

# **UI5 para desenvolvedores SAP/ABAP**

Fabio Pagoti

# UI5 para desenvolvedores SAP/ABAP

Fabio Pagoti

Esse livro está à venda em <http://leanpub.com/ui5>

Essa versão foi publicada em 2015-04-10



This is a [Leanpub](#) book. Leanpub empowers authors and publishers with the Lean Publishing process. [Lean Publishing](#) is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

©2015 Fabio Pagoti

# **Tweet Sobre Esse Livro!**

Por favor ajude Fabio Pagoti a divulgar esse livro no [Twitter!](#)

O tweet sugerido para esse livro é:

Acabo de comprar o livro "UI5 para desenvolvedores SAP/ABAP" no @Leanpub

A hashtag sugerida para esse livro é [#ABAP2UI5](#).

Descubra o que as outras pessoas estão falando sobre esse livro clicando nesse link para buscar a hashtag no Twitter:

<https://twitter.com/search?q=#ABAP2UI5>

# Conteúdo

<b>Introdução ao UI5</b>	<b>1</b>
O que significa UI5?	1
Desenvolvimento Front End vs Back End	1
E o back end?	2
SAPUI5 vs OpenUI5	2
Onde o UI5 é usado?	3
<b>Ambiente de Desenvolvimento</b>	<b>5</b>
Baixando o OpenUI5	5
Servidores Web	6
Servidores Web em plataformas SAP	8
Servidores Web em outras plataformas	16
Hospedagens gratuitas	20
IDEs	21
Navegadores	24
Outras ferramentas importantes	24
<b>Criando um Hello World em UI5</b>	<b>26</b>
Estrutura básica de uma aplicação UI5	28
HTML e <div> principal	28
Bootstrap	30
Minificação	32
Código UI5 e Renderização	32
<b>Usando o Eclipse e o Paradigma MVC</b>	<b>35</b>
Criando um projeto UI5 através do Eclipse	35
Estrutura de uma aplicação UI5 usando MVC	39
<b>Namespaces</b>	<b>45</b>
Dependências entre <i>namespaces</i>	47
sap.ui.Device - Capturando informações sobre dispositivo	48
Bibliotecas de terceiros	55
<b>Depurando um projeto em UI5</b>	<b>59</b>

## CONTEÚDO

<b>Controles de UI</b>	<b>62</b>
Herança de controles	62
Elementos versus Controles	63
ManagedObject	64
Tipos de propriedades de um ManagedObject	64
<b>Controles Simples</b>	<b>68</b>
TextView	71
FormattedTextView	72
Label	72
HTML	73
Button	74
ToggleButton	75
Link	76
Image	77
ImageMap	77
Outros controles simples	79
<b>Controles de Value Holders</b>	<b>80</b>
TextField	82
TextArea	85
PasswordField	85
ValueHelpField	86
DatePicker	87
ComboBox	87
AutoComplete	89
DropDownBox	90
ListBox	92
InPlaceEdit	94
SearchField	95
CheckBox	95
TriStateCheckBox	96
RadioButton	98
RadioButtonGroup	99
Slider	100
RangeSlider	101
RatingIndicator	102
<b>Controles de Leiaute (Layout)</b>	<b>103</b>
Vertical Layout	105
Horizontal Layout	106
Horizontal Divider	108
Panel	109

## CONTEÚDO

Border Layout . . . . .	111
Matrix Layout . . . . .	113
Splitter . . . . .	116

# Introdução ao UI5

## O que significa UI5?

Antes de mais nada, vamos dar nomes aos bois. UI5 é simplesmente um nome curto para *UI development toolkit for HTML5*. Trocando em miúdos, uma nada singela caixa de ferramentas para construção de interfaces gráficas no padrão HTML5.

Logo, quando falamos de UI5 estamos falando de desenvolvimento *front end*.

## Desenvolvimento Front End vs Back End

Se você já trabalhou com desenvolvimento web fora do mundo SAP, seja com ASP, PHP, .NET, Java etc, tenho certeza que os termos *front end* e *back end* são muito claros para você.

Todavia, não podemos descartar o fato de que muitas das pessoas que estão começando a se aventurar no mundo UI5 tem uma única passagem no mundo SAP, principalmente com a linguagem de programação ABAP (eu mesmo sou um exemplo). Apesar de serem pessoas de carne e osso que trabalham com um sistema empresarial, estes termos acabam não sendo usados. Não os culpe... há muitos outros termos na esfera SAP.

No mundo não-SAP, desenvolvedores podem ser divididos em dois grandes grupos: *front end* e *back end*. O termo *end* tem muito a ver com a arquitetura da web, onde temos servidores (computadores que hospedam sites, por exemplo) e clientes (o navegador da sua máquina, por exemplo). Portanto temos duas pontas que se comunicam através de algum meio formando uma rede.

Grosseiramente falando, desenvolvedores *front end* trabalham com a parte “que o usuário vê”. Em outras palavras, são desenvolvedores que tem em seus currículos palavras como HTML, CSS, Javascript, jQuery, SVG e Ajax. Na verdade, estes profissionais nem sempre são desenvolvedores originalmente. Designers gráficos são pessoas que sabem como levar a interface gráfica de uma aplicação a outro patamar e isso pode vir a ser feito via linhas de código. Se você for hoje um desenvolvedor *front end* você pode até se auto-intitular um “artista da era digital”.

Você não estaria errado se falasse que desenvolvedores *back end* são o contrário dos desenvolvedores *front end*. Mas para ilustrar o que isso vem a ser pense nos desenvolvedores preocupados com banco de dados, tabelas, SQL, complexidade de algoritmos, regra de negócio, UML. Desenvolvedores *back end* geralmente tem outras palavras-chave em seus currículos com PHP, Java, .NET, PHP, NodeJS, algum sistema de banco de dados, etc. Se você for um hoje um desenvolvedor *back end* você pode até se auto-intitular um “matemático da era digital”.

Ambos são profissionais que trabalham em conjunto pois em uma aplicação web os dados exibidos para um usuário no *front end* tipicamente são armazenados no *back end*.

E por que esta distinção? Porque simplesmente a vida é muito curta para se aprender tudo. Fazer esta distinção faz as pessoas terem um foco profissional mais definido. Mas é claro que existem desenvolvedores que são bons nos dois assuntos mas a bagagem técnica para chegar lá é bem maior. Leonardo da Vinci era artista, matemático e muitas outras coisas. Com certeza ele usou de sua criatividade para fins matemáticos e sua matemática para tornar suas obras mais próximas da perfeição. Desenvolvedor *full stack* é um termo bastante usado para desenvolvedores que acabam conhecendo **pelo menos um pouco de quase tudo**.

Estes termos são mais importantes para alguém que está recrutando um desenvolvedor do que para os desenvolvedores em si. Independente qual o seu foco, ambos lados podem ser considerados mundos (quicá universos) a parte. O aprendizado em qualquer um deles será sempre constante para quem for interessado.

## De que lado está o desenvolvedor ABAP?

ABAP é uma linguagem de quarta geração. Seu propósito é criar aplicações de negócios e ponto final. No ABAP é muito fácil criar uma tela ou fazer uma seleção no banco de dados. Tão fácil a ponto de o mesmo profissional ser responsável pelas duas pontas. Por isso no mundo ABAP não existe esta distinção. O mesmo desenvolvedor ABAP que cria um componente Web Dynpro (*front end*) constrói as classes ou módulos de função (*back end*) que contém a regra de negócio.

Mas se você é um desenvolvedor ABAP hoje, não se surpreenda se daqui um tempo as vagas de emprego exigirem de você conhecimentos em UI5. Espero que você esteja lendo este livro antes que isso aconteça para estar preparado.

## E o back end?

Você pode se perguntar: “Quer dizer que este livro não abordará nada relacionado a *back end*?” O foco principal deste livro é o UI5 e portando a parte de *front end* de aplicações que usam esta biblioteca. Mas não é ideal separar totalmente os dois lados pois um conversa com outro. Por isso sempre que necessário abordaremos um pouco a parte de *back end* que é interessante para os nossos fins. Se você é familiar ao ecossistema SAP já lhe adianto quem faria parte do *back end*: NetWeaver, Gateway, Hana e HCP são apenas alguns exemplos de produtos da SAP que correspondem a parte *back end* de aplicações.

## SAPUI5 vs OpenUI5

UI5 é um termo curto e um pouco genérico. Isso porque na verdade a SAP não possui apenas um *toolkit* de desenvolvimento para HTML5 mas sim dois. São eles o SAPUI5 e o OpenUI5. Quando

se diz “UI5” estamos querendo dizer qualquer um deles pois a principal diferença entre ambos é comercial e não técnica.

A primeira biblioteca para tal finalidade criada foi o SAPUI5. Ela foi criada em meados de 2011 e é entregue juntamente com alguns produtos da SAP como o NetWeaver, Hana e Hana Cloud Platform. O SAPUI5 não tem nenhum custo adicional para os clientes da SAP mas para ter alguma destas plataformas uma boa quantidade de dinheiro já foi gasta. O SAPUI5 não está disponível para download abertamente.

O OpenUI5 é a versão *open source* do SAPUI5. Por questões legais e comerciais a SAP preferiu dar um nome novo a última. O OpenUI5 segue a licença *Apache 2.0* que permite que a biblioteca seja usada, distribuída e alterada.

Tecnicamente falando, apesar de quase idênticas as duas bibliotecas divergem. A principal diferença entre ambas é que o SAPUI5 possui uma API para criação de gráficos e o OpenUI5 não (pelo menos até o momento). Todavia isso não quer dizer que não seja possível criar gráficos em uma aplicação OpenUI5. Isso só quer dizer que será necessário utilizar uma biblioteca a parte para tal finalidade.

Um exemplo de biblioteca para criação de gráficos que vem ganhando fama a cada dia é o D3 (*Data Driven Documents*). Mas é importante dizer que existem muitas outras. O D3 acaba sendo um pouco mais especial pois ele é entregue juntamente com o OpenUI5, o que significa que você pode utilizá-lo com um esforço um pouco menor nas suas futuras aplicações.

Existem outras bibliotecas javascript que são entregues juntamente com o UI5 que serão abordadas mais adiante.

No mais, ambas bibliotecas são praticamente idênticas e dividem a mesma base de código. Como o OpenUI5 está disponível publicamente de forma gratuita, usaremos ele no livro. Porém saiba que aprendendo um você pode se considerar tão bom quanto no outro.



Para saber mais sobre as diferenças entre o OpenUI5 e o SAPUI5 leia o blog de Andreas Kunz [What is OpenUI5 / SAPUI5](http://scn.sap.com/community/developer-center/front-end/blog/2013/12/11/what-is-openui5-sapui5)<sup>1</sup>

## Onde o UI5 é usado?

Boa parte do que a SAP desenvolve hoje em dia é feito em UI5 na parte do front end. Se você já ouviu falar em alguns nomes como Gateway, SUP, Hana, Fiori ou HCP saiba que o UI5 está totalmente inserido nestes contextos.

## O ABAPer da próxima geração

O desenvolvedor ABAP da próxima geração, que começou com o advento do Hana, deve ter conhecimentos em desenvolvimento Web. Saber programar em Web Dynpro não requer profundos

---

<sup>1</sup><http://scn.sap.com/community/developer-center/front-end/blog/2013/12/11/what-is-openui5-sapui5>

conhecimentos Web como padrões, protocolos, arquiteturas ou detalhes de navegadores. No caso do UI5, é diferente. É preciso entender HTML, CSS, Javascript e ser familiar ao jQuery para então começar a engatinhar no UI5. E é para este caminho que o ABAPer da próxima geração deve seguir.

