# Diagram 1: HTTP Communication Basics

**Web Clients**

**Web Server**

HTTP is request/response -based, stateless communication between clients and servers.  
Requests and responses carry information in their headers and body.

|  |  |  |
| --- | --- | --- |
|  | HTTP Request | HTTP Response |
| Headers | URL, HTTP Method, etc. | Status code, content type (MIME type), etc. |
| Body | User-submitted data  (Form parameters) | Content |

Server accepts request on a specific port,   
then sends an  
**HTTP Response** back to the client…

Client sends an **HTTP Request** for a resource on the Server (from a browser, for example)…

# Diagram 2: Requests for Dynamic Content

The Web Server application handles HTTP communication, but can only respond with **static resources**.   
  
Static resources are the same every time they are served to a client. For **dynamic functionality,** the help of another application is needed.

The **Web Container** can take the information in the request, run some Java code, talk to a database, and generate a response. The Web Server can now serve it back to the client, just like a static resource.

**Web Container  
(Application)**

**Web Server   
(Application)**

**Static   
Resources**

**file  
.pdf**

**page  
.html**

**image  
.jpg**

**Web Server (Machine)**



**Web Client**

Request for dynamic   
content is forwarded

**Database**

**custom  
.html**

# Diagram 3: Servlet Basics

The **Deployment Descriptor** (web.xml) contains the system’s configuration – all the Servlets and their URL mappings, initialization information, and more.

**Web Server (Machine)**

**Web Container (Application)**

**Servlet B**  
init()  
service()  
doGet()  
…   
destroy()

**web.xml**

1. The Web Container refers to DD to see which particular Servlet the request URL is mapped to
2. A new HTTP request is   
   given to the web container

**Servlet A**  
init()  
service()  
doGet()  
…  
destroy()

1. The Web Container calls the Servlet’s service() method in a **new thread**

**Web Server   
(Application)**

1. The Servlet generates a response page
2. The response is sent back to the client

# Diagram 4: Request Lifecycle Details

**Web Server (Machine)**

**Web Container (Application)**

1. A new HTTP request is given to the web container
2. The Web Container refers to DD to determine which Servlet the request URL is mapped to

**web.xml**

**Web Server   
(Application)**

1. The container instantiates one **HttpServletRequest** and one **HttpServletResponse** object to represent this HTTP request
2. The Web Container calls the Servlet’s **service()** method in a **new thread,** passing to it the request/response objects
3. The response is sent back to the client

**Servlet A**  
init()  
service()  
doGet()  
…  
destroy()

1. **service()** calls the appropriate **do\_\_\_()** method, based on the HTTP method in the request header, passing to it the request/response objects.
2. The invoked **do\_\_\_()** method generates a response (***or*** forwards the request/response objects to another Servlet – back to step 4)

# Diagram 5: Web Container Lifecycle Details

**Web Server (Machine)**

**Web Container (Application)**

1. The Web Container starts up.

**Web App**

1. The Web Container creates a **ServletContext** object to hold config data for the entire application. It is accessible from ALL application components.
2. This object is loaded with context initialization parameters from the DD.
3. The Web Container refers to DD for initialization information.

**Web App**

**Servlet  
Context**

**web.xml**

**web.xml**

**SC**

**Servlet A**  
init()  
service()  
doGet()  
…  
destroy()

1. The Container creates one instance of each servlet, either at start-up or upon the first request addressed to it. \*
2. The **init()** method is called on each Servlet instance. **destroy()** will be called when the instance is destroyed.

**Web App**

**Servlet B**

**SC**

**web.xml**

**Servlet C**

\* Each Container has a default (lazy vs. eager instantiation),   
but this can be set manually for individual servlets.

# Diagram 6: Filters

1. Container checks the DD to see which Servlet and Filters should process this request, as well as the order

**web.xml**

**Servlet A:**url-pattern: **/url  
  
Filter X:**url-pattern: **/url**

1. The first Filter’s **doFilter()** is called. Calling **filterChain.doFilter()** from a doFilter() method will pass the request/response to the **next** Filter (or, finally, the Servlet)

**Servlet A**  
init()  
service()  
doGet()  
…  
destroy()

1. A **FilterChain** object is created to keep track of how Filters are stacked for this HTTP request

**FilterChain**

**Web Server   
(Application)**

1. Container creates **HttpServletRequest** and **HttpServletResponse** objects to process a new HTTP request

**Filter X**doFilter()

**Web Container (Application)**

**Web Server (Machine)**