Software Platforms

LM in Computer Engineering

Massimo Maresca

Technologies

Technologies to move from Java Programs to Software Platforms

- Socket and HTTP Connection
- Threads
- Asynchronous I/O
- Dynamic Class Loading
- Reflection
- Annotations

Platform defines: Services and Interfaces

Applications: provide Service implementations through the Interfaces

In Summary

A PLATFORM can be seen as an APPLICATION CONTAINER

Static vs Dynamic Web - Servlets

- A Web Server typically provides **static content**, i.e., content extracted from the file system through a path name.
- As such it has nothing to do with a Platform.

- Servlets support the creation of **dynamic content**, i.e. , the activation of Applications (i.e., Servlets) that process the input parameters and produce an output, typically in HTML form.
- Applications can be loaded/unloaded/managed when the platform is running (Dynamic Class Loading).

Java Community Process - Servlet Technology

- The Java Community Process (JCP) is a mechanism that allows to develop standard technical specifications for Java technology.
- The JCP involves the use of Java Specification Requests (JSRs), the documents that describe proposal for adding to the Java platform.
- A **servlet** (JSR 340) is a Java™ technology-based Web component, managed by a **container**, that generates **dynamic content**. Servlets are **platform-independent Java classes** that are compiled to **platform-neutral byte code that can be loaded dynamically** into and run by a **Java technology-enabled Web server**.
- **Containers**, sometimes called **servlet engines**, are **Web server extensions that provide servlet functionality**. Servlets interact with Web clients via a **request/response paradigm** implemented by the servlet container.

Servlet Container (from JSR 340)

- A **servlet container** is a part of a Web server or application server that
  - provides the network services over which requests and responses are sent,
  - decodes MIME-based requests, and
  - formats MIME-based responses.
- A servlet container also **contains and manages servlets through their lifecycle**.
- A servlet container can be built into a host Web Server or installed as an add-on component to a Web Server via that server's native extension API.
- Servlet containers can also be built into or possibly installed into **Web-enabled application servers**.
- All servlet containers must support **HTTP as a protocol for requests and responses**, but additional request/response-based protocols such as HTTPS (HTTP over SSL) may be supported.

An Example (from JSR 340)

The following is a typical sequence of events:

1. A client (e.g., a Web browser) accesses a Web server and makes an HTTP request.
2. The request is received by the Web server and handed off to the servlet container. The servlet container can be running in the same process as the host Web server, in a different process on the same host, or on a different host from the Web server for which it processes requests.
3. The servlet container determines which servlet to invoke, based on the configuration of its servlets, and calls it with objects representing the request and response.
4. The servlet uses the request object to find out who the remote user is, what HTTP POST parameters may have been sent as part of this request, and other relevant data. The servlet performs whatever logic it was programmed to perform, and generates data to send back to the client. It sends this data back to the client via the response object.
5. Once the servlet has finished processing the request, the servlet container ensures that the response is properly flushed and returns control back to the host Web server.

Comparison with other technologies (from JS 340)

- In functionality, servlets are similar to the Common Gateway Interface (CGI).
- Servlets have the following advantages over other server extension mechanisms:
  - They are generally much faster than CGI scripts because a different process model is used.
  - They use a standard API that is supported by many Web servers.
  - They have all the advantages of the Java programming language, including ease of development and platform independence.
  - They can access the large set of APIs available for the Java platform.

The Servlet interface (from JSR 340)

- The Servlet interface is the **central abstraction** of the Java Servlet API.
- All servlets implement this interface either directly, or more commonly, by extending a class that implements the interface.
- The two classes in the Java Servlet API that implement the Servlet interface are GenericServlet and HttpServlet.
- For most purposes, Developers will extend HttpServlet to implement their servlets.

Http Servlet Handling Methods (JSR 340)

- The HttpServlet abstract subclass adds the following methods to the basic Servlet interface:
  - doGet for handling HTTP GET requests
  - doPost for handling HTTP POST requests
  - doPut for handling HTTP PUT requests
  - doDelete for handling HTTP DELETE requests 2-6 Java Servlet Specification
  - doHead for handling HTTP HEAD requests
  - doOptions for handling HTTP OPTIONS requests
  - doTrace for handling HTTP TRACE requests.

Typically, when developing HTTP-based servlets, a Servlet Developer will only concern himself with the **doGet** and **doPost** methods. The other methods are considered to be methods for use by programmers very familiar with HTTP programming.

A first program

- Program MyServlet0 shows a simple way to include a servlet in a Web Server.
- Depending on the URL sent by the browser either static content loading or servlet execution can be activated.
- A typical passive socket mechanism is in place. GET requests which include a /servlet keyword go to the servlet based dynamic Web while the other requests go to the static Web processing.
- The URL includes the name of the servlet to be activated.

Moving to a container

- We distinguish the Web domain from the internal Class domain. The URL contains an external identifier which has to be mapped on the internal name of the servlet class to be activated.
- More in general we separate servlet management from servlet execution. We introduce a Management Plane, through a Management Console.
- A servlet manager takes care of loading/unloading the servlets. Loaded servlets get included in an internal repository (e.g., a Hash Table) in such a way to allow a fast search upon activation request. The repository may contain pairs including external servlet names and associated classes.

A second program

- Program MyServlet1 shows a simple way to include a servlet in a Web Server along with a CLI based Management Console.
- The Management Console includes two commands, namely:

  load <classname>

  unload <classname>

Where <classname> refers to a directory including the actual class code and a metadata file to map the external servlet name (to be included in the URL) to the internal class name.

Metadata file vs. Annotation

- The association between internal servlet name and external servlet id may appear in a configuration file and/or in the servlet class.
- In the first case the container must see a servlet as a complex file structure (i.e., a directory or an archive file) including the actual class and a metadata file which includes mapping information.
- In the second case the container must process the servlet class through reflection at deployment to extract the mapping information from the internal annotations.