Agenda completa com Primefaces Schedule

Escrito em 13/11/2018 18:00 por Benigno

Olá pessoal iremos criar nesse tutorial um exemplo de agenda de compromissos simples com o componetente SCHEDULE. Iremos demonstrar toda a configuração, inclusive como modificar a cor de fundo de eventos, configuração do calendário para português timezone e formatação de data e hora. No próximo post falaremos como integrar a sua aplicação(com schedule) com o Google Calendar.

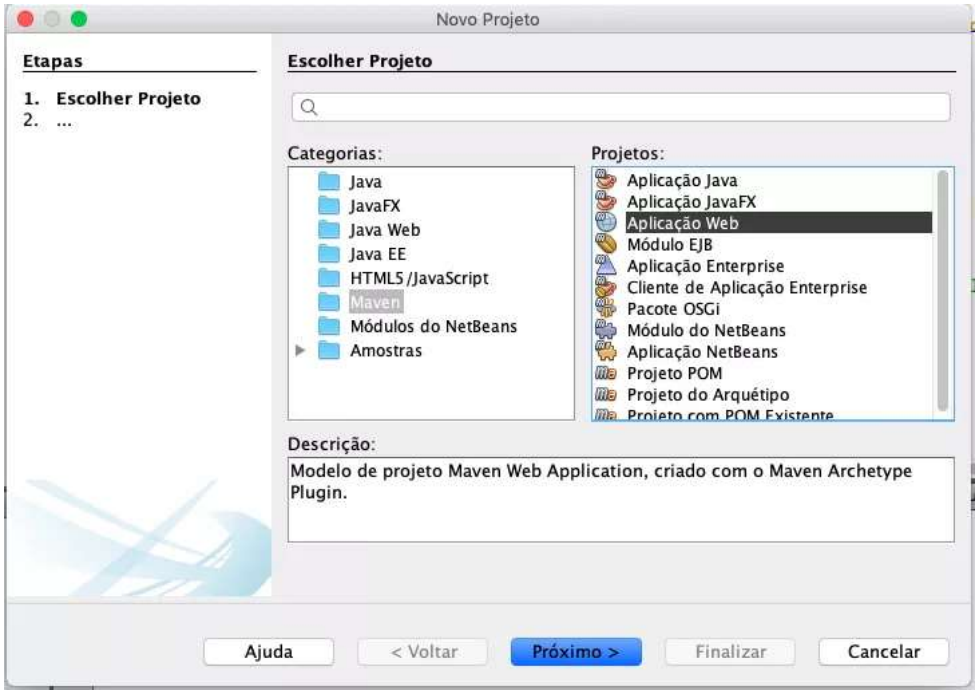
O componente schedule esta presente no primefaces desde as primeiras versões, tendo um grande aprimoramento a partir da versão 4. Ele disponibiliza um calendário baseado nos calendários mais famosos dos principais sistemas operacionais com suporte a JSF e totalmente gerenciado por eventos. Hoje esse componente conta com diversas opções de configuração e customização, como veremos no tutorial abaixo. Este será o primeiro tutorial de uma série de postagens sobre agenda/calendário com JSF. Abaixo um pouco sobre as principais propriedades do componente e sua utilização na prática.

id – Identificador Único do componente na arvore de componentes.
rendered – Valor booleano para identificar se o componente deve ou não ser renderizado.
value – Uma instância de org.primefaces.model.ScheduleModel representada no managed bean.
locale – Define a língua, estilo da data, etc. Necessita configuração em javascript.
view – O tipo de visualização: month, agendaDay, agendaWeek, basicWeek, basicDay.
initialDate – Objeto do tipo date com a data base para ser exibida quando o calendário abrir.
showWeekends – Booleano que demonstra se exibe ou não os fins de semana.
allDaySlot – Booleano que determina se o slot all-day, que fica no topo dos calendários deverá ou não ser exibido.
timeZone – É utilizado para definir a data/hora do componente de acordo com a região que você esta.
locale – Utilizado para definir a língua que o componente será apresentado, por padrão é inglês, mas nós utilizaremos pt-br(português do Brasil).
timeFormat – Define o padrão do formato da hora a ser exibida no componente.
axisFormat – Determina o formato que do texto que será exibido na vertical no componente.
minTime – Horário mínimo a ser exibido no componente
maxTime – Horário máximo a ser exibido no componente
style – CSS

