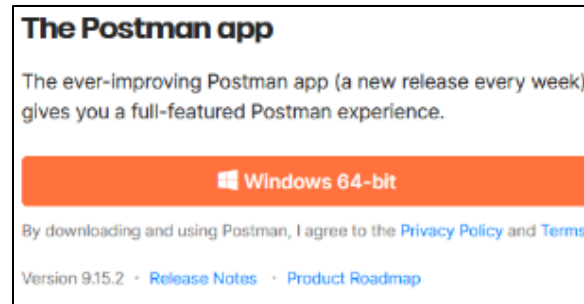
Les request les plus utilisées sont :

- Get: récupère les informations.
- Post: demande d'ajout d'information pas encore existante sur le serveur.
- Put: modification complète de l'objet ( modifie tous les paramètres de l'objet concerné).
- Patch: Modification partielle de l'objet.
- Delete: demande la suppression complète de l'objet.
- Copy: demande a copier totalement une donnée déjà existante.
- Head: demande des informations sur la ressource ciblée.
- Options: demande la liste des manière autorisées de communiquer avec le serveur.

# Request / Response

Afin de mettre en pratique simplement les requêtes

Vous pouvez par exemple Télécharger Postman : [Postman](#)



En test de fonctionnement de cette application vous pouvez également

Ouvrir **Postman** puis choisir **Get** et saisir

<https://api.chucknorris.io/jokes/random>

Puis **Send**

# Request / Response

Lors de la communication avec le serveur la Response contient un code pouvant être parfois visible ou implicite du fait de sa récurrence ( par exemple 200/304 )

Structure des codes :

- 1er chiffre renvoi à la catégorie de réponse :
  - *Information (1xx)*
  - *Succès (2xx)*
  - *Redirection (3xx)*
  - *Erreur Client (4xx)*
  - *Erreur Serveur (5xx)*



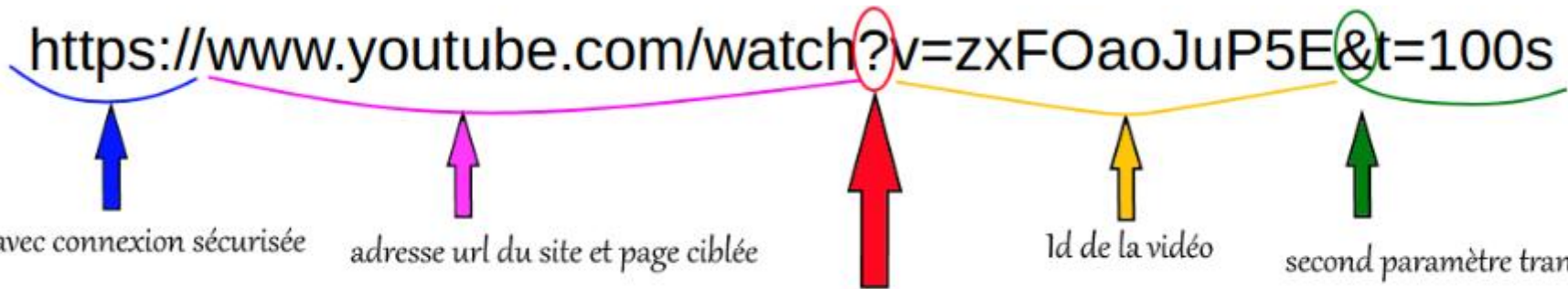
# Request / Response

Structure des codes (suite)

- Les 2 et 3èmes chiffres correspondent au code de l'erreur :
  - *101 : acceptation du changement de protocole*
  - *204 : requête traitée avec succès mais sans information à renvoyer*
  - *310 : la requête est victime de trop de redirection ou d'une boucle de redirection*
  - *404 : ressource non trouvée*
  - *505 : Version HTTP non gérée par le serveur*

# URL

Qu'est ce qu'un URL ?

- <https://www.youtube.com/watch?v=zxFOaoJuP5E&t=100s>
- 
- The diagram shows the URL `https://www.youtube.com/watch?v=zxFOaoJuP5E&t=100s` with several annotations:
- A blue bracket under `https://` points to the text "Site avec connexion sécurisée".
  - A pink bracket under `www.youtube.com/watch` points to the text "adresse url du site et page ciblée".
  - A red circle around the `?` character has a red arrow pointing to it from below, with the text "après ce symbole ce trouvent les paramètres" below the arrow.
  - A yellow bracket under `v=zxFOaoJuP5E` points to the text "Id de la vidéo".
  - A green bracket under `&t=100s` points to the text "second paramètre transmis dans l'URL".

# URL

Une requête communique au travers d'une URL (ici les paramètres font partis de la requête envoyée au serveur de YouTube afin qu'il affiche une certaine vidéo à un temps donné.)

Test de manipulation d'une URL =>

<https://www.clelia.fr/MireAleatoireParametrable/index.jsp?nbLignes=40&nbColonnes=40>