No livro “Next generation ABAP development / Rich Heilman, Thomas Jung. — 2nd ed.”, que foi escrito em 2011 (antes do UI5), dos 20 capítulos do livro 8 são de alguma forma relacionados a desenvolvimento Web (MVC, web services, Web Dynpro, BSP). O livro “ABAP to the Future / Paul Hardy” já aborda UI5. E se ainda não está convencido, já há um livro oficial da SAP Press exclusivo de UI5: “Getting Started with SAPUI5 / Miroslav Antolovic”.



Como autor, não vejo problema algum de citar livros que podem ser entendidos como “concorrentes” do livro que você lê neste prezado momento. Afinal, meus primeiros passos no UI5 foram dados graças ao Miroslav e seu excelente livro. O foco deste livro e dos outros supracitados é diferente mas se você está na dúvida, eu recomendo ler todos.

# Ambiente de Desenvolvimento

## Baixando o OpenUI5

Conforme dito anteriormente, usaremos o OpenUI5 ao invés do SAPUI5 para os exemplos, exercícios e aplicações presentes no livro. Para fazer o download da biblioteca, visite a página oficial do OpenUI5:

[OpenUI5.org](http://OpenUI5.org)<sup>2</sup>

Abra a página “Download”. Você irá notar que há duas pacotes passíveis de serem baixados para a versão mais recente: o pacote de Runtime e a SDK (sigla para *Software Development Toolkit*).

### Pacote de Runtime

O pacote de Runtime contém toda a biblioteca do OpenUI5 que você pode vir a utilizar em uma aplicação, seja ela voltada a desktop ou mobile.

### Pacote com SDK

O pacote de SDK é mais completo. Além de incluir o pacote de *runtime*, o SDK contém a documentação oficial do OpenUI5, guia do desenvolvedor, além de exemplos de códigos, aplicações de demonstração e informações detalhadas sobre versionamento.



A SDK também está disponível online no endereço abaixo. Apesar disso, é recomendável que você faça o download da SDK para trabalhar localmente pois é muito mais rápido do que acessar a versão online.

<https://openui5.hana.ondemand.com/#content/Overview.html><sup>3</sup>

---

<sup>2</sup><http://OpenUI5.org>

<sup>3</sup><https://openui5.hana.ondemand.com/#content/Overview.html>

## Servidores Web

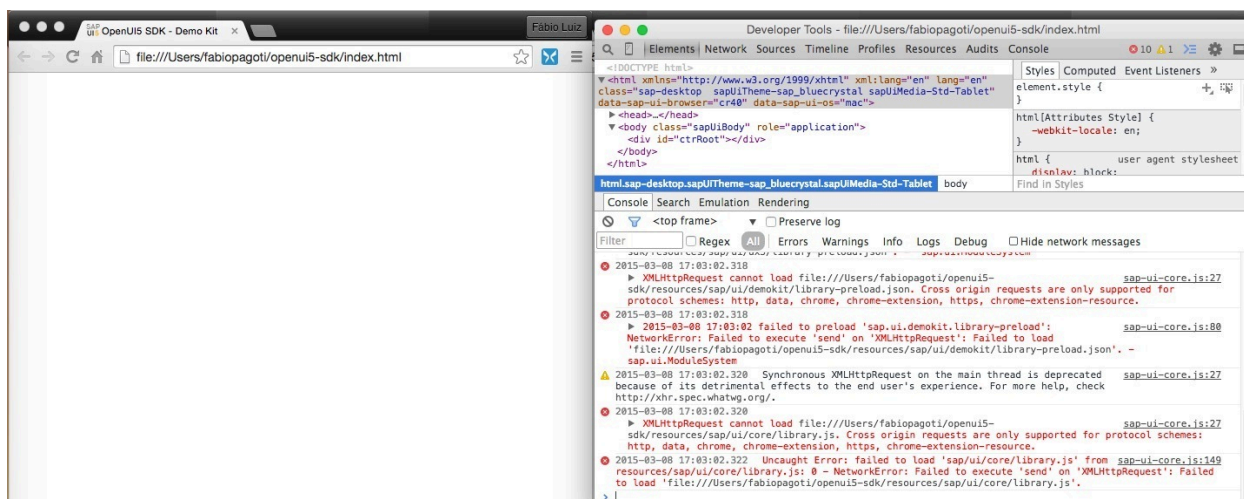
Para explicar a importância de servidores web, precisamos falar um pouco sobre os navegadores.

Além de exibir páginas na internet, os navegadores também são capazes de acessar a estrutura de diretórios do seu sistema operacional (se você nunca fez isso, experimente digitar “file://” na barra de endereço). Além disso, é possível abrir páginas HTML diretamente a partir do sistema de arquivos (por exemplo, file://C:/app/index.html).

O UI5 se baseia em páginas estáticas para criação de aplicações. Em outras palavras, todo conteúdo criado pelo desenvolvedor é definido estaticamente em arquivos .html, .css e .js. Em um primeiro momento você poderia imaginar que durante a construção das aplicações em UI5 bastaria abrir tais arquivos .html diretamente no navegador utilizando o sistema de arquivos. Infelizmente isso nem sempre irá funcionar. Há algumas situações nas quais seu navegador não permitirá que as coisas saiam como planejado.

Para citar uma destas ocasiões, uma vez que sua aplicação interaja com algum servidor remoto, por exemplo fazendo uma requisição HTTP GET ou POST, o navegador não permitirá a interação do arquivo local (sendo acessado através de “file:/”) com este servidor. Por uma questão técnica e de segurança relacionada ao funcionamento dos navegadores, isso não é permitido. Há alguns navegadores que disponibilizam configurações avançadas ou até mesmo add-ons para que este tipo de bloqueio não seja realizado, mas tais alterações não são recomendáveis por questões de segurança.

Você pode fazer este teste abrindo o arquivo index.html da SDK do OpenUI5 a partir do seu navegador. Este arquivo está presente na raiz da SDK uma vez que você extraia seu conteúdo. Como você estará usando o sistema de arquivos, o navegador não exibirá o documento e irá gerar log de erros que podem ser vistos usando a ferramenta de desenvolvedor no Google Chrome.



**Erro: Requisições para outras origens não pode ser feito a partir do sistema de arquivos**

Para contornar este problema relacionado a cerne dos navegadores, é necessário a utilização de um servidor web, que pode ser remoto ou local.

Se o servidor web for remoto, isso significa que quando você testar sua aplicação, você com certeza estará usando um domínio e não a estrutura de diretórios local de seu computador - o que não impede o navegador de fazer seu trabalho.

Caso o servidor web seja local, além de usar seu computador para criar os arquivos .html, .css, .js, você também o usará para hospedar sua aplicação - da mesma forma que um serviço de hospedagem faz com algum website que você tenha.

Usando um servidor web local, você pode acessar os arquivos estáticos no navegador através do seu próprio endereço IP ou *hostname*. Na prática, não é necessário descobrir qual o endereço IP de seu computador ou criar um *hostname* para ele. O termo *localhost* é um *hostname* pré-definido pelos sistemas operacionais e que significa “sua própria máquina”. Logo, você poderá acessar os arquivos usando um endereço como o abaixo:

localhost:8080/app/index.html



8080 é um exemplo de porta muito utilizada por alguns servidores web. O número pode ser praticamente qualquer um porém os números 8000, 8080 e 8888 são mais comuns de serem usados.



Se você adora redes e prefere usar um endereço IP, você pode substituir “localhost” pelo IP 127.0.0.1

Neste momento, algumas perguntas pairam no ar:

- Devo utilizar um servidor web remoto ou local?

Você pode usar um outro outro ou até mesmo ambos (local para desenvolvimento e remoto funcionando como um ambiente produtivo). A melhor escolha vai depender do ambiente e da plataforma na qual você está desenvolvendo.

- Quais os servidores web remotos disponíveis?

Existe realmente uma infinidade. Uma rápida busca por “hospedagem web” no Google resultará em milhões de resultados. Apesar de ser possível usar UI5 para criação de sites em nada relacionados a plataformas SAP, seus propósito não é esse. Logo, as duas plataformas na esfera SAP mais famosas que atuam como um servidor web remoto são o SAP HANA e o SAP NetWeaver. Ambas são plataformas capazes de responder a requisições HTTP, e esta é a tarefa de um servidor web! É totalmente possível desenvolver suas aplicações diretamente nestes servidores remotos e portanto não há necessidade de se criar um servidor web local nestes casos.

- **Como tornar minha máquina um servidor local?**

Mesmo que a aplicação que você esteja construindo venha a ser executada na plataforma SAP HANA ou SAP NetWeaver, nada impede que você crie um servidor local para funcionar como um ambiente de desenvolvimento. Para transformar seu computador em um servidor web você precisa instalar algum software ou utilizar alguma biblioteca para tal finalidade.

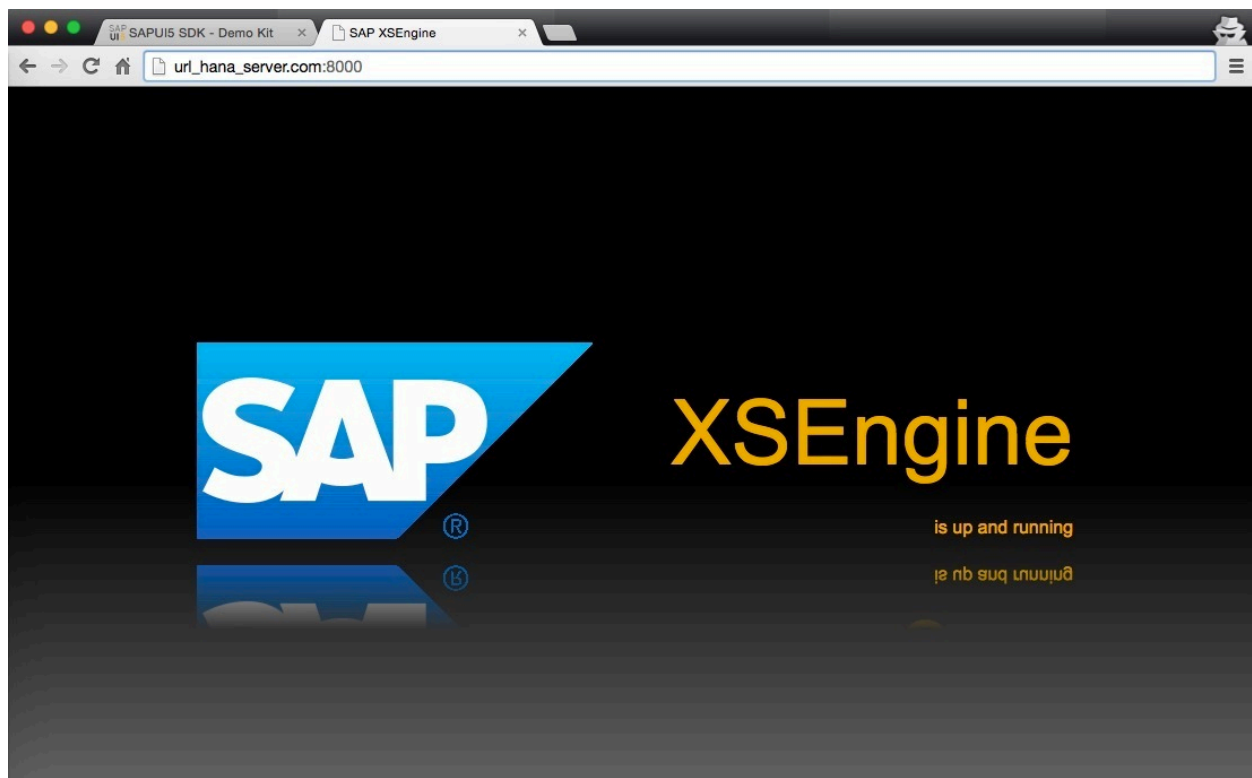
- **Quais as opções que tenho para transformar meu computador em um servidor web?**

Várias. Algumas mais simples outras mais complexas. Veremos mais adiante o processo de preparação em alguns exemplos bem conhecidos.