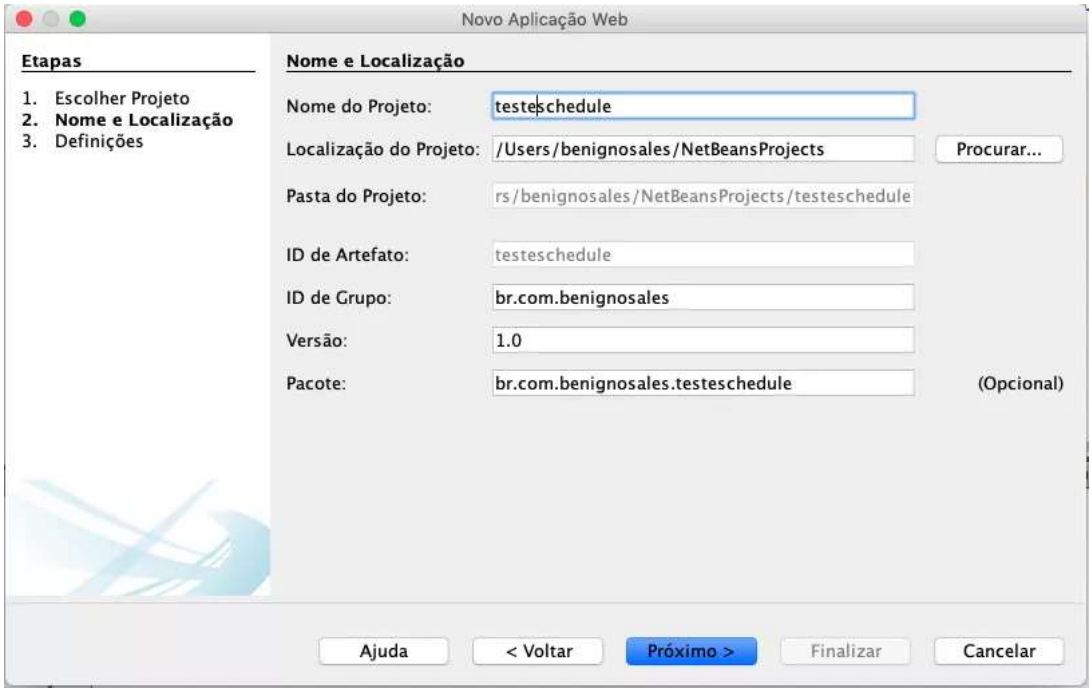
Para nosso projeto, vamos utilizar:

- Netbeans 8.1
- Glassfish 4.1.2 com Mojarra 2.2.1
- Primefaces 6.1

Crie uma nova aplicação Java Web com maven no Netbeans, de acordo com as imagens:

