The **12-Factor App** methodology provides best practices for building scalable, maintainable, and cloud-native applications. The key principles include:

1. **Codebase** – A single codebase tracked in version control with multiple deployments.
2. **Dependencies** – Explicitly declare dependencies using package managers (e.g., Maven, Gradle).
3. **Config** – Store configuration in environment variables, not in the code.
4. **Backing Services** – Treat databases, message queues, and caches as attached services.
5. **Build, Release, Run** – Strictly separate build, release, and execution stages.
6. **Processes** – Run applications as stateless processes; share state via external services.
7. **Port Binding** – Self-contained services should expose ports dynamically.
8. **Concurrency** – Scale out by running multiple instances instead of threads.
9. **Disposability** – Fast startup and graceful shutdown for efficient scaling.
10. **Dev/Prod Parity** – Keep development, staging, and production environments as similar as possible.
11. **Logs** – Treat logs as event streams and forward them to external systems.
12. **Admin Processes** – Run management tasks as one-time processes (e.g., database migrations).

**2. Spring Security**

1. **Authentication & Authorization** – Supports username-password, OAuth2, JWT, LDAP, and SSO.
2. **Role-Based Access Control (RBAC)** – Restrict access based on user roles.
3. **CSRF Protection** – Prevents Cross-Site Request Forgery attacks.
4. **Session Management** – Prevents session hijacking and fixation attacks.
5. **Secure API Access** – Protects REST APIs with OAuth2 and JWT authentication.
6. **Method-Level Security** – Uses annotations (@PreAuthorize, @Secured) to enforce security policies.
7. **Password Encoding** – Supports BCrypt, PBKDF2, and Argon2 for secure password hashing.
8. **Security Filters** – Uses a customizable filter chain to enforce security policies at different levels.