## **Servidores Web em plataformas SAP**

### **HANA**

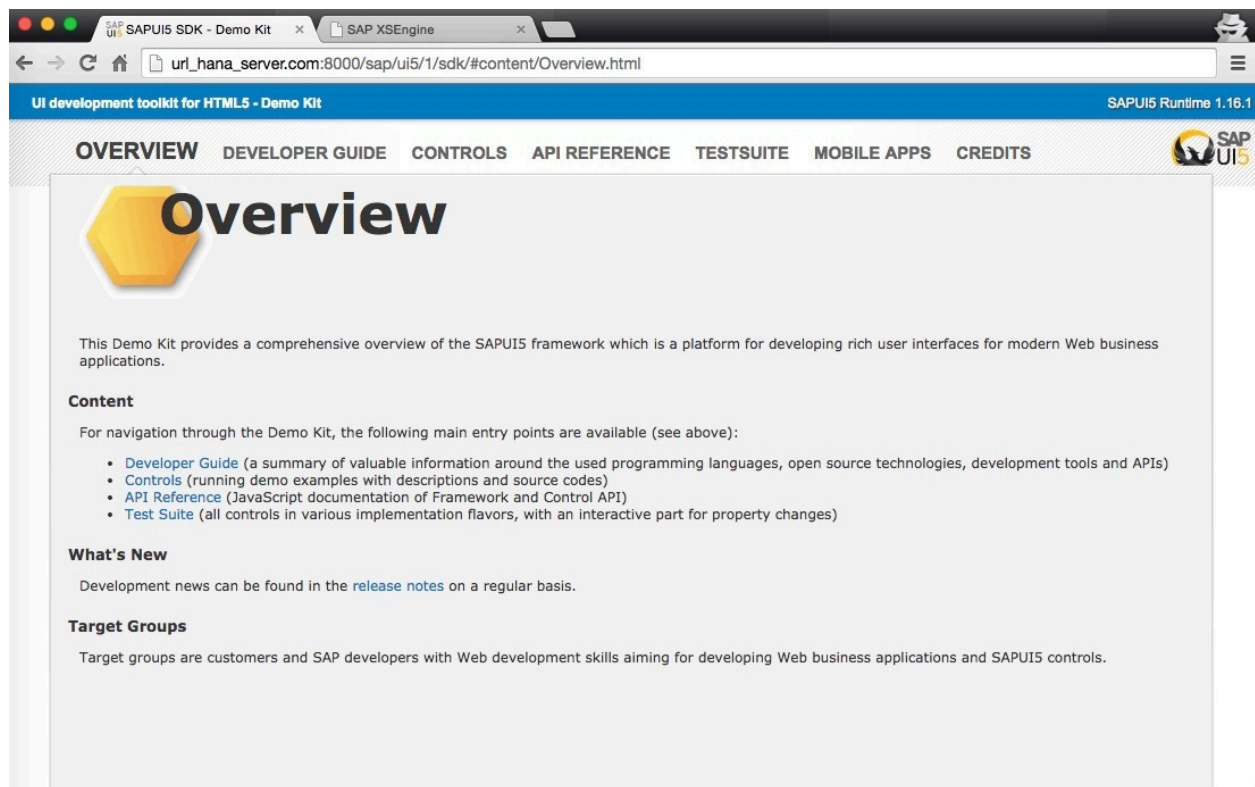
Assunto mais comentado no ecossistema SAP há alguns anos, o SAP HANA é mais do que um banco de dados *in-memory*, é uma plataforma. Um das capacidades desta plataforma (tanto em sua versão *on-premise* quando em *núvem*) é a de funcionar como o servidor web. O SAP HANA permite que aplicações sejam construídas diretamente nele usando uma IDE na *núvem* chamada SAP Web IDE (discutida mais adiante).



Página raiz do servidor web embutido na plataforma SAP HANA

É importante lembrar que a SDK do SAPUI5 é entregue juntamente com algumas plataformas SAP. O HANA é um bom exemplo. Nele, o SDK do SAPUI5 será sempre disponibilizado de acordo com o caminho abaixo:

*urlServidorHana.com:porta/sap/ui5/1/sdk/*

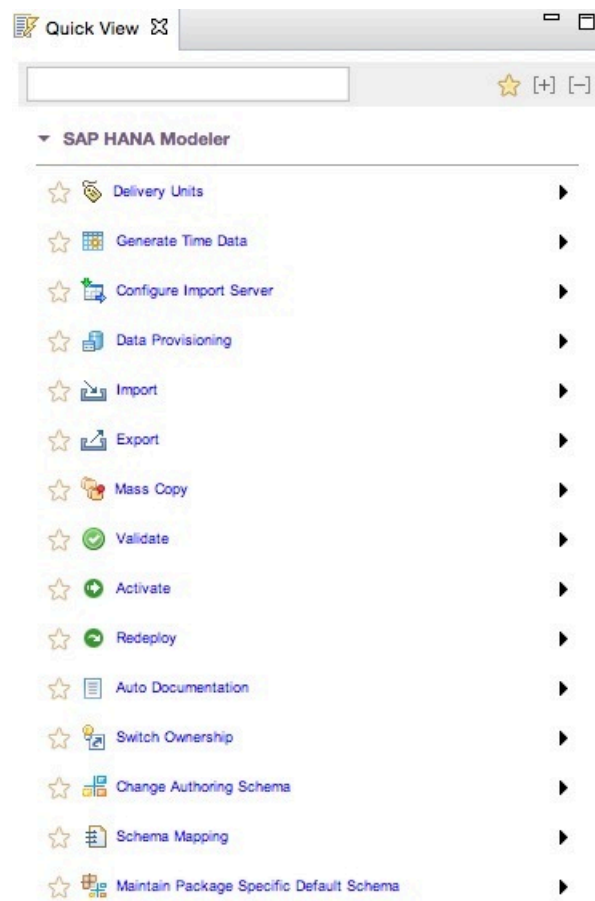


### SDK do SAPUI5 instalada em um servidor HANA

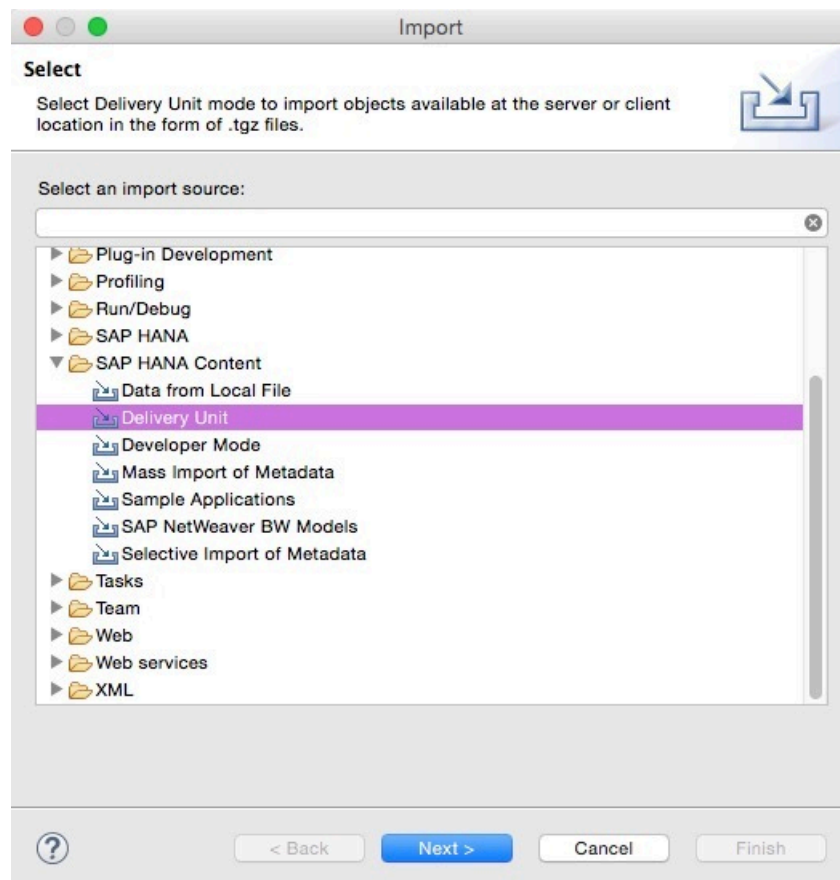


Ao usar o SDK nas suas aplicações, use sempre URLs relativas (exemplo: “/sap/ui5/1/sdk/resources/sap-ui-core.js”) pois cada servidor terá seu próprio domínio mas o caminho para a SDK será sempre o mesmo.

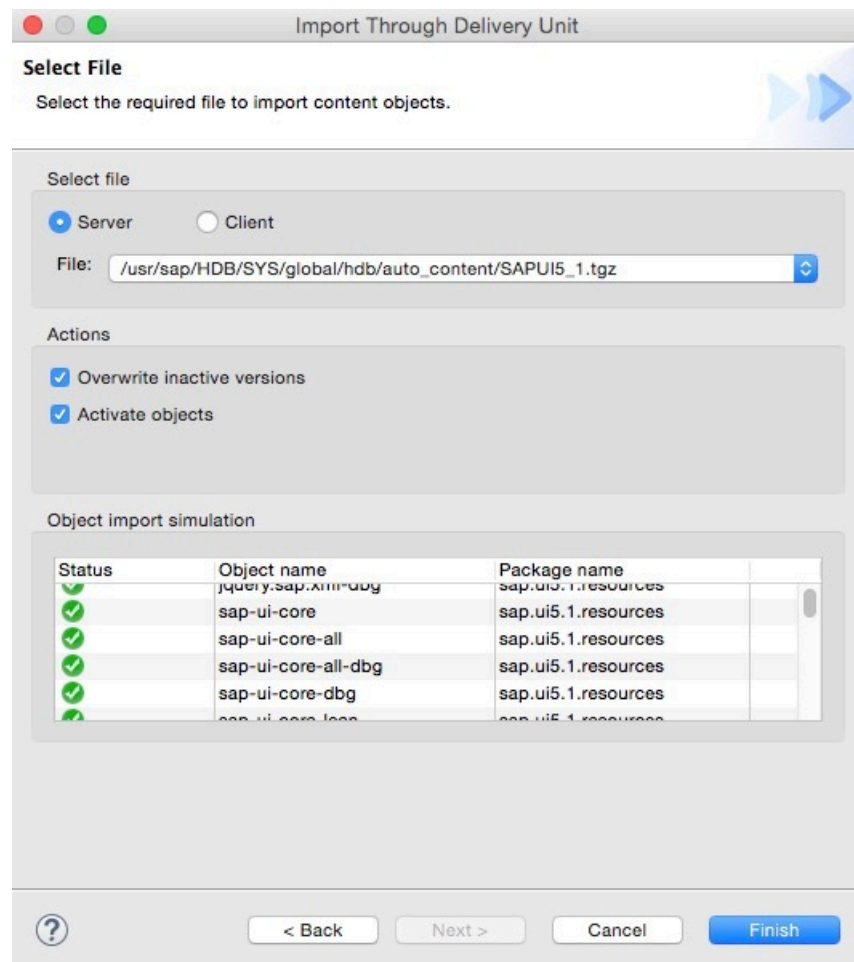
Dependendo do *Support Package* do HANA, o SAPUI5 pode vir pronto para ser usado ou empacotado em uma *delivery unit* que deve ser importada para que o mesmo seja instalado. No último caso, a importar da *delivery unit* pode ser feita utilizando o HANA Studio (Eclipse).



