2 doGet

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        System.out.println("--------inside like GET---");

        System.out.println((Integer.parseInt(request.getParameter("postId"))));

}


protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
```

Servlet code to download the JAR

Servlet code to download the JAR

```
// a bunch of imports here

public class CodeReturn extends HttpServlet {

   public void doGet(HttpServletRequest request, HttpServletResponse response)
                                          throws IOException, ServletException {

      response.setContentType("application/jar");

      ServletContext ctx = getServletContext();
      InputStream is = ctx.getResourceAsStream("/bookCode.jar");

      int read = 0;
      byte[] bytes = new byte[1024];

      OutputStream os = response.getOutputStream();
      while ((read = is.read(bytes)) != -1) {
         os.write(bytes, 0, read);
      }
      os.flush();
      os.close();
   }
}
```

We want the browser to recognize that this is a JAR, not HTML, so we set the content type to "application/jar".

This just says, "give me an input stream for the resource named bookCode.jar".

Here's the key part, but it's just plain old I/O!! Nothing special, just read the JAR bytes, then write the bytes to the output stream that we get from the response object.

---

http://www.avajava.com/tutorials/lessons/how-do-i-serve-up-a-pdf-from-a-servlet.html