As anotações são funcionalidades introduzidas no [**Spring a partir da versão 2.5**](http://www.devmedia.com.br/curso/introducao-ao-spring-framework/191), possibilitando aos desenvolvedores um aumento significativo em produtividade. Isto se deve à facilidade de configuração via anotações, o que era mais complicado nas versões anteriores do Spring, pois toda configuração era via XML. Mas nem só de anotações vive o Spring. Existem muitos conceitos importantes que devem ser bem compreendidos. Dois dos principais são a Injeção de Dependência e Inversão de Controle. Existem muitos artigos bons que explicam estes conceitos, principalmente o de Martin Fowler, e outros aqui mesmo na DevMedia, portanto, nos restringiremos a explicações superficiais e práticas destes conceitos.

A Inversão de Controle, basicamente, é uma forma diferente de manipularmos o controle sobre um objeto. Não somos nós que iremos gerenciar a criação de instância dos objetos (no popular, dar o famoso "new"). Quem se encarregará de fazer isso, no nosso caso, é o próprio Spring. E a injeção de dependências? Bom, como não vamos criar um novo objeto, então precisaremos que alguém nos dê esse objeto, ou mais precisamente, nos dê essa dependência. Além disso, precisaremos que este alguém (o Spring) injete essa dependência em nossa classe. Confuso? Vamos ver como é simples, a partir de um exemplo.

Os mecanismos do Spring para implementar estes conceitos de injeção de dependência e inversão de controle, necessitam de algumas anotações. As anotações que veremos neste artigo estão baseadas na versão 3.1 do Spring. A **Tabela 1** mostra essas anotações:

|  |  |
| --- | --- |
| **@Component** | Anotação genérica para qualquer componente gerenciado pelo Spring. Esta anotação faz com que o bean registrado no Spring possa ser utilizado em qualquer bean, seja ele um serviço, um DAO, um controller, etc. No nosso exemplo, ele será responsável por um Bean que representa uma entidade. |
| **@Repository** | Anotação que serve para definir uma classe como pertencente à camada de persistência. |
| **@Service** | Anotação que serve para definir uma classe como pertencente à camada de Serviço da aplicação. |
| **@Autowired** | A anotação @ Autowired fornece controle sobre onde e como a ligação entre os beans deve ser realizada. Pode ser usado para em métodos setter, no construtor, em uma propriedade ou métodos com nomes arbitrários e / ou vários argumentos. |

A anotação @Service representa esta classe como se fosse um serviço. Já as anotações @Autowired permitem que o Spring injete as dependências nesta classe