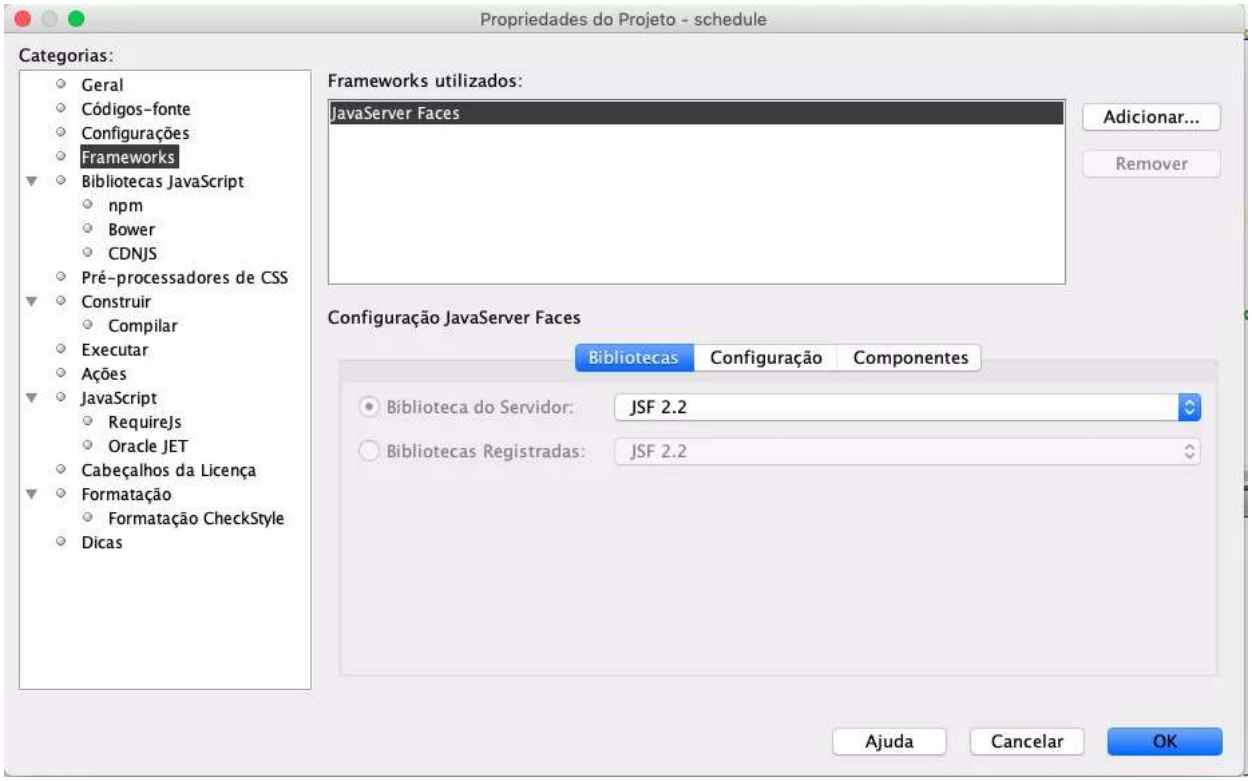


Defina os dados do projeto, como o Nome do Projeto, pasta onde o código será armazenado, pacote, etc.

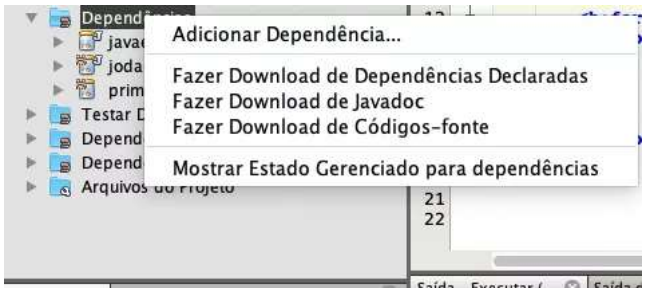


Por último, selecione a versão do JAVA EE e qual o Servidor de Aplicação irá utilizar, no nosso caso utilizamos Java EE 7 com Glassfish 4.1.1.

Clique com o botão direito no nome do projeto e na Categoria Frameworks, clique em Adicionar e selecione JavaServer Faces.



No projeto, clique com o botão direito em Dependências, em seguida clique em Adicionar Dependência.



Pesquise e acrescente duas dependências: Primefaces 6.1 e JodaTime 2.9.9.

```
<dependency>
            <groupId>org.primefaces</groupId>
            <artifactId>primefaces</artifactId>
            <version>6.1</version>
</dependency>
<dependency>
            <groupId>joda-time</groupId>
            <artifactId>joda-time</artifactId>
            <version>2.9.9</version>
</dependency>
```