Importando uma *Delivery Unit* contendo SAPUI5 - Passo 1



Importando uma *Delivery Unit* contendo SAPUI5 - Passo 2

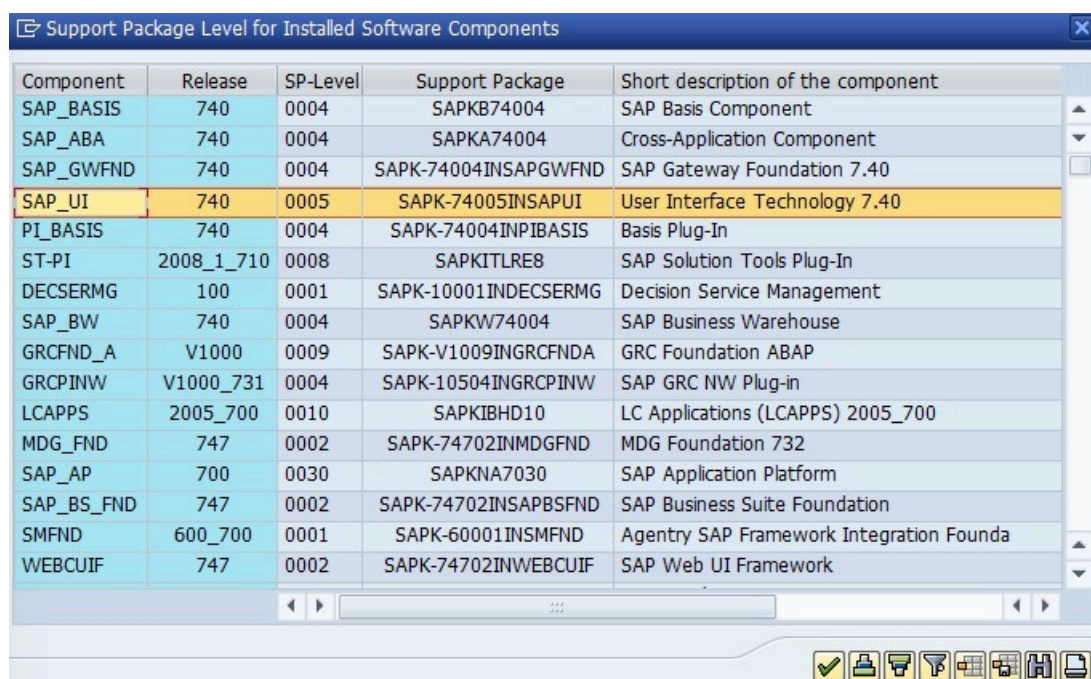


Importando uma *Delivery Unit* contendo SAPUI5 - Passo 3

## NetWeaver

O SAP NetWeaver (ABAP, Java ou em nuvem) também atua como um servidor web. Os serviços disponíveis estão cadastrados na famosa transação SICF.

O UI5 é entregue como um add-on de ABAP, o que significa que não pode ser desinstalado. Uma maneira simples de verificar se o UI5 está instalado em algum servidor é procurando pelo componente "SAP\_UI" na lista de componentes de software. Esta lista pode ser acessada usando o SAP GUI indo em Sistemas > Status.



Component	Release	SP-Level	Support Package	Short description of the component
SAP_BASIS	740	0004	SAPKB74004	SAP Basis Component
SAP_ABA	740	0004	SAPKA74004	Cross-Application Component
SAP_GWFND	740	0004	SAPK-74004INSAPGWFND	SAP Gateway Foundation 7.40
<b>SAP_UI</b>	<b>740</b>	<b>0005</b>	<b>SAPK-74005INSAPUI</b>	<b>User Interface Technology 7.40</b>
PI_BASIS	740	0004	SAPK-74004INPIBASIS	Basis Plug-In
ST-PI	2008_1_710	0008	SAPKITLRE8	SAP Solution Tools Plug-In
DECSERMG	100	0001	SAPK-10001INDECSERMG	Decision Service Management
SAP_BW	740	0004	SAPKW74004	SAP Business Warehouse
GRCFND_A	V1000	0009	SAPK-V1009INGRCFND	GRC Foundation ABAP
GRCPINW	V1000_731	0004	SAPK-10504INGRCPINW	SAP GRC NW Plug-in
LCAPPS	2005_700	0010	SAPKIBHD10	LC Applications (LCAPPS) 2005_700
MDG_FND	747	0002	SAPK-74702INMDGFND	MDG Foundation 732
SAP_AP	700	0030	SAPKNA7030	SAP Application Platform
SAP_BS_FND	747	0002	SAPK-74702INSAPBSFND	SAP Business Suite Foundation
SMFND	600_700	0001	SAPK-60001INSMFND	Agentry SAP Framework Integration Founda
WEBCUIF	747	0002	SAPK-74702INWEBCUIF	SAP Web UI Framework

### Componente SAP\_UI

O *Support Package* mínimo do NetWeaver necessário para a instalação do componente SAP\_UI5 varia conforme a versão da plataforma.

Versão NetWeaver	Support Package
700	21
701	06
702	06
731	04

Quando a instalação é realizada um serviço é gerado na transação SICF no caminho abaixo:

sap/public/bc/ui5\_ui5/

**Maintain service**

Create Host/Service

Filter Details

Virtual Host:  Service Path:

ServiceName:

Description:

Lang.:  Ref.Service:

Apply Reset Fine-Tune

Virtual Hosts / Services	Documentation	Referenz Service
default_host	VIRTUAL DEFAULT HOST	
sap	SAP NAMESPACE; SAP IS OBLIGED NOT TO DELIVER ANY SER...	
option	RESERVED SERVICES AVAILABLE GLOBALLY	
public	PUBLIC SERVICES	
bc	Basis Tree (Basis Functions)	
abap	Services from NW Foundation ABAP	
bsp	BSP Design2008	
icf	Internet Communication Framework	
icons	SAP Icons	
icons_rtl	Icons RTL	
its	Internet Transaction Server (ITS)	
NWDEMO_MODEL	NW demo model	
NW_ESH_TST_AUTO	NW Enterprise Search (Test Automation)	
pictograms	Pictograms	
sec	Security Node	
themes	UI Theming Repository: Public Access (ro)	
tmp_wd_mimes	Temporary Folder for Web Dynpro MIMES (MIME Deployment ...	
trex	TREX	
ui2	Entry Point for UI2 Services	
ui5_ui5	SAPUI5 library called via HTTP out of MIME repository	
ILMRWC	Reporting work center - ILM Cockpit	
ur	Unified Rendering	
wdtracetool	Web Dynpro Trace Tool	
webdynpro	Web Dynpro MIME Handling	
webicons		
workflow	Business Workflow - Public Services	
bsp	BUSINESS SERVER PAGES (BSP)	
BusinessSuite	Business Suite	
ES	Enterprise Search	
HRPDV	HR PD Visualization	
HRRenewal	HR Renewal	
icf_check	Testing ICF Environment	

### Transação SICF - Serviço UI5

O Eclipse (IDE discutida mais adiante) pode ser usado para criar aplicações SAPUI5 em um servidor ABAP (NetWeaver). Neste caso quando o desenvolvedor testa a sua aplicação ele o estará fazendo no ambiente de desenvolvimento que nada mais é que um servidor web remoto.



Para mais informações de como instalar o SAPUI5 nas diferentes plataformas SAP, leia a OSS Note 1747308.

## Servidores Web em outras plataformas

Além de utilizar um servidor remoto, você pode tornar seu próprio computador em um servidor web. Abaixo são listados alguns softwares que são capazes de realizar tal tarefa. Todos eles podem ser usados para criar um servidor web produtivo mas aqui são listados como forma de criar um ambiente de desenvolvimento local.

### Mongoose

O **Mongoose** - Cesanta ([code.google.com/p/mongoose/](https://code.google.com/p/mongoose/))<sup>4</sup> provavelmente é o servidor web mais simples de ser instalado e um dos mais leves existente. Está disponível Windows, MacOS e Linux.



Para fazer o download do Mongoose, use o endereço <http://cesanta.com/mongoose.shtml><sup>5</sup>

Por padrão, o Mongoose transforma o diretório em que ele está situado na pasta raiz do servidor web. A porta padrão utilizada é a 8080. Abrindo o navegador com o endereço *localhost:8080* é possível ver os arquivos diretamente abaixo deste diretório.

É possível mudar o diretório padrão, porta e outras propriedades do Mongoose através da página de edição de configuração (clicando com o direito no ícone do Mongoose na barra de aplicações). Com isso será aberta a página abaixo

---

<sup>4</sup><https://code.google.com/p/mongoose/>

<sup>5</sup><http://cesanta.com/mongoose.shtml>

The screenshot shows the Mongoose web server v.5.5 [FREE EDITION] settings page. The browser address bar shows 'localhost:8080/\_\_\_mg\_mc#settings'. The page has three tabs: 'Settings' (active), 'Traffic Hexdump', and 'Requests/Errors/Latency Stats'. Below the tabs, it says 'Config file location: /Applications/mongoose.conf [does not exist]'. The 'Current Settings' section is divided into two columns of input fields. The left column includes 'access\_control\_list', 'auth\_domain' (mydomain.com), 'cgi\_pattern [pro]' (\*.cgi\$|\*.pl\$|\*.php\$), 'document\_root' (/Users/fablopagoti/openui5-sdk), 'enable\_proxy', 'global\_auth\_file', 'hexdump\_file [pro]', 'listening\_port' (8080), and 'ssi\_pattern [pro]' (\*.shtml\$|\*.shtm\$). The right column includes 'access\_log\_file', 'cgi\_interpreter [pro]', 'dav\_auth\_file [pro]', 'enable\_directory\_listing' (yes), 'extra\_mime\_types', 'hide\_files\_patterns', 'index\_files' (index.html, index.htm, index.shtml, index.shtm), 'run\_as\_user', and 'url\_rewrites'. At the bottom, there are two buttons: 'Delete config file and reset all settings to default values' and 'Save settings to the config file'. A link 'configuration options reference' is also present. At the very bottom, it says 'Have suggestion or feature request? Send us a message'.

localhost:8080/\_\_\_mg\_mc#settings

## Mongoose web server v.5.5 [FREE EDITION]

[Settings](#) [Traffic Hexdump](#) [Requests/Errors/Latency Stats](#)

Config file location: /Applications/mongoose.conf [does not exist]

