HttpServlet

public void init(ServletCfg)
{
    // most imp code
} init();   ?

extends →

MyServlet

public void init(ServletCfg)
{
    // conn open
}   super.init(cfg);

HttpServlet
=============
public void init(ServletConfig cfg)throws....
{
    // ✓ impt code  ✓          ✓
    init();
}
public void init() throws.....
{
    // blank
}

MyServlet
=============
public void init() throws....
{
    // conn open code
}

Database communication in Servlet
==============================
1. Java allows us to communicate with databases in our Servlet because almost every Java web application requires the support of DB for performing its activity.

2. Now since, we have a Servlet running at the backend, it is the Servlet developers responsibility to **receive the required data**, **communicate with the DB**, **fetch the proper result** and **finally return it back as an HTML response** to the browser.

Steps required for DB communication in Servlet
====================================
1. Whenever we write a Servlet which is going to interact with the DB then majorly we take 3 steps:

a. **Opening the DB connection** : Required only once.
b. **Processing the request** : Required for every request.
c. **Disconnecting with the DB** : Required only once.

2. For performing **step b** we can override **doGet()** or **doPost()** method since they are executed on every request.

3. For performing **step c** we can override **destroy()** method since it is called only once when the **Server stops** or site is **undeployed.**

4. For **step a** which also needs to be done only once we have two options:

   a) We can override **init(ServletConfig)** method because it is executed only once on first request. But we must remember that the **init(ServletConfig)** method has a very impt definition or body defined in **HttpServlet** class and if we override it, that body will not get a chance to run.

 b) Java proposes two solutions to this problem:
    i) We can call parent class **init(ServletConfig)** from our Servlet's **init(ServletConfig)** using super keyword as show below:

   **super.init(scfg);**

   ii) There is one more init() in HttpServlet class which is non parameterized and has a blank body. Following is the prototype of this inti() in HttpServlet class:

   **public void init() throws ServletException;**

Moreover the parameterized version of **init(ServletConfig)** defined inside **HttpServlet** class automatically calls the non parameterized version of **init()**. So instead of overriding **init(ServletConfig)**, we can override **init()**. In this case without using **super** keyword both the **init()** methods will run.

The container will call parameterized version **init(ServletConfig)** which will excute **HttpServlet's init (ServletConfig)** and from this **init(ServletConfig)**, non parameterized **init()** will be called which will run our **init()**