

# Capítulo 3



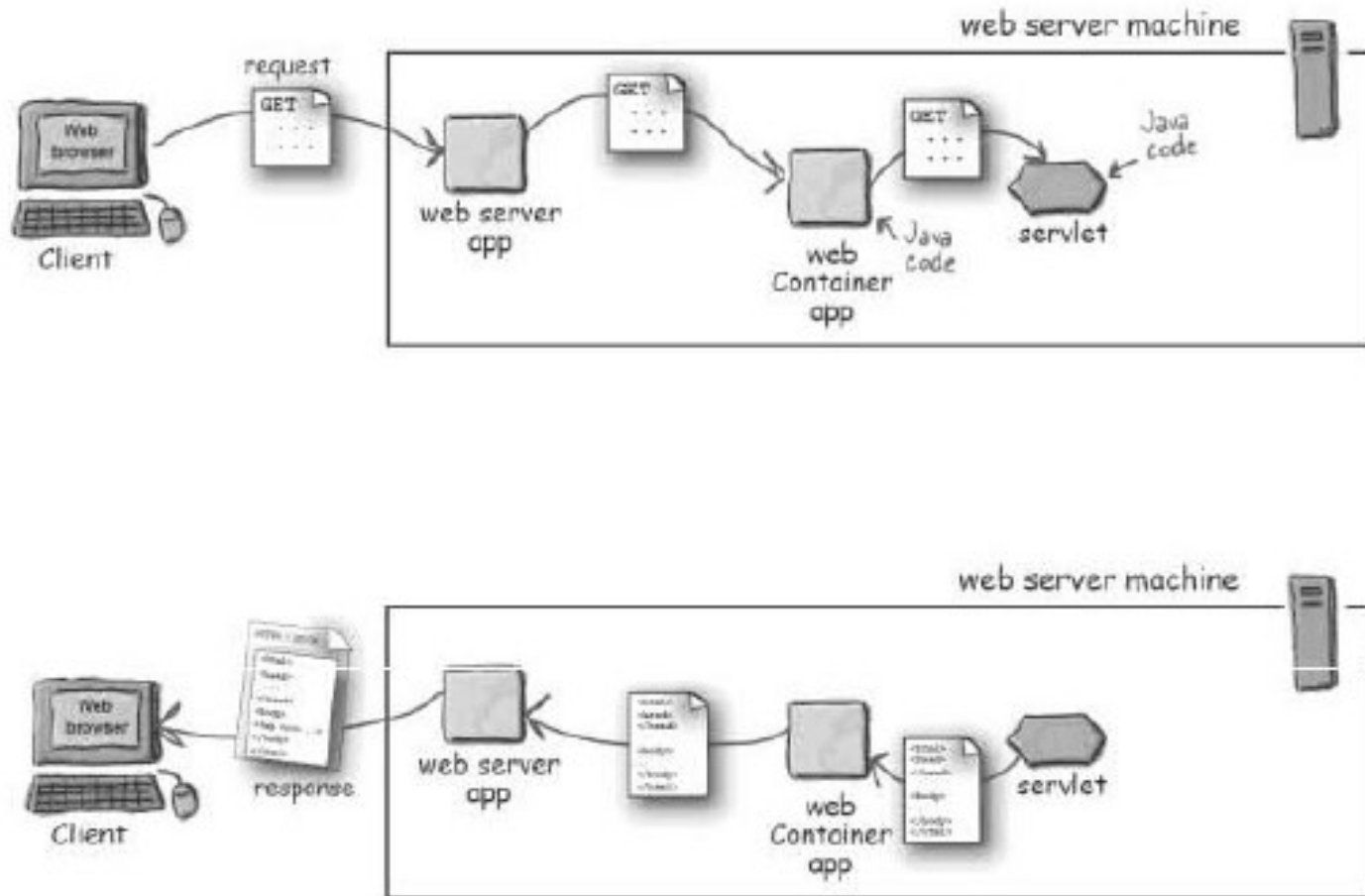
## Contenedor de Servlet

# Contenedor



- Los servlets no tienen un método main()
  - Están bajo control de otra aplicación, llamada contenedor
  - Servidor web recibe un requerimiento por un servlet
    - Se reenvía el requerimiento hacia el contenedor que tiene implementado el servlet
  - El contenedor entrega al servlet el HTTP response y HTTP request
  - El contenedor invoca los métodos de lo servlet