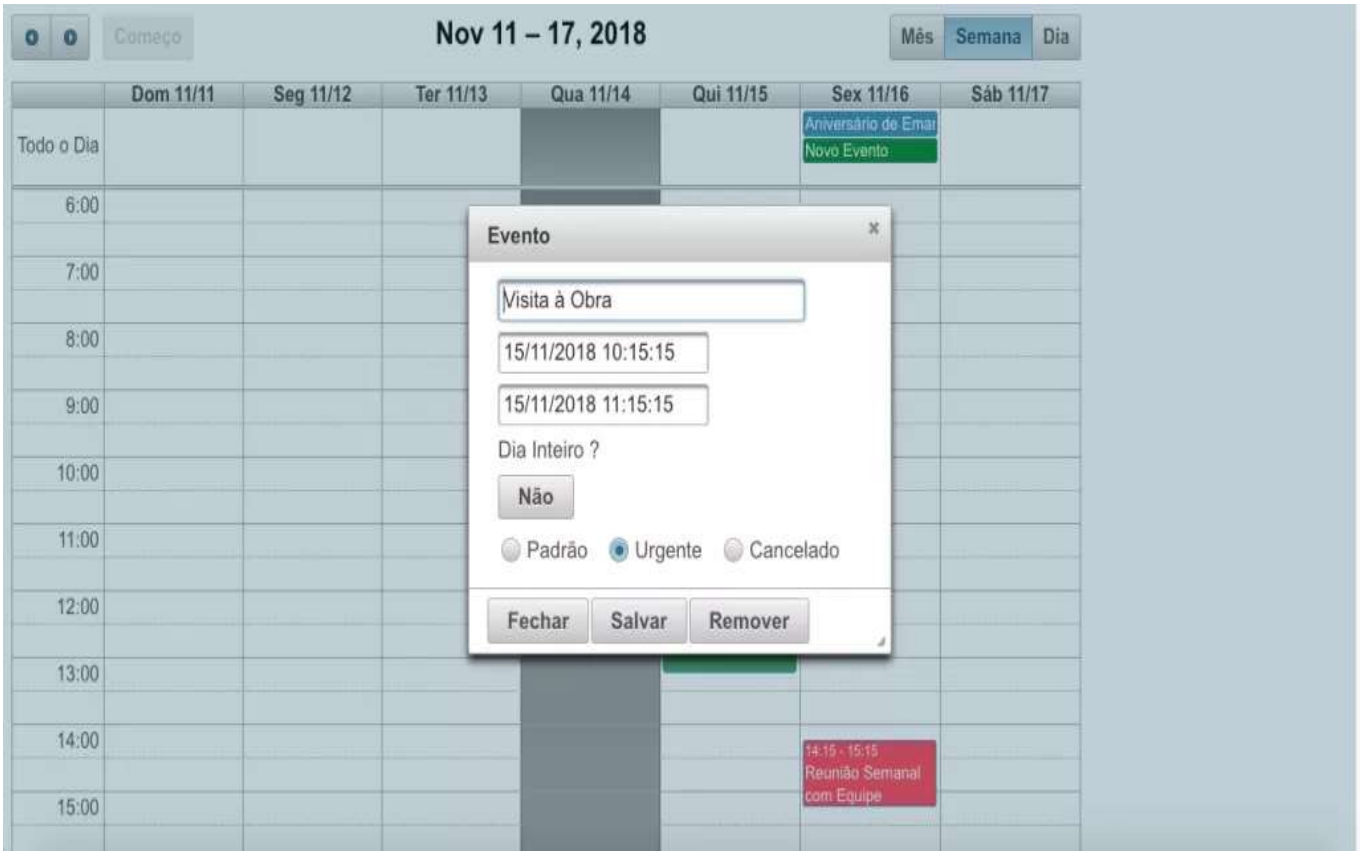
Crie arquivo index.xhtml e insira o seguinte código:

```
    <h:head>
            <title>Schedule Example</title>
        </h:head>
        <h:body>
            <h:outputStylesheet library="css" name="schedule.css"/>
            <h:outputScript library="js" name="schedule.js"/>
            <p:growl id="avisosGrowl"/>
            <h:form>
                <p:schedule value="#{scheduleBean.model}"
                            widgetVar="scheduleExemplo"
                            id="scheduleExemplo" timeZone="America/Fortaleza" locale="pt-br"
                            view="agendaWeek" timeFormat="HH:mm" axisFormat="H:mm" minTime="6:00" maxTime="19:00"
                            style="width: 1000px; height: 700px;">
                    <p:ajax event="dateSelect" listener="#{scheduleBean.onDateSelect}" update="#{p:component('scheduleExemploForm')}" oncc
                    <p:ajax event="eventSelect" listener="#{scheduleBean.onEventSelect}" update="#{p:component('scheduleExemploForm')}" or
                </p:schedule>
            </h:form>
            <h:form id="scheduleExemploForm">
                <p:dialog widgetVar="eventDialog" header="Evento" responsive="true" modal="true">
                    <p:panelGrid columns="1" id="panelGridEvento" layout="grid" styleClass="ui-panelgrid-blank" columnClasses="ui-grid-col
                        <p:inputText value="#{scheduleBean.evento.titulo}" placeholder="Título" size="30"/>