Current Settings

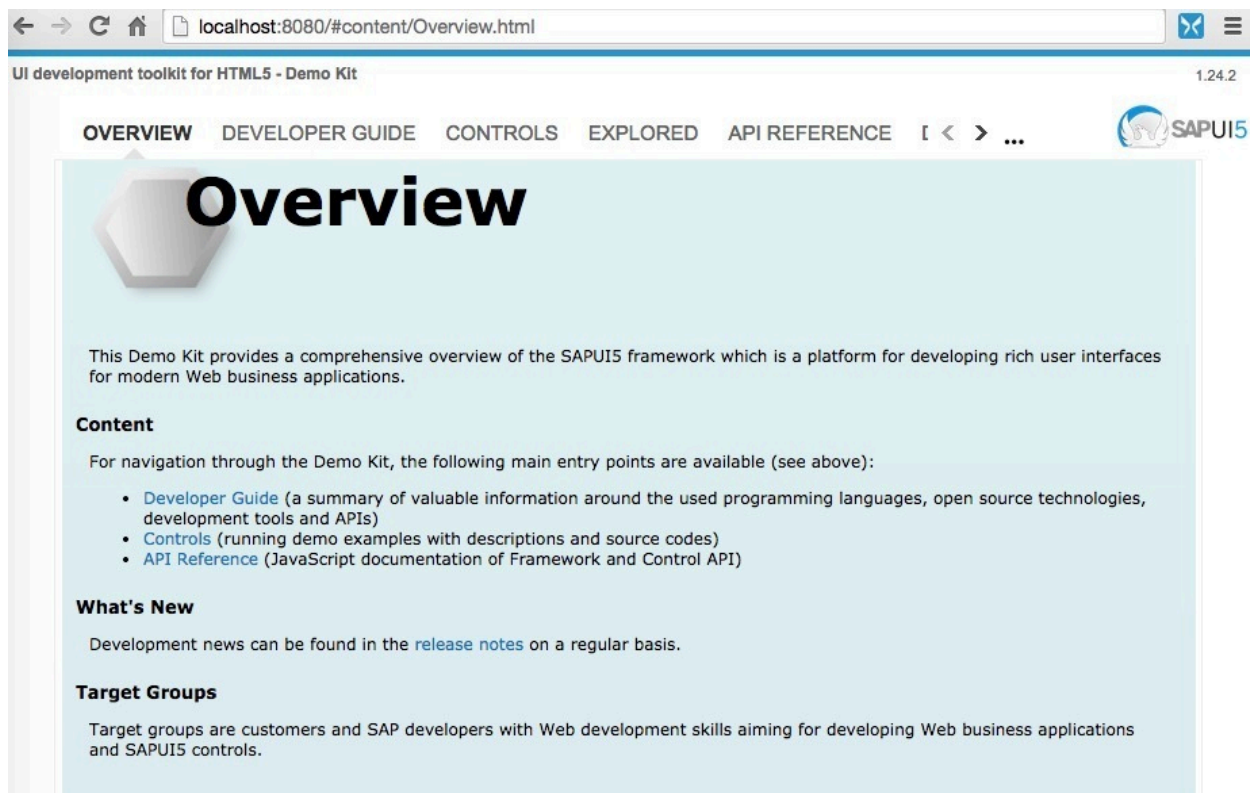
access_control_list		access_log_file	
auth_domain	mydomain.com	cgi_interpreter [pro]	
cgi_pattern [pro]	*.cgi\$ *.pl\$ *.php\$	dav_auth_file [pro]	
document_root	/Users/fablopagoti/openui5-sdk	enable_directory_listing	yes
enable_proxy		extra_mime_types	
global_auth_file		hide_files_patterns	
hexdump_file [pro]		index_files	index.html,index.htm,index.shtml,inde
listening_port	8080	run_as_user	
ssi_pattern [pro]	*.shtml\$ *.shtm\$	url_rewrites	

[Delete config file and reset all settings to default values](#) [Save settings to the config file](#) [configuration options reference](#)

Have suggestion or feature request? [Send us a message](#)

Configuração do Mongoose - [http://localhost:8080/\\_\\_\\_mg\\_mc#settings](http://localhost:8080/___mg_mc#settings)

Experimente mudar o diretório padrão para o diretório no qual você extraiu o SDK do OpenUI5. Você poderá então ter acesso a documentação da biblioteca localmente.



SDK do OpenUI5 acessada localmente

## MAMP

O MAMP [mamp.info] é um servidor web de fácil utilização. O tamanho da aplicação é maior se comparado ao Mongoose pois o MAMP inclui o PHP, um servidor MySQL e o servidor web em si (que pode ser Apache ou Nginx).

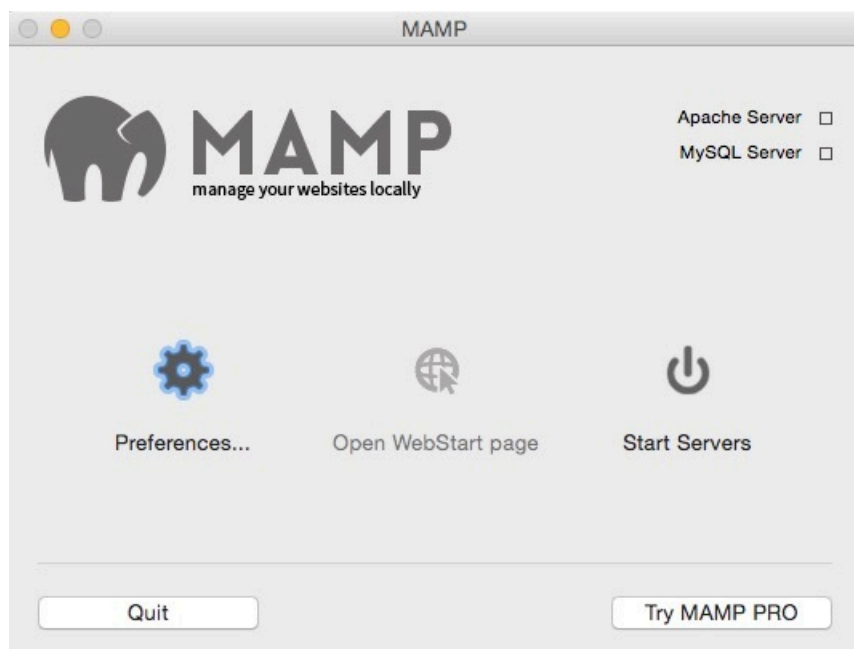


Para fazer o download do MAMP, use o endereço <http://www.mamp.info/en/downloads/><sup>6</sup>

Semelhante ao Mongoose, toda a sua configuração pode ser feita através de uma simples interface gráfica.

---

<sup>6</sup><http://www.mamp.info/en/downloads/>



Configuração do MAMP

## Apache Tomcat

O Apache Tomcat é um dos softwares para servidores web mais conhecidos. Ele é amplamente utilizado por aplicações que usam tecnologia Java, em especial JavaServer Pages.

O processo de instalação e configuração do Tomcat é sensivelmente mais complexo que o do Mongoose e MAMP. Ainda, o Apache Tomcat contém alguns pré-requisitos como o JRE (Java Runtime Environment) instalado e algumas variáveis de sistemas definidas.

O principal motivo pelo qual o Apache Tomcat merece alguma atenção no contexto de UI5 é por conta do Eclipse (uma IDE discutida mais adiante). Esta é uma das IDEs mais indicadas para a construção de aplicações em UI5. No presente momento, a estrutura de diretórios de um projeto UI5 criado no Eclipse é idêntica a projetos voltados para a tecnologia JavaServer Pages. Se você utilizar o Eclipse para criar projetos em UI5 a serem testadas usando o Apache Tomcat, é recomendável baixar plugins para o Eclipse que se integram com o Apache Tomcat e que permitem configurar o ambiente de teste da sua aplicação de forma automatizada.

Michael Herzog traz uma boa explicação de como montar seu ambiente de desenvolvimento usando o Apache Tomcat integrado ao Eclipse em seu post [How to install a basic development environment for SAPUI5<sup>7</sup>](http://scn.sap.com/community/developer-center/front-end/blog/2013/06/01/how-to-install-a-basic-development-environment-for-sapui5) na SCN.

---

<sup>7</sup><http://scn.sap.com/community/developer-center/front-end/blog/2013/06/01/how-to-install-a-basic-development-environment-for-sapui5>

## Node.js

Node.js é um plataforma escrita em JavaScript que nasceu há poucos anos atrás e que já possui um universo ao seu redor. Com seu conceito de pacotes é possível criar aplicações em poucas linhas de código JavaScript.

Há inúmeras formas de se criar um servidor web estático usando Node.js, uma das formas de fazer isso é descrita por John Patterson em seu blog [UI5 SDK on Node.js](#)<sup>8</sup>. Apesar do termo 'SAPUI5' estar na URL, o mesmo processo se aplica ao OpenUI5.

## Outros servidores Web

Para tornar a lista um pouco mais completa, abaixo estão servidores web bastante populares também. Todos eles, são implementações do projecto *Apache HTTP Server* que incluem outras bibliotecas e funcionalidades.

- ApacheHaus
- Apache Lounge
- BitNami WAMP Stack
- WampServer
- XAMPP

## Hospedagens gratuitas

Os servidores que possuem o SAP HANA ou o SAP NetWeaver instalado provavelmente não são desligados com frequência pelas empresas que os têm. Se você quiser criar o seu próprio servidor web hospedando todas as suas aplicações UI5 e que esteja disponível o tempo todo basta instalar algum dos softwares listados anteriormente em um computador que esteja ligado constantemente. Além disso, você vai precisar de um domínio ou IP fixo para que seu servidor seja facilmente alcançado. Obviamente isso tem um custo. Se você planeja ter centenas ou milhares de visitantes simultaneamente, talvez seja vantajoso contratar um serviço de hospedagem a parte. Mas se na verdade o que você precisa simplesmente é ter suas aplicações online para possuir uma espécie de portfólio que será visto por uma pessoa ou outra, vale a pena conhecer algumas possibilidades de hospedar suas aplicações UI5 gratuitamente.



Para saber um pouco mais da importante de ter um portfólio online, assista a primeira metade da minha palestra [Uso do GitHub para hospedar aplicações OpenUI5](#)<sup>9</sup> realizada no SAP Inside Track São Paulo em 2014.

---

<sup>8</sup><http://scn.sap.com/community/developer-center/front-end/blog/2014/01/05/sapui5-sdk-on-nodejs>

<sup>9</sup><https://www.youtube.com/watch?v=z0diqLx8Q0>

## Google Drive

Além de servir como um backup dos seus arquivos na nuvem, o Google Drive permite que você compartilhe arquivos com outras pessoas. Usando o conceito de compartilhamento do Google Drive, é possível hospedar seus arquivos gratuitamente. A única desvantagem deste tipo de hospedagem é que a URL gerada pelo Google é bem longa e não intuitiva. Isso porém não é um problema grande uma vez que você pode ter um site/blog contendo links para suas aplicações ou até mesmo usar algum serviço para criação de URLs curtas.

Maurício Lauffer explica na SCN em seu post [Hospedando SAPUI5 app no Google Drive](#)<sup>10</sup> como realizar tal procedimento (que também pode ser feito usando OpenUI5).

## GitHub Pages

O [GitHub](#)<sup>11</sup> é “facebook” dos desenvolvedores. Se você trabalha com desenvolvimento mas nunca ouviu falar nele, é possível que seu nome seja Lucas Silva e Silva <sup>12</sup>.

