لما الـ scope يكون `prototype` في Spring، الـ **container ما بيكملش إدارة دورة حياة الـ bean بعد إنشائه** — يعني Spring ينشئ الـ bean ويطبّق الـ init callbacks، لكن **مش بيتابع الكائن بعد كده ومش بينفذ destroy/@PreDestroy** له. ده قرار تصميمي في Spring

ليه كده؟ (السبب بالتفصيل)

1. **دور الحاوية (container) يقتصر على الإنشاء والتهيئة** للـ prototype: Spring ينشئ، يحقن التبعيات، ينفّذ init (`@PostConstruct`, `afterPropertiesSet`, إلخ)، ثم يسلم المرجع للعميل — وبعد التسليم **ما فيش سجل متابعة** لذلك الكائن من طرف الحاوية. لذلك الحاوية لا تستدعي دوال التدمير لأنّه ببساطة لا تعرف متى ينتهي استخدام الكائن.

2. **لو الحاوية سجّلت كل كائن prototype للـ destroy لاحقاً، ممكن يحصل leak** — لأن تطبيقات قد تنشئ آلاف كائنات prototype، وتتوقع الجافا garbage collector يتعامل معهم بعد انتهاء الاستخدام. إذا الحاوية احتفظت بمراجع لكل prototype لحين الإغلاق ثم دمّرتها، ده سيؤدي لتنظيم زائد ومخاطر تسريب ذاكرة. لذلك قرار Spring هو ترك مسؤولية التدمير للعميل.

3. **نتيجة عملية:** الـ lifecycle callbacks الخاصة بالتهيئة تُستدعى دائماً، أما callbacks الخاصة بالتدمير (مثل `@PreDestroy`, `DisposableBean.destroy()`, أو `destroy-method`) **فلا تُستدعى تلقائياً** للـ prototype