Spring Boot Annotations – Option B (Level 2)

Annotation	Description	Example
@SpringBootApplication	Main Spring Boot entry point	public class App {}
@Component	Generic Spring component	@Component class A {}
@Service	Business logic layer	@Service class UserService {}
@Repository	DAO layer + exception translation	@Repository class UserRepo {}
@Controller	MVC controller returning views	@Controller class HomeController {}
@RestController	REST controller returning JSON	@RestController class Api {}
@Autowired	Dependency injection	@Autowired UserService service;
@Qualifier	Resolve multiple bean conflicts	@Qualifier("bean1")
@Primary	Preferred bean when multiple exist	@Primary class DefaultService {}
@Lazy	Lazy initialization of bean	@Lazy @Autowired Service s;
@RequestMapping	Root endpoint mapping	@RequestMapping("/api")
@GetMapping	GET endpoint	@GetMapping("/users")
@PostMapping	POST endpoint	@PostMapping("/save")
@PutMapping	PUT endpoint	@PutMapping("/update")
@DeleteMapping	DELETE endpoint	@DeleteMapping("/remove")
@PatchMapping	PATCH endpoint	@PatchMapping("/modify")
@PathVariable	Extract URL variable	@GetMapping("/users/{id}")
@RequestParam	Extract query parameter	@RequestParam String name
@RequestBody	Bind JSON body	@PostMapping("/save")
@ResponseStatus	Set custom HTTP status	@ResponseStatus(HttpStatus.CREATED)
@CrossOrigin	Enable CORS	@CrossOrigin(origins="*")
@ExceptionHandler	Handle exceptions	@ExceptionHandler(Exception.class)

@RestControllerAdvice	Global exception handler	class GlobalHandler {}
@Entity	Marks a JPA entity	@Entity class User {}
@Table	Custom table name	@Table(name="users")
@Id	Primary key	@Id private Long id;
@GeneratedValue	ID generation	@GeneratedValue(strategy=IDENTITY)
@Column	Custom column	@Column(nullable=false)
@OneToOne	One-to-one mapping	@OneToOne UserProfile p;
@OneToMany	One-to-many mapping	@OneToMany List<Order> orders;
@ManyToOne	Many-to-one mapping	@ManyToOne User user;
@ManyToMany	Many-to-many mapping	@ManyToMany Set<Role> roles;
@JoinColumn	Join key column	@JoinColumn(name="user_id")
@Transactional	Transaction boundary	@Transactional public void save() {}
@EnableTransactionManagement	Enable @Transactional	Used in config class
@Valid	Trigger validation	@PostMapping public save(@Valid User u)
@NotBlank	String must not be blank	@NotBlank String name;
@NotNull	Must not be null	@NotNull Long id;
@Size	Size limit	@Size(max=50) String name;
@Email	Email validation	@Email String email;
@Min / @Max	Numeric limits	@Min(1) int qty;
@Pattern	Regex validation	@Pattern("[A-Z]{3}")
@Getter/@Setter	Generate getters/setters	@Getter @Setter class A {}
@Data	Full Lombok model	@Data class User {}
@Builder	Builder pattern	@Builder class User {}
@Scheduled	Scheduled tasks	@Scheduled(fixedRate=1000)
@EnableScheduling	Enable scheduling	@EnableScheduling class Config {}

@Cacheable	Cache method results	@Cacheable("users")
@Async	Run method async	@Async void sendEmail() {}
@Value	Inject config values	@Value("${app.name}")
@ConfigurationProperties	Bind properties to class	@ConfigurationProperties(prefix="app")
@SpringBootTest	Full app context test	@SpringBootTest
@WebMvcTest	Controller layer test	@WebMvcTest(UserController.class)
@DataJpaTest	Repository layer test	@DataJpaTest
@MockBean	Mock dependency in tests	@MockBean UserService svc;
@Test	JUnit test	@Test void testA() {}
@Aspect	AOP aspect class	@Aspect class LogAspect {}
@Before	Run before joinpoint	@Before("execution(*)")
@After	Run after joinpoint	@After("execution(*)")
@Around	Wrap method execution	@Around("execution(*)")
@Pointcut	Define pointcut expression	@Pointcut("execution(*)")