The Spring Framework is a Java platform that provides comprehensive infrastructure support for developing Java applications.

Agenda

- Kurze Einführung/Wiederholung
  - Maven
  - Tomcat
  - Servlets
  - MVC, JSP, JSTL
- Hauptteil Spring
  - Spring WebMVC
  - Mockito
  - Spring Dependency Injection
  - RestController

Mavens Standard-Lebenszyklus (vereinfacht)

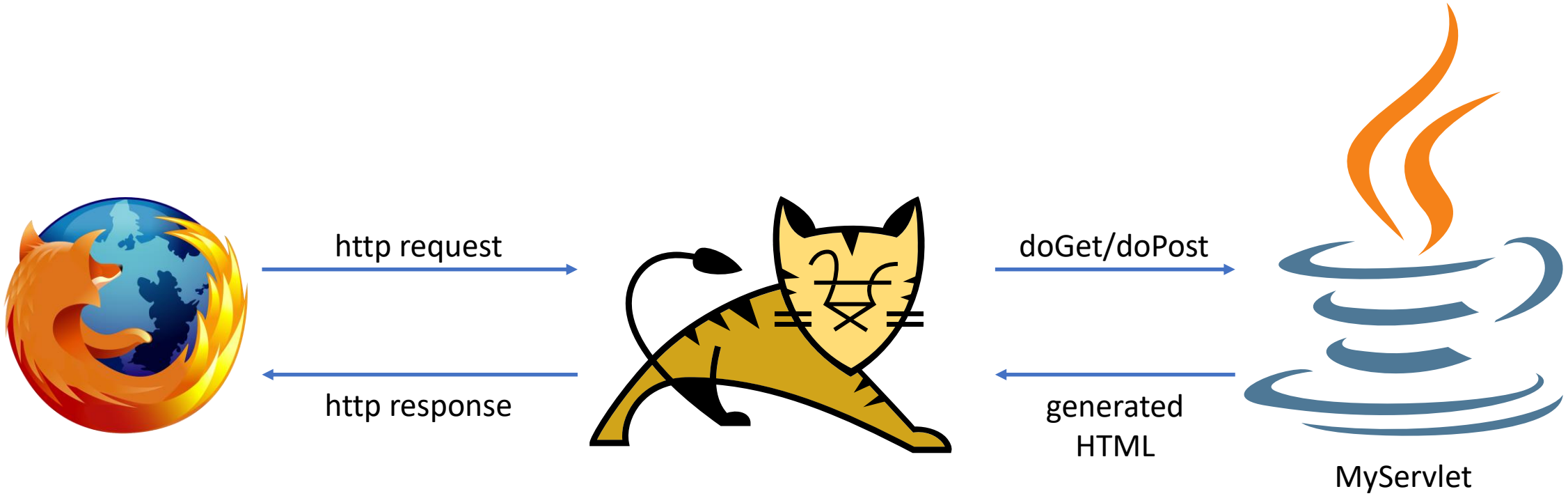
Externe Abhängigkeiten

- Reale Projekte haben meist Abhängigkeiten zu Bibliotheken und Frameworks in Form von JARs (z.B. JUnit oder Spring)
  - Auf welchen Webseiten finden wir diese JARs?
  - Wo landen sie auf unserem Rechner?
  - Legen wir die JARs einfach im git Repository ab? Oder muss sie jeder Entwickler selber ins richtige Verzeichnis herunterladen?
- Maven lädt die externen Abhängigkeiten automatisch aus einem zentralen online-Repository in ein lokales offline-Repository herunter.

Maven Scopes (Auszug)

Scope	Im compile classpath	Im test classpath	Wird mit ausgeliefert
compile (default)	✓	✓	✓
provided (z.B. servlet-api)	✓	✓	
test (z.B. junit)		✓	