Algo que nem todos os usuários do GitHub conhecem é o [GitHub Pages](#)<sup>13</sup>. O GitHub pages é um serviço gratuito que permite usar o GitHub como hospedagem de aplicações web. Em comparação com o Google Drive, o GitHub tem uma grande vantagem de usar URLs com bons nomes (como por exemplo <http://fabiopagoti.github.io/NYT-Launchpad-UI5/><sup>14</sup>). Se você é familiar ao GitHub, saiba que o nome da URL nada mais é do que *seuUsuarioNoGitHub.github.io/repositorio*. Logo você pode ter um conjunto de páginas para cada um dos seus repositórios no GitHub.



Para saber como usar tal serviço, assista a segunda metade da minha palestra [Uso do GitHub para hospedar aplicações OpenUI5](#)<sup>15</sup> realizada no SAP Inside Track São Paulo em 2014.

## IDEs

De posse do pacote de *runtime* ou (preferencialmente) do SDK do UI5 e de um servidor web funcional, seja ele qual for, você está bem próximo de começar a colocar a mão na massa. O próximo passo é escolher uma IDE (*Integrated Development Environment*) de sua preferência.

Se você não é familiar ao termo IDE, basta entender que ele é um software que facilita a construção de outros softwares. Existem IDEs gratuitas ou pagas, famosas ou pouco conhecidas, flexíveis ou engessadas e genéricas ou específicas no que se tange a linguagem de programação utilizada

---

<sup>10</sup><http://scn.sap.com/community/portuguese/blog/2014/08/28/hospedando-sapui5-app-no-google-drive>

<sup>11</sup>[www.github.com](http://www.github.com)

<sup>12</sup>Mundo Da Lua

<sup>13</sup><https://pages.github.com/>

<sup>14</sup><http://fabiopagoti.github.io/NYT-Launchpad-UI5/app/WebContent>

<sup>15</sup><https://www.youtube.com/watch?v=z0diqLx8Q0>

nas mesmas. Exemplos de IDEs bem conhecidas são o Eclipse, Netbeans, Visual Studio, Xcode e SublimeText.



Quem vem do mundo ABAP está acostumado a utilizar a transação SE80 do SAP para criar aplicações. Logo a SE80 é a IDE mais conhecida de quem trabalha com ABAP (apesar de ser possível utilizar o Eclipse também já há alguns anos).

## Bloco de Notas e outros processadores de texto

O bloco de notas é simplesmente um processador de texto e não uma IDE. Uma IDE conta com recursos que facilitam a vida do desenvolvedor e qualquer processador de texto embutido num sistema operacional está longe de prover tais recursos.

Mas vale mencionar que é possível criar aplicações UI5 usando o bloco de notas por exemplo pois tudo que você irá criar são arquivos .html, .css e .js. Porém, você não será tão produtivo se comparado a alguém usando uma IDE. Ainda, você terá que redobrar a atenção para não cometer erros de digitação pois o bloco de notas não irá lhe avisar que você os cometeu.

Se o que você precisa é simplesmente fazer uma pequena alteração em algum arquivo já conhecido, não há problema algum em fazer esta alteração usando um processador de texto simples.

## Eclipse

O Eclipse é uma IDE usada por desenvolvedores Java, PHP, Javascript, Go, etc. Uma das vantagens dela é ser extremamente flexível através de seus plugins. A SAP começou a adotar o próximo Eclipse como IDE em suas plataformas no meio dos anos 2000, ganhando ainda mais destaque com o advento do SAP HANA.



Para fazer o download do Eclipse visite <https://www.eclipse.org/downloads/><sup>16</sup>. Atualmente as versões Luna (4.4) e Kepler (4.3) são suportadas.

Para criar aplicações em UI5 do Eclipse, recomenda-se instalar o *SAPUI5 Tools*. Este é um plugin que adapta o Eclipse para o desenvolvimento de aplicações em UI5, criando wizards de projetos, *code completion*, e outras funcionalidades.



Para fazer o download do *SAPUI5 tools*, visite a página <https://tools.hana.ondemand.com/><sup>17</sup>

---

<sup>16</sup><https://www.eclipse.org/downloads/>

<sup>17</sup><https://tools.hana.ondemand.com/#sapui5>

## SAP Web IDE

A fundação Eclipse possui um projeto chamado [Orion](http://eclipse.org/orion/)<sup>18</sup> que é a base para a IDE chamada SAP Web IDE. O Orion tem por objetivo levar a experiência em desenvolvimento de software para a nuvem o que significa ter uma IDE rodando no navegador.

O *SAP HANA Cloud Platform (HCP)* permite a utilização da SAP Web IDE para a construção de aplicações baseadas em UI5. Além disso, a SAP Web IDE também é usada para a criação, extensão e adaptação de aplicações Fiori.

No final de 2014 foi lançada uma versão da SAP Web IDE que pode ser instalada *On-Premise* (localmente), principalmente para fins de teste. Wouter Lemaire demonstra como realizar tal processo em seu post [Start with the SAP Web IDE On-Premise](#)<sup>19</sup> na SCN.

Além do Eclipse a SAP Web IDE tende a ser as duas IDEs foco da SAP. Infelizmente a versão On-Premise não conta com vários dos recursos disponíveis no HANA Cloud como *code completion*, integração com SAP HANA Cloud, desenvolvimento de templates e plugins e outros.

## SublimeText

O [SublimeText](http://www.sublimetext.com/)<sup>20</sup> é uma IDE bem leve mas ao mesmo tempo muito poderosa. Semelhante ao Node.js, esta aplicação trabalha com o conceito de pacotes que permitem flexibilizar muito seu funcionamento. O SublimeText é tão poderoso que o autor deste livro optou por escrevê-lo nele ao invés de utilizar processadores de texto como o Microsoft Word ou Google Docs.

Apesar de ser uma IDE extremamente popular, a SAP não contempla o SublimeText como parte da sua estratégia de desenvolvimento. Contudo isso não impediu que a comunidade de desenvolvedores SAP criassem pacotes e outros projetos para SublimeText que facilitam o desenvolvimento de aplicações UI5. Como exemplo,

- [SublimeUI5](#)<sup>21</sup>
- [Generator-OpenUI5](#)<sup>22</sup>
- [UI5 SplitApp Boilerplate](#)<sup>23</sup>

## JSBin e JSFiddle

Em breve!

---

<sup>18</sup><http://eclipse.org/orion/>

<sup>19</sup><http://scn.sap.com/community/developer-center/front-end/blog/2014/12/24/start-with-the-sap-web-ide-on-premise>

<sup>20</sup><http://www.sublimetext.com/>

<sup>21</sup><https://github.com/qmacro/SublimeUI5>

<sup>22</sup><https://github.com/saschakiefer/generator-openui5>

<sup>23</sup><https://github.com/6of5/UI5SplitApp-Boilerplate>

## Navegadores

Suas aplicações UI5, por serem web, são executadas em navegadores. Diferentemente do desenvolvimento em ABAP, o desenvolvimento web preocupa-se com compatibilidade de clients. Logo, você sempre deveria testar sua aplicação nos diferentes navegadores existentes: Chrome, Firefox, Safari, Opera e o mais sem graça de todos, Internet Explorer.

E é usando o navegador que você irá testar, explorar e depurar sua aplicação UI5. Se você nunca utilizou um navegador para depurar uma aplicação web, ficará maravilhado em como o Google Chrome e o Mozilla Firefox foram criados pensando bastante em quem desenvolve software.

### Google Chrome

O Google Chrome conta nativamente com um recurso chamado “Ferramentas do Desenvolvedor”, através é possível saber tudo que se passa antes, durante e depois que uma página web é carregada em sua tela.

### Mozilla Firefox

O Mozilla Firefox hoje conta com recursos semelhantes ao Google Chrome mas nem sempre foi assim. O uso add-on [Firebug](http://getfirebug.com/)<sup>24</sup> que traz tais funcionalidades há anos ainda é muito comum entre vários desenvolvedores amantes do Firefox.

## Outras ferramentas importantes

### POSTMAN

Para interagir com o *back end*, uma aplicação web utiliza chamadas HTTP a servidores remotos. Estas chamadas devem ser feitas da maneira correta e na qual o servidor remoto exige.

O [POSTMAN](https://chrome.google.com/webstore/detail/postman-rest-client/fdmmgilgnpjigdojojpjoooidkmcomcm)<sup>25</sup> é uma aplicação embarcada do Google Chrome que permite criar requisições HTTP a partir de uma interface gráfica bem intuitiva. Ela permite também capturar a resposta do servidor remoto e exibir em diferentes formatos.

Esta ferramenta ajuda muito desenvolvedores *back end* a testarem serviços criados nos servidores bem como desenvolvedores *front end* a entenderem como as requisições devem ser montadas em suas aplicações.

---

<sup>24</sup><http://getfirebug.com/>

<sup>25</sup><https://chrome.google.com/webstore/detail/postman-rest-client/fdmmgilgnpjigdojojpjoooidkmcomcm>

## XML Viewer

Muitas aplicações web se baseiam no formato XML para construção de suas telas, adaptação, internacionalização, configuração e consumo de dados. Exibir arquivos num formato que facilite a sua leitura é uma funcionalidade presente em grande parte das IDEs como o Eclipse. Contudo, por vezes a maneira mais simples de exibir tal arquivo XML é diretamente no navegador, que por padrão não costumam nos ajudar indentando o XML.

Isso porém pode ser resolvido através de extensões famosas do Google Chrome e do Mozilla Firefox por exemplo. Particularmente, uso a extensão [XV — XML Viewer](#)<sup>26</sup> do Google Chrome.

## JSON Formatter

O formato .json também é formato muito comum em aplicações web e que representa um objeto em JavaScript. Da mesma forma que arquivos XML, arquivos .json podem ser formatados utilizando extensões dos navegadores. Particularmente utilizo a extensão [JSON Formatter](#)<sup>27</sup> do Google Chrome.

---

<sup>26</sup><https://chrome.google.com/webstore/detail/xv-%E2%80%94-xml-viewer/eeocglpgjdpafaedpblffpeebgmgddk>

<sup>27</sup><https://chrome.google.com/webstore/detail/json-formatter/bcjindccaagfpajjmafapmmgkkggoa>

# Criando um Hello World em UI5

Uma vez que você tenha preparado seu ambiente de desenvolvimento, podemos começar a criar aplicações em UI5. Ansioso? Eu também mal podia esperar por este momento!



Leia o Apêndice 1 para entender a estrutura de diretórios utilizada no livro

Como criaremos nossa primeira aplicação, vamos fazê-la o mais simples possível. Crie uma pasta chamada 'hello\_world' no diretório usado por seu servidor web local. Esta pasta armazenará o único arquivo do nosso projeto. Em seguida, abra um editor de texto ou sua IDE favorita e crie um arquivo chamado index.html. Nele, copie o conteúdo abaixo (o código será explicado mais adiante).

