1 -Declarando managed bean

```java

package bean;

import java.io.Serializable;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;

@ManagedBean
@SessionScoped
public class User implements Serializable{

    private String username;
    private String password;

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }


}
```

- **Username e password são propriedades do bean;**
- **O escopo é do tipo sessionscoped. Escopos que podem ser associados ao bean:**
    1. Sessionscoped : o bean será mantido enquanto a sessão http esta ativa
    2. Requestscoped: o bean será mantido enquanto houver requisições http
    3. Viewscoped: o bean será mantido enquanto ele estiver na mesma view(pagina)
    4. Applicationscoped: o bean será mantido enquanto a aplicação esta online(o bean será acessível a todos)
    5. Nonescoped: o bean não será mantido nem inicializado no escopo porem ele pode ser acessado por outro bean herdando assim o escopo do bean que o requisitou.

2 - Inicializando managed bean

```java

package bean;

import java.io.Serializable;
import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.ManagedProperty;
import javax.faces.bean.SessionScoped;

@ManagedBean
@SessionScoped
public class User implements Serializable{

    @ManagedProperty(value = "teste")
    private String username;

    private String password;

    @PostConstruct
    private void init(){
        System.out.println("iniciando o bean..usuario tem valor = "+username);
    }

    @PreDestroy
    private void clean(){
        System.out.println("destruindo o bean.");
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }


}
```

- **@ManagedProperty inicializa uma propriedade do bean.**
- **@PostConstruct chama um método quando o bean é inicializado ou instanciado..**
- **@PreDestroy chama um método antes do bean ser destruído.**

3 - Gerenciando dependências de bean

JSF suporta inversão de controle(IoC) ou seja o managed bean pode ser acoplado em tempo de execução sem precisar manipular esse acoplamento no código da aplicação;

```java

package bean;

import java.io.Serializable;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.ManagedProperty;
import javax.faces.bean.NoneScoped;

@ManagedBean(name = "profession")
@NoneScoped
public class Profession implements Serializable{

    @ManagedProperty(value = "Software Engineer")
    private String title;

    @ManagedProperty(value = "TI")
    private String industry;

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getIndustry() {
        return industry;
    }

    public void setIndustry(String industry) {
        this.industry = industry;
    }


}
/*##############################################################################################*/

package bean;

import java.io.Serializable;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.ManagedProperty;
import javax.faces.bean.SessionScoped;

@ManagedBean
@SessionScoped
public class User implements Serializable{

  @ManagedProperty("#{profession}")
  private Profession profession;

    public Profession getProfession() {
        return profession;
    }

    public void setProfession(Profession profession) {
        this.profession = profession;
    }



}
```

Acessando managed beans através do código

```java

package bean;

import java.io.Serializable;
import javax.el.ELContext;
import javax.el.ExpressionFactory;
import javax.el.ValueExpression;
import javax.faces.application.Application;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import javax.faces.context.FacesContext;

@ManagedBean
@SessionScoped
public class OutroBean implements Serializable{

    public User getUser(){

        FacesContext context = FacesContext.getCurrentInstance();
        Application application = context.getApplication();
        ELContext el = context.getELContext();
        ExpressionFactory ef = application.getExpressionFactory();
        ValueExpression value = ef.createValueExpression(el,"#{user}",User.class);
        User user = (User) value.getValue(el);

        return user;
    }
}
```

Usando @Named e @Inject para pegar dados de outro bean

```java
package bean;

import java.io.Serializable;
import javax.faces.bean.SessionScoped;
import javax.inject.Inject;
import javax.inject.Named;

@Named
@SessionScoped
public class User implements Serializable{

  @Inject
  private Profession profession;

    public Profession getProfession() {
        return profession;
    }

    public void setProfession(Profession profession) {
        this.profession = profession;
    }


}
```

```java

package bean;

import java.io.Serializable;
import javax.annotation.PostConstruct;
import javax.inject.Named;

/**
 *
 * @author Deivid
 */
@Named
public class Profession implements Serializable {

    private String title;

    private String industry;

    @PostConstruct
    public void ini() {
        this.title = "Software Engineer";
        this.industry = "TI";
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getIndustry() {
        return industry;
    }

    public void setIndustry(String industry) {
        this.industry = industry;
    }

}


```