                        <p:calendar pattern="dd/MM/yyyy HH:mm:ss"  value="#{scheduleBean.evento.dataInicio}" placeholder="Data Início">
                        </p:calendar>
                        <p:calendar  value="#{scheduleBean.evento.dataFim}" pattern="dd/MM/yyyy HH:mm:ss"  placeholder="Data Fim">
                        </p:calendar>
                        <p:outputLabel value="Dia Inteiro ?" for="diaInteiro"/>
                        <p:selectBooleanButton id="diaInteiro" value="#{scheduleBean.evento.diaInteiro}" onLabel="Sim" offLabel="Não" labe
                        <p:selectOneRadio value="#{scheduleBean.evento.tipoEvento}">
                            <f:selectItems value="#{scheduleBean.tiposEventos}" var="tipoEvento" itemLabel="#{tipoEvento.descricao}" itemV
                        </p:selectOneRadio>
                    </p:panelGrid>
                    <f:facet name="footer">
                        <p:commandButton value="Fechar" onclick="PF('eventDialog').hide(); return false;"/>
                        <p:commandButton value="Salvar" update="#{p:component('avisosGrowl')},#{p:component('scheduleExemplo')},@form" prc
                        <p:commandButton value="Remover" update="#{p:component('avisosGrowl')},#{p:component('scheduleExemplo')},@form" ac
                    </f:facet>
                </p:dialog>
            </h:form>
        </h:body>
```