## Hello World

---

```
1  <!DOCTYPE html>
2  <html>
3
4  <head>
5      <meta http-equiv='X-UA-Compatible' content='IE=edge' />
6      <title>Hello World</title>
7
8      <!-- Carregando o UI5 instalado localmente -->
9      <script id='sap-ui-bootstrap'
10         src='../..../openui5-sdk/resources/sap-ui-core.js'
11         data-sap-ui-theme='sap_bluecrystal'
12         data-sap-ui-libs='sap.ui.commons'>
13      </script>
14
15      <!-- Carregando o UI5 a partir de CDN
16      <script id='sap-ui-bootstrap'
17         src='https://openui5.hana.ondemand.com/resources/sap-ui-core.js'
18         data-sap-ui-theme='sap_bluecrystal'
19         data-sap-ui-libs='sap.ui.commons'>
20      </script>
21      -->
22
23  <script>
24
```

```
25     var hello = new sap.ui.commons.TextView({
26         text: 'Hello World',
27     });
28
29     hello.placeAt('content');
30
31 </script>
32
33 </head>
34
35 <body class='sapUiBody'>
36     <div id='content'></div>
37 </body>
38
39 </html>
```

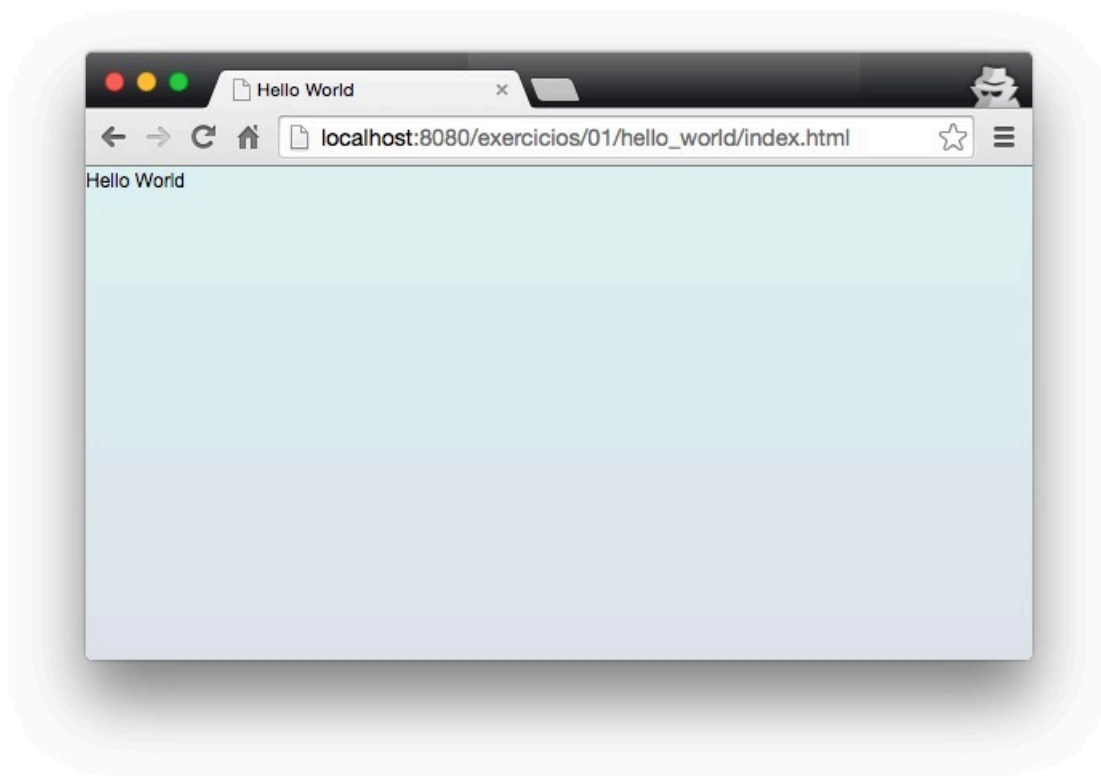
---

Com o servidor web ligado, use o navegador para acessar o arquivo index.html.



Lembre-se: o diretório no qual seus arquivos estarão deve ser alcançável a partir do diretório configurado no seu servidor web.

O resultado esperado é exibido na imagem abaixo.



Hello World em UI5

Note que na URL estamos usando o endereço *localhost*, o que indica que estamos acessando o arquivo com ajuda do nosso servidor web local.

## Estrutura básica de uma aplicação UI5

Vamos estudar um pouco o que se passa no nosso arquivo `index.html`



### Por que `index.html`?

O nome do seu arquivo html poderia ser realmente qualquer um. Porém recomenda-se o nome *index* pois isso indica o arquivo principal do diretório. Servidores web (como o Mongoose) sempre procuram por um arquivo chamado *index* quando uma requisição é feita usando o caminho incompleto para um arquivo (quando somente o diretório é especificado).

## HTML e `<div>` principal

Primeiramente vamos entender como HTML está estruturado.

```
<!DOCTYPE html>
<html>

<head>
  <meta http-equiv='X-UA-Compatible' content='IE=edge' />
  <title>Hello World</title>

<!-- ... -->

</head>

<!-- ... -->

<body class='sapUiBody'>
  <div id='content'></div>
</body>

</html>
```



Note que comentários no padrão HTML foram inseridos no lugar dos comandos não pertinentes no momento.

- A linha `<!DOCTYPE html>` define que o documento HTML está em sua versão 5.
- O bloco entre `<html>` e `</html>` define o documento HTML em si.
- O bloco entre `<head>` e `</head>` define alguns metadados sobre o documento e seu título. Apesar de não ser exibido no trecho acima, é nele foi inserida uma referência a outro arquivo (*bootstrap*, discutido mais adiante).
- O bloco entre `<body>` e `</body>` define o conteúdo do documento HTML. O que está entre estas tags será renderizado pelo navegador.

O bloco *body* é o mais importante neste momento. Independente se sua aplicação UI5 é simples como um Hello World ou complexa como um aplicativo para acompanhamento de ordens de venda, o conteúdo pode ser sempre este.

Tecnicamente falando, não há impedimento algum de adicionar *tags* entre o `<div id='content'></div>` ou posterior a ele, porém via de regra não há esta necessidade. O `div` será populado dinamicamente usando as classes disponíveis no UI5 via JavaScript.



Se você é familiar ao Web Dynpro, você pode entender o `<div id='content'></div>` como o *container* “`rootUIElementContainer`” de uma visão.

Logo, você pode até criar um template de código contendo esta estrutura de HTML a ser usado em todos seus projetos. Mas é uma boa ideia incluir também neste template o *bootstrap* do UI5.

## Bootstrap

Em um arquivo HTML, é possível fazer referência a um arquivo JavaScript usando a tag `<script>`, como o exemplo abaixo:

```
<head>  
  <script src="lib/jquery.js"></script>  
</head>
```

Esta referência se faz necessária caso você queira usar os objetos existentes no arquivo .js dentro da sua página HTML. É possível ainda especificar arquivos usando uma URL absoluta pertencente a um outro domínio.

```
<head>  
  <script src="https://code.jquery.com/jquery-2.1.3.js"></script>  
</head>
```

Quando uma referência como esta é processada pelo navegador, o mesmo realiza o download do recurso (no caso, um arquivo JavaScript) a partir do servidor web que o possui. Logo, o servidor web remoto apenas hospeda o código JavaScript mas não o executa. Esta tarefa é realizada pelo próprio navegador. Em outras palavras, o processamento das páginas e seus scripts é feito no lado do cliente.



Esta é uma grande diferença em relação ao processamento de telas no ABAP, que realizado pelo servidor NetWeaver.

O UI5 é composto por não um mais vários arquivos JavaScript (todos em versão convencional e minificada - conceito explicado mais adiante). Felizmente não há necessidade de importar um a um na sua página HTML. O **Bootstrap** nada mais é que do que a forma simples de inicializar o UI5.

## index.html

---

```
<!-- Carregando o UI5 instalado localmente -->
<script id='sap-ui-bootstrap'
  src='../..../openui5-sdk/resources/sap-ui-core.js'
  data-sap-ui-theme='sap_bluecrystal'
  data-sap-ui-libs='sap.ui.commons'>
</script>
```

---

A tag `<script>` utilizada no *bootstrap* é um pouco mais elaborada. Ela possui algumas propriedades adicionais exibidas na tabela abaixo:

Nome da propriedade	Função
src	Especificar o caminho do 'core' do UI5, que está presente no caminho <i>resources/sap-ui-core.js</i> dentro do pacote de <i>runtime</i> ou SDK
data-sap-ui-theme	Tema usado na aplicação. O valor inserido nesta propriedade afeta o .css que é usado na sua aplicação
data-sap-ui-libs	Define quais os controles (elementos de tela) que serão usados na sua aplicação.



Adapte a propriedade *src* de acordo com a localização do UI5 no seu computador. No caso acima, o arquivo *sap-ui-core.js* está localizado três diretórios acima do diretórios no qual o arquivo *index.html* está localizado.

Através do *bootstrap* é possível carregar apenas partes do UI5 que serão usados em seu código. Isso evita sobrecarga na aplicação uma vez que o UI5 é compreende mais de 200 controles diferentes.

Note ainda que o *bootstrap* pode carregar o UI5 diretamente de uma CDN (*Content Delivery Network*), usando o endereço abaixo:

SAPUI5	<a href="https://sapui5.hana.ondemand.com/resources/sap-ui-core.js">https://sapui5.hana.ondemand.com/resources/sap-ui-core.js</a>
OpenUI5	<a href="https://openui5.hana.ondemand.com/resources/sap-ui-core.js">https://openui5.hana.ondemand.com/resources/sap-ui-core.js</a>

Por conveniência, deixei uma versão de *bootstrap* usando a CDN comentada no nosso arquivo *index.html*.

### index.html

---

```
<!-- Carregando o UI5 a partir de CDN
  <script id='sap-ui-bootstrap'
    src='https://openui5.hana.ondemand.com/resources/sap-ui-core.js'
    data-sap-ui-theme='sap_bluecrystal'
    data-sap-ui-libs='sap.ui.commons'>
  </script>
-->
```

---

Para mais informações sobre o bootstrap, leia a página [Initializing and Loading SAPUI5<sup>28</sup>](#) (também disponível na documentação da SDK).

## Minificação

Para otimizar o tempo de download de arquivos JavaScript, é comum servidores web armazenarem versões *minificadas* dos mesmos. Versão minificadas possuem o mesmo código fonte funcional, porém sem quebras de linhas e espaços (que fazem com que o tamanho dos arquivos seja maior). A biblioteca utilizada anteriormente também é disponibilizada em forma minificada: (repare no **min** no nome do arquivo).

```
<head>
  <script src="https://code.jquery.com/jquery-2.1.3.min.js"></script>
</head>
```

## Código UI5 e Renderização

No caso da nossa aplicação *Hello World*, todo o código fonte UI5 estava presente na própria página principal da aplicação. Essa organização não é indicada para aplicações mais complexas pois desrespeita o conceito MVC, o qual é característico do UI5. Contudo, vale a pena expor o que acontece em detalhes quando se usa o UI5:

---

<sup>28</sup><https://openui5.hana.ondemand.com/#docs/guide/a04b0d10fb494d1cb722b9e341b584ba.html>

index.html

---

```
<script>

    var hello = new sap.ui.commons.TextView({
        text: 'Hello World',
    });

    hello.placeAt('content');

</script>
```

---

O código acima cria um objeto do tipo `TextView`, presente no pacote `sap.ui.commons`, que foi registrado para uso no *bootstrap*. Este objeto é armazenado na variável de referência “hello”, que por sua vez é usada para chamada do método “placeAt”. O método `placeAt` renderiza o controle em questão (no caso o `TextView`) em uma área do documento HTML. Note que ‘content’ é o *id* do único `<div>` inserido no nosso arquivo.