# Contenedor



# Contenedor



- Lo que provee el contenedor:
  - Soporte de comunicaciones
    - Provee el camino para que los servlets se comuniquen con el servidor web
    - No es necesario construir ServerSocket, escuchar en un puerto, crear stream de datos, etc
    - El contenedor se preocupa del protocolo entre él y el servidor web
    - Los servlet solo se preocupan de su lógica de negocio
  - Gestión ciclo de vida de servlets
    - Cuida la carga de clases, inicialización e instanciación de los servlets
    - Invocación de métodos y deja las instancias disponibles para el garbage collector

# Contenedor

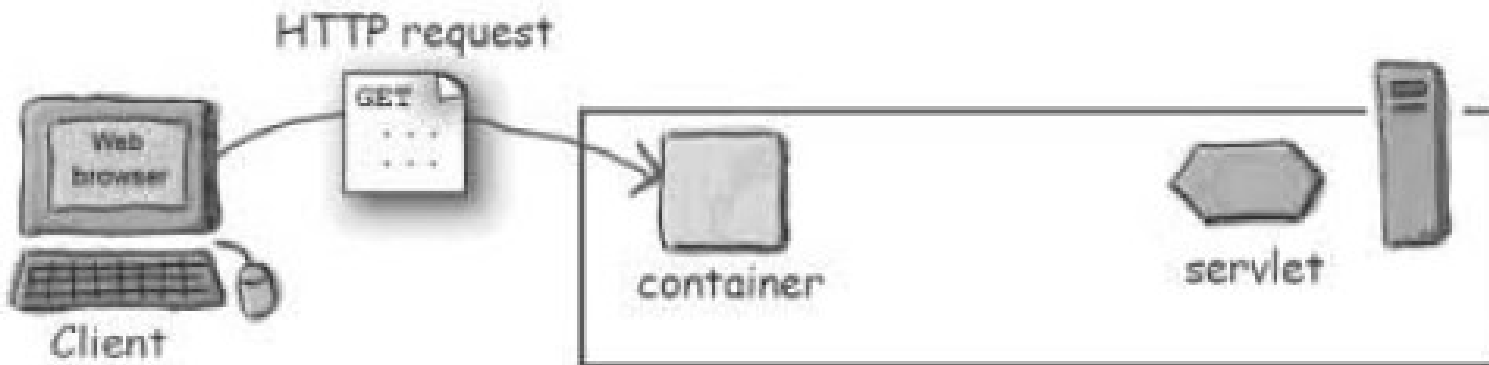


- Lo que provee el contenedor:
  - Soporte Multithread
    - Crea automáticamente un thread para cada requerimiento que recibe para un servlet
    - Cuando el servlet terminó de ejecutar el método que pidió el requerimiento, el thread termina (muere)
  - Seguridad declarativa
    - Se utiliza un descriptor XML para configurar y modificar la seguridad
    - No es necesario codificarla en el servlet
    - Se puede gestionar la seguridad sin la necesidad de estar recompilando las clases
  - Soporte JSP
    - El contenedor se encarga de convertir los JSP en código Java

# Contenedor



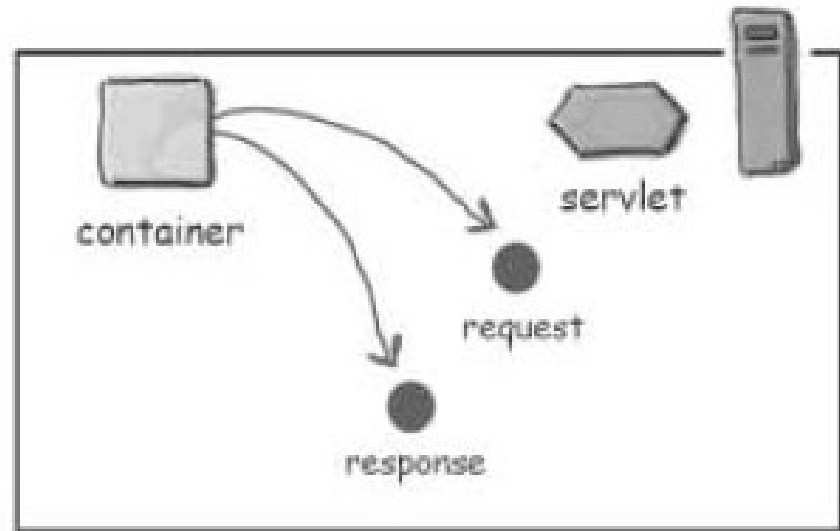
- Como se maneja un requerimiento
  - Un usuario hace click en un link
  - La url apunta a un servlet en vez de una página web estática