No componente Schedule temos o value que é a representação visual do Modelo que será gerenciado no ManagedBean. Todos os atributos utilizados foram explicados no início da postagem, caso queira conhecer melhor os demais atributos do componente, leia sua documentação no site do Primefaces.

Utilizaremos 2 eventos ajax para interagirmos com o calendário, adicionando, excluindo ou alterando eventos. São eles dateSelect, para quando clicarmos em uma data do calendário ele chame um método do bean para inicializar as variáveis do novo evento do calendário, o eventSelect, para quando clicarmos em um evento existente o visualizarmos na caixa de dialogo(dialog) e podermos excluí-lo ou altera-lo. Na caixa de dialog utilizaremos um inputText para o título, 2 calendars para as datas inicial e final, um selectBooleanButton para selecionarmos se o evento é de dia inteiro ou não e um selectOneRadioButton para indicar o tipo de evento que estamos criando/editando. O layout simples ficará de acordo com as imagens abaixo:

Evento Selecionado



Vamos criar uma Classe modelo Evento, um Enum TipoEvento e um ScheduleEvent customizado chamado CustomScheduleEvent. Abaixo o código de cada classe:

```
public class Evento {

    private Long id;
    private String titulo;
    private Date dataInicio;
    private Date dataFim;
    private boolean diaInteiro;
    private TipoEvento tipoEvento;

    public Evento() {
        this.tipoEvento = TipoEvento.PADRAO;
        this.titulo = "";
        this.diaInteiro = false;
    }

    public Evento(Long id, String titulo, Date dataInicio, Date dataFim, boolean diaInteiro, TipoEvento tipoEvento) {
        this.id = id;
        this.titulo = titulo;
        this.dataInicio = dataInicio;
        this.dataFim = dataFim;
        this.diaInteiro = diaInteiro;
        this.tipoEvento = tipoEvento;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getTitulo() {
        return titulo;
    }

    public void setTitulo(String titulo) {
        this.titulo = titulo;
    }

    public Date getDataInicio() {
        return dataInicio;
    }

    public void setDataInicio(Date dataInicio) {
        this.dataInicio = dataInicio;
    }

    public Date getDataFim() {
        return dataFim;
    }

    public void setDataFim(Date dataFim) {
        this.dataFim = dataFim;
    }

    public boolean isDiaInteiro() {
        return diaInteiro;
    }

    public void setDiaInteiro(boolean diaInteiro) {
        this.diaInteiro = diaInteiro;
    }

    public TipoEvento getTipoEvento() {
        return tipoEvento;
    }

    public void setTipoEvento(TipoEvento tipoEvento) {
        this.tipoEvento = tipoEvento;
    }

    @Override
    public int hashCode() {
        int hash = 3;
        hash = 29 * hash + Objects.hashCode(this.id);
        return hash;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) {
```

```
            return true;
        }
        if (obj == null) {
            return false;
        }
        if (getClass() != obj.getClass()) {
            return false;
        }
        final Evento other = (Evento) obj;
        if (!Objects.equals(this.id, other.id)) {
            return false;
        }
        return true;
    }
}
```

TipoEvento(Enum)

```
public enum TipoEvento {
    /* TIPO DE EVENTO - CSS */
    PADRAO("Padrão", ""),
    URGENTE("Urgente", "urgente"),
    CANCELADO("Cancelado", "cancelado");

    private final String descricao;
    private final String css;

    private TipoEvento(String descricao, String css) {
        this.css = css;
        this.descricao = descricao;
    }

    public String getCss() {
        return css;
    }

    public String getDescricao() {
        return descricao;
    }

}
```

CustomScheduleEvent

```
public class CustomScheduleEvent implements ScheduleEvent {

    private String id;
    private String title;
    private Date startDate;
    private Date endDate;
    private String styleClass;
    private Object data;
    private String url;
    private String description;
    private boolean allDay;
    private boolean editable;

    public CustomScheduleEvent() {
    }

    public CustomScheduleEvent(String title, Date start, Date end, boolean allDay, Object data) {
        this.title = title;
        this.startDate = start;
        this.endDate = end;
        this.allDay = allDay;
        this.data = data;
    }

    public CustomScheduleEvent(String title, Date start, Date end, String styleClass, boolean allDay, Object data) {
        this.title = title;
        this.startDate = start;
        this.endDate = end;
        this.styleClass = styleClass;
        this.allDay = allDay;
        this.data = data;
    }

    @Override
    public String getId() {
        return id;
    }

    @Override
    public void setId(String id) {
        this.id = id;
    }

    @Override
    public String getTitle() {
        return title;
    }

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

    @Override
    public Date getStartDate() {
        return startDate;
    }

    public void setStartDate(Date startDate) {
        this.startDate = startDate;
    }

    @Override
    public Date getEndDate() {
        return endDate;
    }

    public void setEndDate(Date endDate) {
        this.endDate = endDate;
    }

    @Override
    public boolean isAllDay() {
        return allDay;
    }

    public void setAllDay(boolean allDay) {
        this.allDay = allDay;
    }

    @Override
    public String getStyleClass() {
        return styleClass;
```

```
        }

        public void setStyleClass(String styleClass) {
            this.styleClass = styleClass;
        }

        @Override
        public Object getData() {
            return data;
        }

        public void setData(Object data) {
            this.data = data;
        }

        @Override
        public String getUrl() {
            return url;
        }

        public void setUrl(String url) {
            this.url = url;
        }

        @Override
        public String getDescription() {
            return description;
        }

        public void setDescription(String description) {
            this.description = description;
        }

        @Override
        public boolean isEditable() {
            return editable;
        }

        public void setEditable(boolean editable) {
            this.editable = editable;
        }
    }
```

A classe CustomScheduleEvent é uma implementação de ScheduleEvent onde criamos um construtor que recebe todos os atributos como parâmetros pois não existe um construtor assim na implementação Default.

