**Advantages of Servlets:**

1. **Platform Independence:** Servlets run on any platform that supports a servlet container (such as Apache Tomcat), making them highly portable across different operating systems.
2. **High Performance:** Servlets are more efficient compared to traditional CGI (Common Gateway Interface) because they run within the server’s JVM, eliminating the need to start a new process for each request.
3. **Scalability:** Servlets are multi-threaded. The servlet container manages multiple client requests using a thread pool, which leads to better resource utilization and allows the handling of concurrent requests efficiently.
4. **Session Management:** Servlets provide built-in support for session management using HttpSession, allowing the server to maintain user-specific information across multiple requests.
5. **Security:** Java and servlet containers come with strong security features like authentication, authorization, and HTTPS support. Servlets also benefit from the inherent security features of the Java platform, such as strong type checking and automatic memory management.
6. **Extensibility:** Servlets can easily integrate with other Java technologies such as JavaBeans, JDBC (for database connectivity), and other APIs, which provides powerful capabilities for building complex, dynamic web applications.
7. **Lifecycle Management:** The servlet container manages the servlet lifecycle (initialization, request handling, and destruction), providing built-in mechanisms for resource management like connection pooling.
8. **Full Control Over HTTP:** Servlets offer direct control over HTTP requests and responses, including custom handling of GET, POST, PUT, DELETE, etc., making them flexible for building REST APIs and other web services.

**Disadvantages of Servlets:**

1. **Complexity:** Servlets can become difficult to manage in large applications as they require extensive handling of the request and response objects. Developers must manually manage the HTML output and the structure of responses, which can lead to bulky code.
2. **No Separation of Concerns:** Without JSP or another view technology, servlets mix business logic with presentation logic (HTML code and server-side logic are intertwined). This makes the code harder to maintain and debug.
3. **Difficulty in Handling Dynamic Content:** While servlets can handle dynamic content generation, they are less efficient and user-friendly compared to templating engines like JSP or modern frameworks like Thymeleaf and Velocity. Embedding HTML directly in servlets is not ideal.
4. **Lack of Built-In UI Support:** Servlets alone do not provide the tools for managing front-end interfaces (UI). Unlike JSP or other templating frameworks, servlets are mainly focused on backend logic, which means you need to combine them with front-end technologies (HTML, JavaScript, etc.) for the view layer.
5. **Manual Session and Cookie Management:** While session management is supported, it can be cumbersome when complex session logic or custom cookies are required. Developers need to manually track and manage session attributes.
6. **Limited View Technology:** Servlets lack templating and form-handling capabilities out of the box. This makes them less suited for applications that rely heavily on dynamically generated HTML, where templating solutions (such as JSP, Thymeleaf) would be more appropriate.
7. **Learning Curve:** For new developers, working with servlets can involve a steep learning curve due to the need to understand servlet lifecycle, HTTP protocols, session management, and multithreading.