# Contenedor



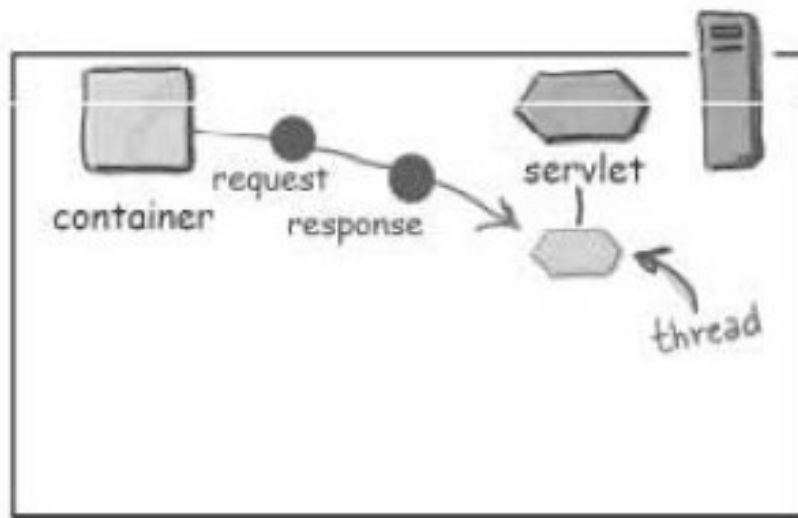
- El contenedor ve que el requerimiento es para un servlet
- Crea dos objetos: HttpServletResponse y HttpServletRequest



# Contenedor



- El contenedor encuentra el Servlet asociado a la URL del requerimiento
- Crea un nuevo thread para el requerimiento
- Entrega el request y response creados al thread