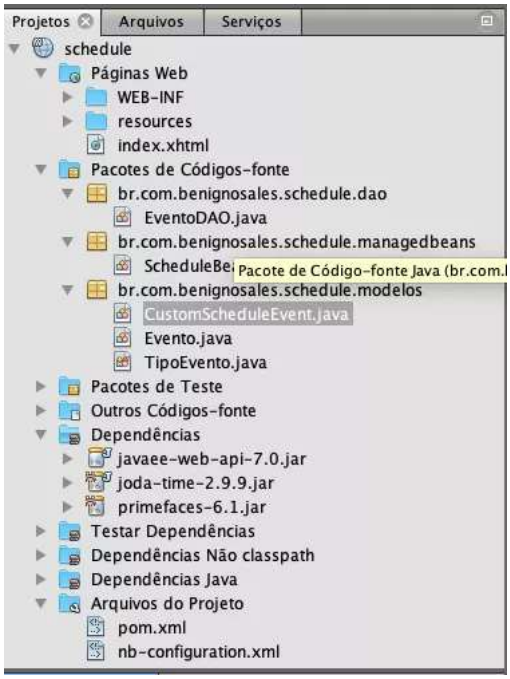
Para realizar a interação com a view, segue abaixo o bean que vai gerenciar a aplicação.

```
@Named
@ViewScoped
public class ScheduleBean implements Serializable {

    private ScheduleModel model;
    private Evento evento;
    private ScheduleEvent event;
    private List<ScheduleEvent> scheduleEvents;
    @Inject
    private EventoDAO eventoDAO;
    private static final Logger LOG = Logger.getLogger(ScheduleBean.class.getName());

    public ScheduleBean() {
        event = new CustomScheduleEvent();
        model = new DefaultScheduleModel();
        evento = new Evento();
    }

    @PostConstruct
    public void init() {
        if (this.model != null) {
            List<Evento> eventos = this.eventoDAO.listarTodos();
            if (this.scheduleEvents == null) {
                this.scheduleEvents = new ArrayList();
            }
            for (Evento eventoAtual : eventos) {
                ScheduleEvent newEvent = new CustomScheduleEvent(eventoAtual.getTitulo(), eventoAtual.getDataInicio(), eventoAtual.get
                if (!this.scheduleEvents.contains(newEvent)) {
                    newEvent.setId(eventoAtual.getId().toString());
                    this.scheduleEvents.add(newEvent);
                    this.model.addEvent(newEvent);
                }
            }
        }
    }

    public Evento getEvento() {
        return evento;
    }

    public void setEvento(Evento evento) {
        this.evento = evento;
    }

    public void salvar() {
        ScheduleEvent newEvent = new CustomScheduleEvent(this.evento.getTitulo(), this.evento.getDataInicio(), this.evento.getDataFim(
        if (evento.getId() == null) {
            model.addEvent(newEvent);
        } else {
            newEvent.setId(event.getId());
            model.updateEvent(newEvent);
        }
        eventoDAO.salvar(evento);
        FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_INFO, "Evento Salvo", "Evento Salvo");
        addMessage(message);
    }

    public void remover() {
        model.deleteEvent(event);
        eventoDAO.remover(evento);
        FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_INFO, "Evento Removido", "Evento Removido");
        addMessage(message);
    }

    public void onDateSelect(SelectEvent selectEvent) {
        this.evento = new Evento();
        Date dataSelecionada = (Date) selectEvent.getObject();
        DateTime dataSelecionadaJoda = new DateTime(dataSelecionada.getTime());
        this.evento.setDataInicio(dataSelecionada);
        this.evento.setDataFim(dataSelecionadaJoda.plusHours(1).toDate());
    }

    public void onEventSelect(SelectEvent selectEvent) {
        event = (CustomScheduleEvent) selectEvent.getObject();
        this.evento = (Evento) event.getData();
    }

    public void onEventResize(ScheduleEntryResizeEvent event) {
        FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_INFO, "Evento Redimensionado", "Dia:" + event.getDayDelta() + ",
        addMessage(message);
    }
```

