**Fundamental Concepts**

1. **Java Programming:**
   * **Core Java:** Understand basic syntax, OOP principles, exception handling, collections framework, multithreading, and Java I/O.
   * **Advanced Java:** Explore Java 8+ features like lambda expressions, streams, and the new Date/Time API.
2. **Version Control:**
   * Learn Git for version control. Understand branching, merging, and common workflows like GitFlow.

**Web Development Basics**

1. **HTTP and REST:**
   * Understand the basics of HTTP, including methods (GET, POST, PUT, DELETE), status codes, and headers.
   * Learn about RESTful web services and principles of REST architecture.
2. **HTML, CSS, JavaScript:**
   * Basic knowledge of frontend technologies is useful to understand how the backend interacts with the client side.

**Spring Framework**

1. **Spring Core:**
   * **Dependency Injection (DI) and Inversion of Control (IoC):** Learn how Spring manages object dependencies.
   * **Spring Configuration:** Understand annotation-based and XML-based configurations.
2. **Spring Boot:**
   * Learn how to create Spring Boot applications, auto-configuration, Spring Boot starters, and application properties.
   * Explore embedded servers (like Tomcat) and how Spring Boot simplifies deployment.

**Database Management**

1. **SQL and Databases:**
   * Understand relational databases, SQL queries, normalization, and indexing.
   * Get hands-on with a relational database like PostgreSQL, MySQL, or MariaDB.
2. **JPA and Hibernate:**
   * Learn the basics of JPA and ORM.
   * Explore Hibernate as a JPA implementation, entity mappings, and JPQL.
3. **Spring Data JPA:**
   * Understand repositories, CRUD operations, query methods, and custom queries.
   * Learn about pagination and sorting.

**RESTful Web Services**

1. **Spring MVC:**
   * Learn how to create RESTful web services using Spring MVC.
   * Understand request mappings, request/response handling, and error handling.
2. **Spring Security:**
   * Basics of securing applications, authentication, and authorization.
   * Learn how to integrate JWT (JSON Web Tokens) for securing REST APIs.

**Testing**

1. **Unit Testing:**
   * Learn JUnit for unit testing Java applications.
   * Explore Mockito for mocking dependencies.
2. **Integration Testing:**
   * Understand how to perform integration testing using Spring’s testing support.

**DevOps and Deployment**

1. **Build Tools:**
   * Learn Maven or Gradle for building and managing dependencies in your Java projects.
2. **CI/CD:**
   * Basics of Continuous Integration/Continuous Deployment. Familiarize yourself with tools like Jenkins, GitLab CI, or GitHub Actions.
3. **Containerization:**
   * Learn Docker for containerizing applications.
   * Understand Docker Compose for multi-container applications.
4. **Cloud Platforms:**
   * Get familiar with cloud platforms like AWS, Google Cloud, or Azure.
   * Learn how to deploy Java applications on these platforms.

**Advanced Topics**

1. **Microservices:**
   * Understand the microservices architecture, advantages, and challenges.
   * Learn Spring Cloud for building microservices.
2. **Reactive Programming:**
   * Explore reactive programming concepts and learn about Project Reactor.
   * Understand how to build reactive applications using Spring WebFlux.

**Resources and Practice**

1. **Books:**
   * "Effective Java" by Joshua Bloch.
   * "Spring in Action" by Craig Walls.
   * "Java Persistence with Hibernate" by Christian Bauer and Gavin King.
2. **Online Courses:**
   * Platforms like Udemy, Coursera, Pluralsight, and free resources like YouTube can be very helpful.
3. **Practice Projects:**
   * Build small projects to apply what you’ve learned. Examples could be a simple CRUD application, a REST API for a blog, or an e-commerce backend.
4. **Contribute to Open Source:**
   * Join open-source projects to gain practical experience and learn from the community.