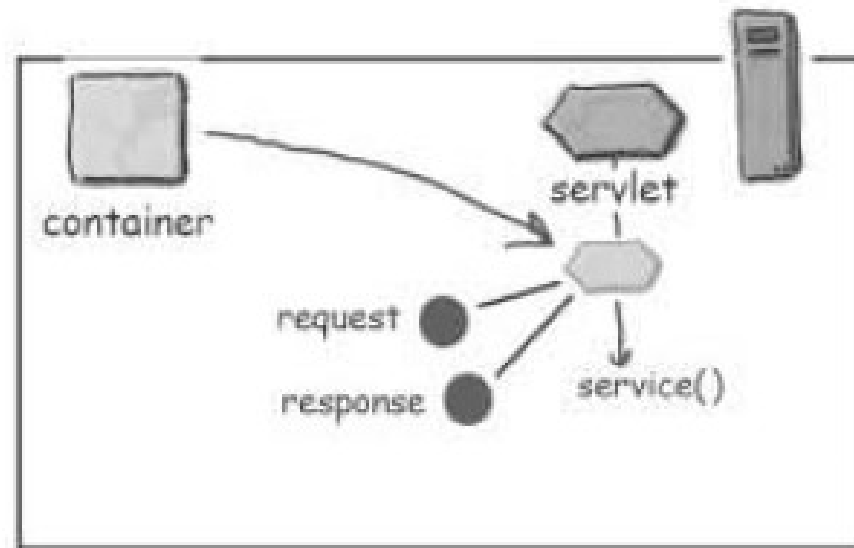




# Contenedor



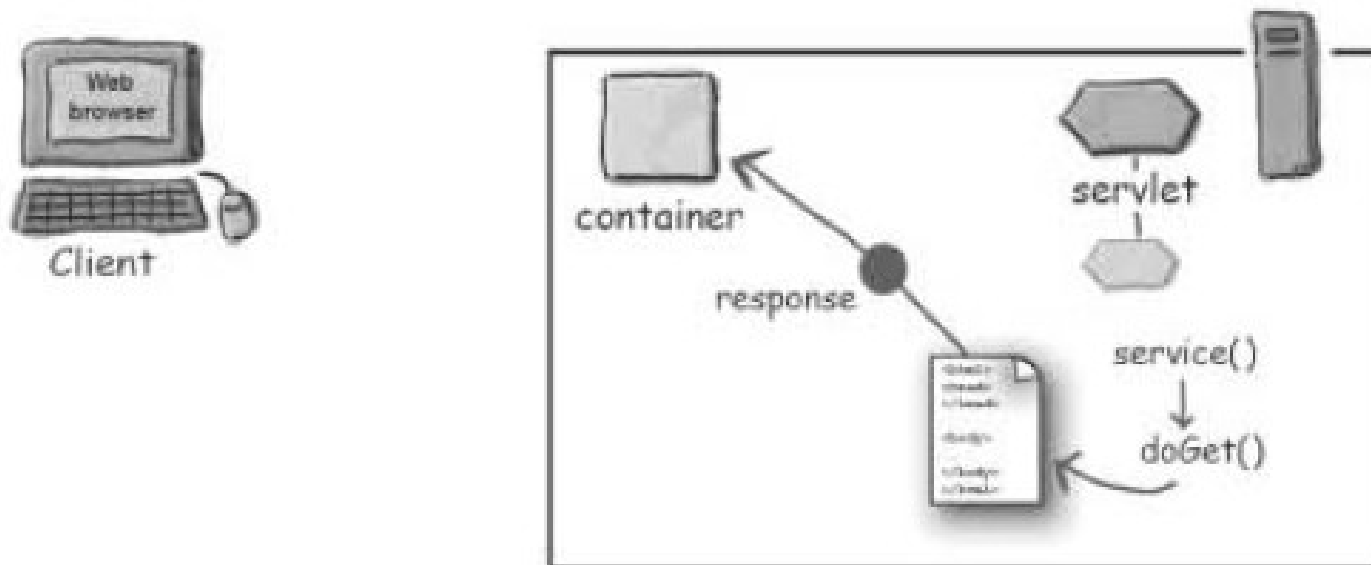
- El contenedor invoca el método “service” del servlet
- Dependiendo del tipo del request, “service” puede invocar a doGet o doPost
- Para este caso el request es un HTTP GET



# Contenedor



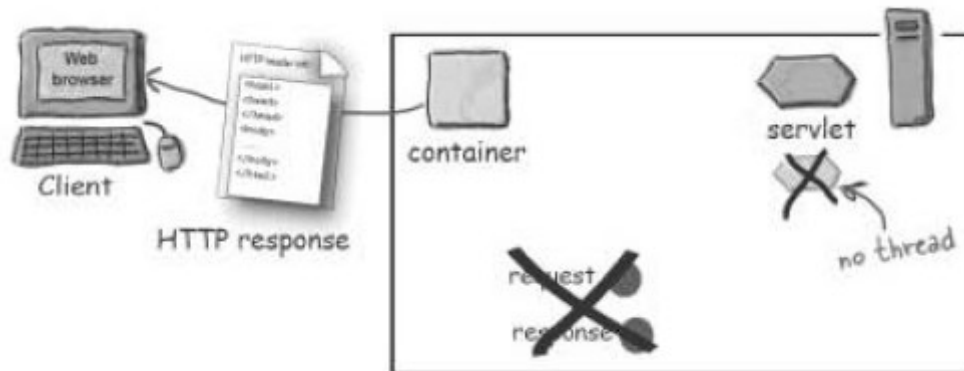
- Método doGet genera dinámicamente la página y cosas en el objeto “response”
- El contenedor sigue manteniendo una referencia al objeto response



# Contenedor



- El thread finaliza
- El contenedor convierte el objeto “response” en un HTTP response y lo envía al cliente
- Elimina los objetos request y response



# Contenedor



- Como el contenedor encuentra el servlet?
  - La URL que viene en el request se asocia a un servlet específico en el servidor
  - Un servlet puede tener 3 nombres:
    - Nombre de la clase: lo asigna el creador del servlet y considera el package dentro del cual está desarrollado
    - Deployment name: nombre interno secreto que no tiene por que ser igual al nombre de la clase
    - Nombre público: el nombre que el cliente conoce, que está codificado en el HTML sobre el que hace click
      - Este nombre público se envia al servidor por medio del requerimiento HTTP
  - Asignación de nombres de servlet mejoran la flexibilidad y seguridad

# Contenedor



- DD: Deployment Descriptor
  - Cuando se implementa un servlet en un contenedor web se debe crear un DD
    - Se indica al contenedor como debe ejecutar los servlets y JSP
  - Se usan dos elementos para asignar URL a servlets
    - <servlet-mapping>: asocia el nombre interno con la URL pública
    - <servlet>: asocia el nombre interno con el nombre de la clase del servlet
  - Además se puede usar el DD para:
    - definir roles de seguridad, páginas de error
    - Bibliotecas, información de configuración inicial
  - Permite cambiar las aplicación sin modificar el código