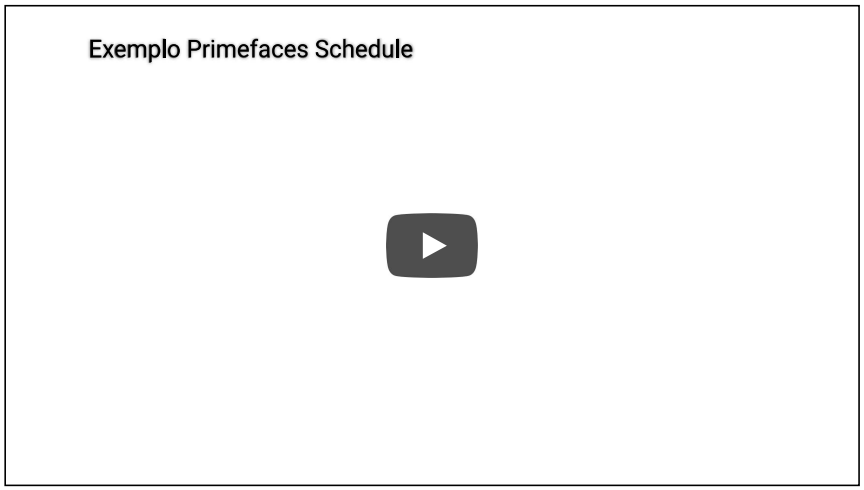
```
        private void addMessage(FacesMessage message) {
            FacesContext.getCurrentInstance().addMessage(null, message);
        }

        public TipoEvento[] getTiposEventos() {
            return TipoEvento.values();
        }
    }
```

Eu suprimi os getters e setters para otimizar o espaço da postagem. Temos o ScheduleModel, que vai espelhar a agenda na view e os onDateSelect e o onEventSelect que são os métodos chamados ao clicar em uma data para inserir um novo evento e para editar ou excluir um evento da agenda. Um detalhe importante que pode gerar problemas de NullPointer é que a lista de ScheduleEvents e o ScheduleModel não podem ser instanciados mais do que uma vez, do contrário, quando você for selecionar os eventos existentes, será exibido um erro de NullPointer.

Com os códigos e configurações acima, ficaremos com a seguinte estrutura de classes e dependências:



Pronto! O resultado você pode conferir no vídeo abaixo:



Espero que tenham gostado e, se puderem, compartilhem com os colegas programadores. Abraço!

Relacionado

Deixando seu sistema Responsivo com Primefaces (http://benignosales.com.br/tutorial/deixando-seu-sistema-responsivo-com-primefaces/)
25 de janeiro de 2018
Post similar

Compactação de PDF utilizando em Java com iText (http://benignosales.com.br/tutorial/compacta...de-pdf-utilizando-em-java-com-itext/)
11 de dezembro de 2018
Post similar

Criando uma Página de Login Completa com JSF e Primefaces (http://benignosales.com.br/tutorial/criando-uma-pagina-de-login-completa-com-jsf-e-primefaces/)
4 de setembro de 2017
Com 9 comentários

0 comentários em “Agenda completa com Primefaces Schedule”

1. *Antonio Augusto* disse:
   Seu comentário está aguardando moderação. Esta é uma pré-visualização, seu comentário ficará visível assim que for aprovado.

8 de abril de 2019 às 20:45 (http://benignosales.com.br/tutorial/agenda-completa-com-primefaces-schedule/#comment-12)
Muito bom. Me ajudou bastante o exemplo onde utilizou o calendar

Responder

Escreva uma resposta ou comentário

O seu endereço de e-mail não será publicado. Campos obrigatórios são marcados com *

Comentário

Nome *

E-mail *

Site

Notifique-me sobre novas publicações por e-mail.

Enviar Comentário