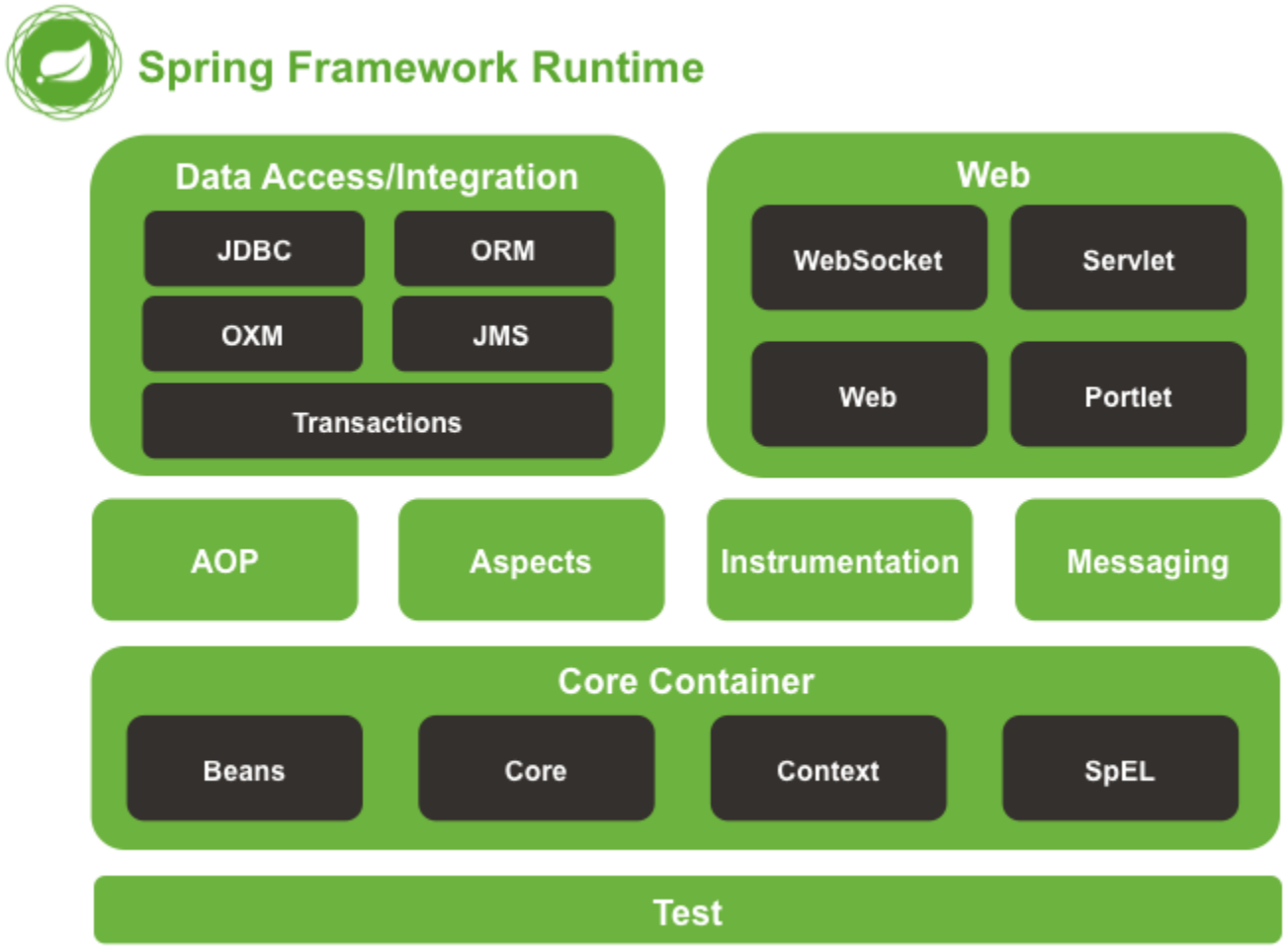
Tomcat und Servlets



Model-View-Controller Muster

- Das Generieren von HTML mittels out.println macht wenig Spaß
- Schöner wäre es, wenn das Servlet die anzuzeigenden Daten bündeln und an eine Art „Dynamische HTML-Seite“ weiterreichen könnte
- Diese „Dynamischen HTML-Seiten“ nennt man Java Server Pages
- Konkrete Ausprägung des Model-View-Controller Musters

Framework Modules



Dependency Injection

Angenommen, ein Objekt der Klasse Foo benötigt ein Objekt der Klasse Bar:

```
class Foo {
   private Bar bar;
   // ...
}
```

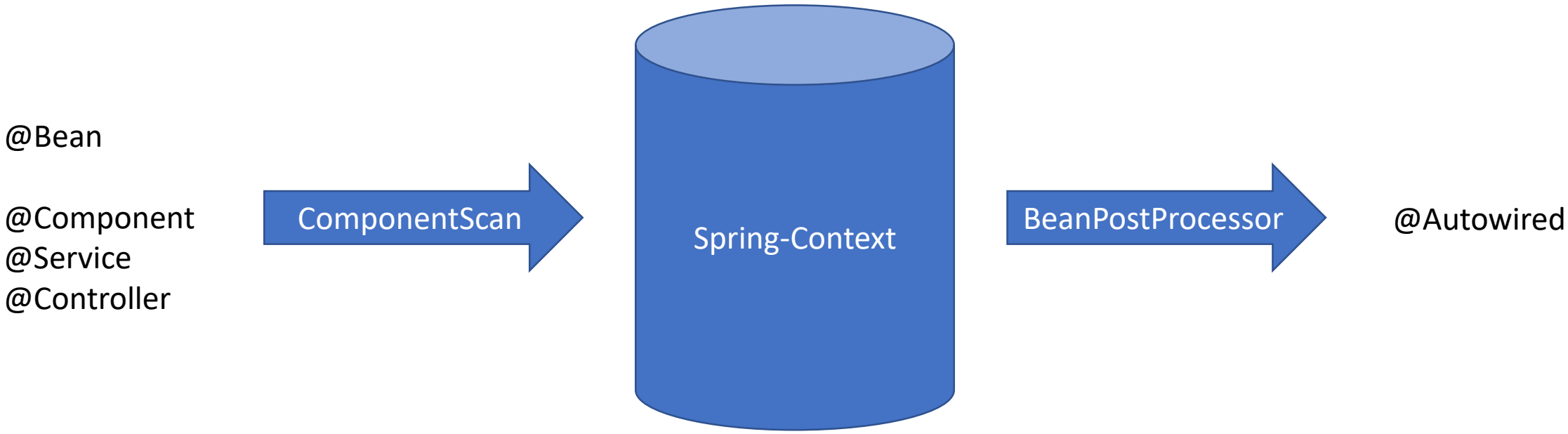
Wie und wo wird das Bar-Objekt erzeugt?

1. Manuell innerhalb der Klasse Foo

2. Manuell außerhalb der Klasse Foo (Dependency Injection)

3. Automatisch außerhalb der Klasse Foo (Dependency Injection Framework)

Dependency Injection über Spring-Context



Drei Varianten von Dependency Injection

```
// Constructor Dependency Injection
private A a;

@Autowired
public Foo(A a) { this.a = a; }


// Setter Dependency Injection
private B b;

@Autowired
public void setB(B b) { this.b = b; }


// Field Dependency Injection
@Autowired
private C c;
```