



Anotações

O objetivo da mudança das antigas versões do Java 2 SE para o Java SE 5 está centrada no aumento da facilidade do desenvolvimento

Esses novos recursos transferem a responsabilidade pela construção de códigos completos (XDoclet, XML, etc) do programador para o compilador

Um destes novos recursos no Java SE 5 é chamado de Anotações (Annotations)

Anotações são como meta-tags que você pode adicionar ao seu código e aplica-los em declaração de pacotes (packages), declaração de tipos, construtores, métodos, campos, parâmetros e variáveis

Com o passar dos anos a implementação de anotações foi intensa, assim hoje podemos anotar os mais diversos tipos de recursos durante o desenvolvimento

RESTFull WebService

```
@Path("bean")
public class NomeBean {
    private NomeService dao = NomeService.getInstance();

    @GET
    @Path("/{nome}")
    @Produces(MediaType.APPLICATION_XML)
    public List<Nome> getByNome(@PathParam("nome") String nome) {
        return dao.buscar(nome);
    }

    @POST
    @Consumes(MediaType.APPLICATION_XML)
    @Produces(MediaType.APPLICATION_XML)
    public Nome addNome(JAXBElement<Nome> obj) {
        Nome n = obj.getValue();
        return dao.salvar(n);
    }
}
```

Enterprise Java Bean

```
@Stateless
public class CarrinhoDao {
    @PersistenceContext(unitName = "JsfWebShop")
    private EntityManager session;
    @EJB
    private ListaDeProdutos dao;
}
```

Java Server Faces

```
@ManagedBean
@SessionScoped
public class QuizBeam {
    private String linguagem;
    private boolean mostraVotacao = false;
}
```

Java To XML Bind

```
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "", propOrder = {
    "logradouro",
    "numero",
    "complemento",
    "bairro",
    "cep",
    "cidade",
    "estado"
})
@XmlRootElement(name = "endereco")
public class Endereco {
    @XmlElement(required = true)
    protected String logradouro;
}
```

Java Persistence API

```
@Entity
@Table(name="CadData")
public class ObjData {
    @Id
    @Column(name="idCadData")
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    private Integer id;
    @Temporal(TemporalType.DATE)
    private Date data;
}
```

Servlet

```
@WebServlet("/cadProduto")
public class CadProduto extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
    }
}
```

Definindo uma Anotação (Tipo Anotação)

```
public @interface MinhaAnotacao {  
    String fazAlgo();  
}
```

O Tipo Anotação (Declarando/Utilizando)

```
@MinhaAnotacao(fazAlgo="O que fazer")  
public void meuMetodo() {  
    ...  
}
```

Tipos de Anotação

"Marcador" O tipo de anotação marcador não tem elementos, exceto o próprio nome

```
public @interface MinhaAnotacao {  
}
```

Uso

```
@MinhaAnotacao  
public void meuMetodo() {  
    ...  
}
```

“Único elemento” O tipo único elemento oferece apenas um único valor

```
public @interface MinhaAnotacao {  
    String fazAlgo();  
}
```

Uso

```
@MinhaAnotacao("O que fazer");  
public void meuMetodo() {  
    ...  
}
```

“Multi-valorado” Anotações deste tipo têm muitos membros

```
public @interface MinhaAnotacao {  
    String fazAlgo();  
    int contador;  
    String data();  
}
```

Uso

```
@MinhaAnotacao(fazAlgo="O que fazer", contador=1,  
    data="06-12-2006")  
public void meuMetodo() {  
    ...  
}
```

Passos para a definição de um tipo de anotação

- Uma declaração de anotação deve iniciar com @, seguido da palavra reservada “interface” seguido do nome da anotação.
- A declaração de métodos não deve conter nenhum parâmetro.
- A declaração de métodos não deve lançar nenhuma exceção (throws).

Os tipos retornados pelos métodos deve ser um dos seguintes tipos

- tipos primitivos (`byte`, `short`, `int`, `char`, `long`, `float`, `double` e `boolean`)
- `String`
- `Class`
- `Enum`
- um array dos tipos acima

Existem dois tipos de anotações disponíveis
no Java SE 8

- Anotações Simples
- Meta Anotações

Anotações Simples

- Override
- Deprecated
- Suppresswarnings

A anotação override

- Indica que o método anotado necessita ser sobreposto (re-implementado) pois é herdado da superclasse

```
public class Test_Override {  
    @Override  
    public String toString() {  
        return super.toString() +  
            "Testando a anotação: 'Override'";  
    }  
}
```

A anotação Deprecated

- Marca o método como “depreciado”

```
public class Test_Deprecated {  
    String saldo = "";  
  
    @Deprecated  
    public void imprimeSaldo() {  
        return saldo;  
    }  
}
```

A anotação SuppressWarnings

- Esta anotação indica ao compilador que as mensagens de '**warning**' não deverão ser emitidas para o elemento anotado bem como para todos os seus sub-elementos

```
@SuppressWarnings({"deprecation"})  
public TestAnnotations() {  
    Test t2 = new Test();  
    t2.doSomething();  
}
```

Meta-Anotações

São anotações de anotações

- Target
- Retention
- Documented
- Inherited

A anotação Target

Esta anotação indica o elemento alvo para a classe a qual o tipo de anotação será aplicada

- `@Target(ElementType.TYPE)` – pode ser aplicado a qualquer elemento de uma classe
- `@Target(ElementType.FIELD)` – pode ser aplicado a um atributo de classe
- `@Target(ElementType.PARAMETER)` – pode ser aplicado a um parâmetro de um método

- `@Target(ElementType.CONSTRUCTOR)` – pode ser aplicado aos construtores da classe
- `@Target(ElementType.LOCAL_VARIABLE)` – pode ser aplicado a variáveis locais de métodos numa classe
- `@Target(ElementType.ANNOTATION_TYPE)` – indica que o tipo declarado é um tipo de anotação

```
import java.lang.annotation.*;
```

```
@Target(ElementType.METHOD)  
public @interface Test_Target {  
    public String doTestTarget();  
}
```

A anotação Retention

A anotação retention indica onde e por quanto tempo as anotações de um determinado tipo serão retidas

- `@Retention(RetentionPolicy.SOURCE)` – Anotações deste tipo serão retidas somente no fonte da classe e será ignorada pelo compilador
- `@Retention(RetentionPolicy.CLASS)` – Anotações deste tipo serão retidas até o tempo de compilação, mas serão ignoradas quando em execução na VM

- `@Retention(RetentionPolicy.RUNTIME)` – Anotações deste tipo serão retidas pela VM assim elas poderão ser lidas (somente) em tempo de execução

```
import java.lang.annotation.*;
```

```
@Retention(RetentionPolicy.RUNTIME)
public @interface Test_Retention {
    String doTestRetention();
}
```

A anotação Documented

- A anotação documented indica que uma anotação com este tipo deverá ser documentada pela ferramenta **Javadoc**

```
import java.lang.annotation.*;
```

```
@Documented  
public @interface Test_Documented {  
    String doTestDocument();  
}
```

A anotação Inherited

- Uma classe anotada com um tipo de anotação marcada como inherited é herdada automaticamente por todas as subclasses da classe a qual esta anotação for declarada

```
import java.lang.annotation.*;
```

```
@Inherited
```

```
public @interface MyAnnotation {  
    boolean isInherited() default true;  
    String doSomething() default "Do what?";  
}
```

Fim