Usando as ferramentas de desenvolvedor do Google Chrome ou Firefox, é possível ver que uma vez que a página tenha sido completamente carregada, o corpo do documento HTML (bloco entre as tags `<body>` `</body>`) foi modificado, resultando no trecho de código abaixo:

TextView renderizado

---

```
<div id="content" data-sap-ui-area="content">
    <span
        id="__view0"
        data-sap-ui="__view0"
        title="Hello World"
        tabindex="-1"
        role="document"
        aria-invalid="false"
        aria-disabled="false"
        class="sapUiTv sapUiTvAlignLeft"
        style="direction:inherit">
        Hello World
    </span>
</div>
```

---

Em suma, seu objeto `TextView` é representado em formato HTML conforme a notação acima. Felizmente, usar UI5 acaba sendo bem mais intuitivo do que ter responsabilidade total sobre o conteúdo das páginas.

Não se preocupe em entender cada uma destas propriedades da tag `<span>`. Entenda apenas que você está criando páginas no padrão HTML5 de uma maneira mais alto nível e de maneira reutilizável.

Parabéns! Você acaba de criar um Hello World usando UI5.

# Usando o Eclipse e o Paradigma MVC

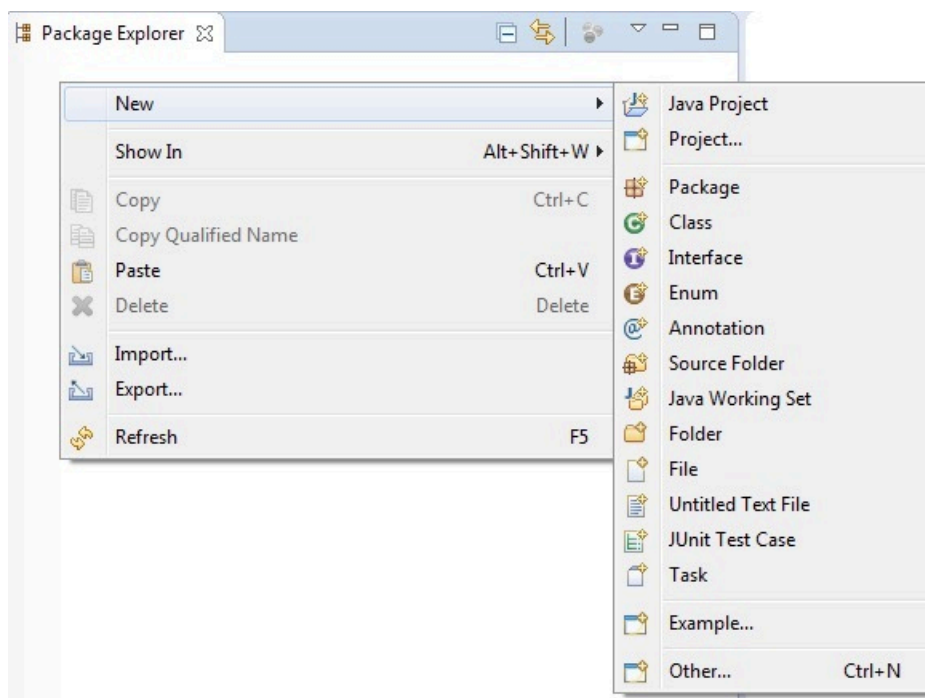
Um Hello World é simplesmente o ponto de partida. Agora iremos criar aplicações um pouco mais elaboradas. Por isso, a partir de agora usaremos o Eclipse com o *SAPUI5 Tools* devidamente instalado para as demonstrações deste livro. Ele nos poupará algum trabalho ao criarmos uma aplicação nova e também quando criamos novas visões.



Você pode usar outra IDE da sua preferência

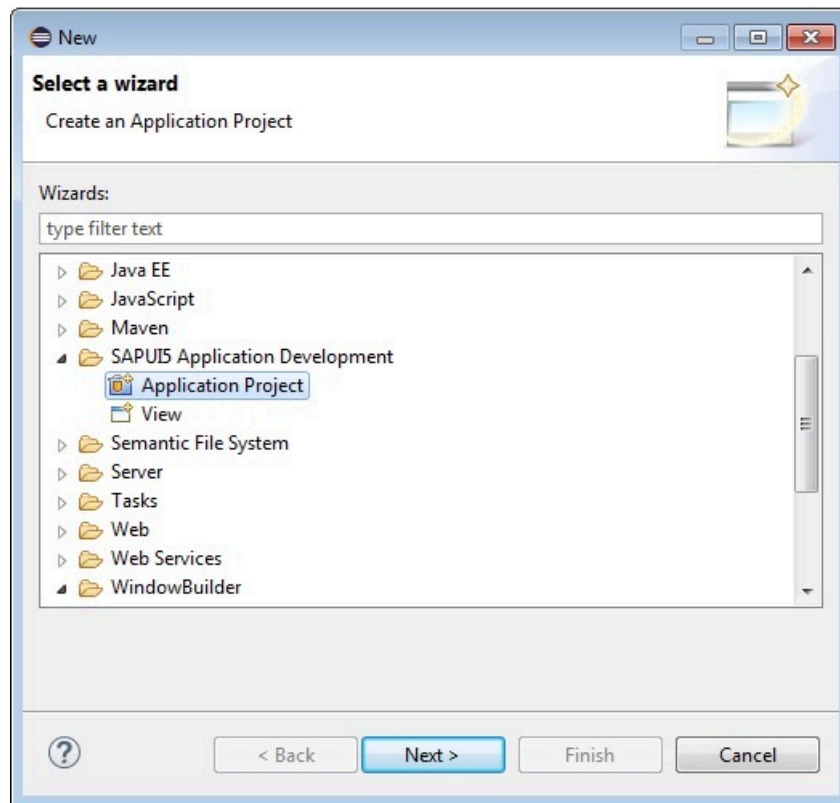
## Criando um projeto UI5 através do Eclipse

Abra o Eclipse e dentro na guia “Project Explorer”, clique com o botão direito e escolha a opção *New > Other*.



Criando um novo projeto

Um popup irá se abrir com o tipo de projeto que deve ser criado. Escolha a opção *SAPUI5 Application Development > Application Project*.



Escolhendo o tipo do projeto



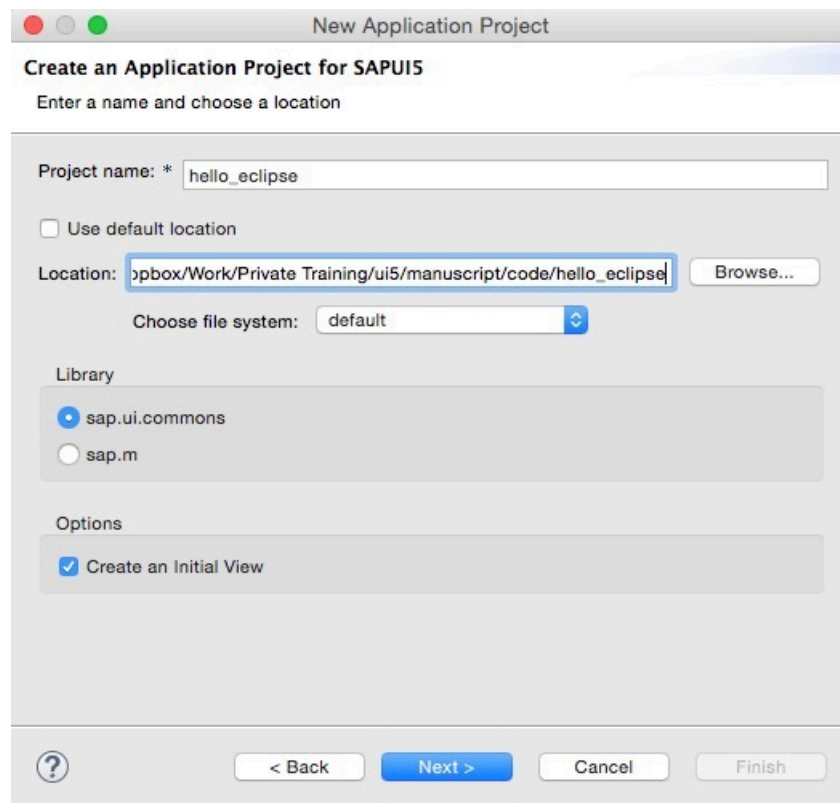
A opção exibida acima só estará disponível caso você tenha instalado o *SAPUI5 Tools* no Eclipse. Por padrão ela não é disponibilizada.

Dê um nome para seu projeto. No caso, foi usado o nome 'hello\_eclipse'. Escolha também o local onde o projeto será gravado. A opção *Use default location* salva seu projeto no *workspace* sendo utilizado pelo Eclipse.



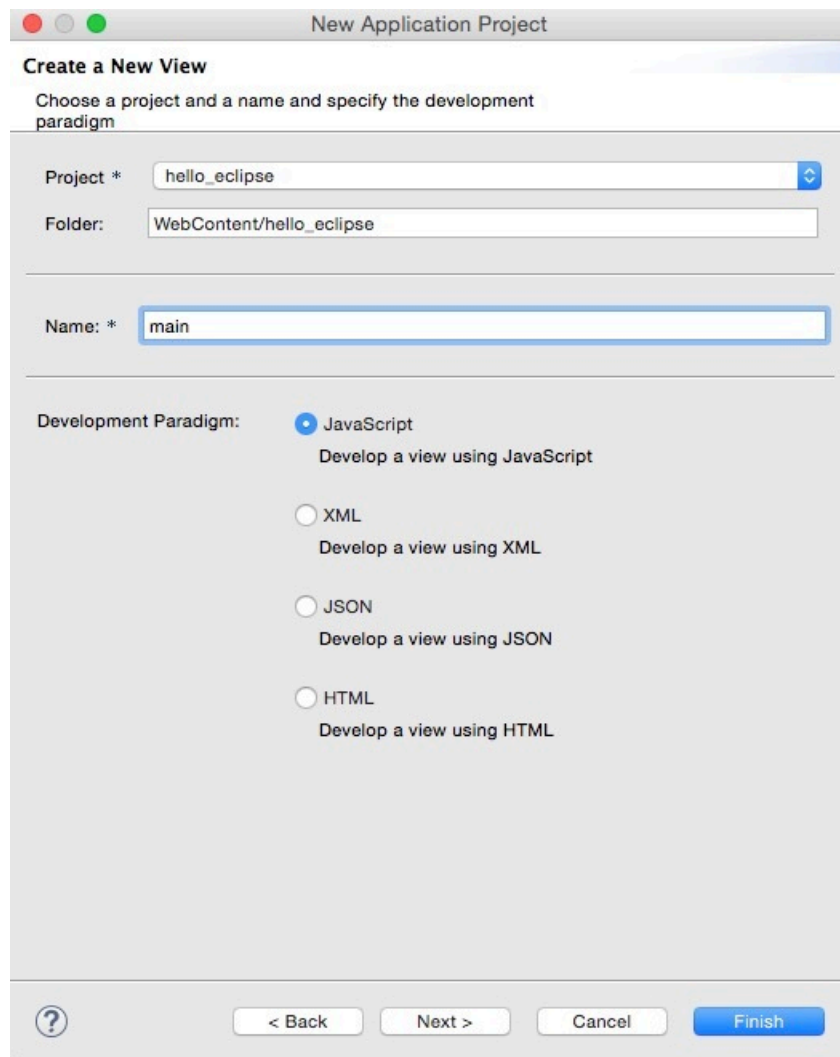
*Workspaces* são formas de organizar vários projetos relacionados. Como você irá criar vários projetos UI5, recomenda-se ter um *workspace* para este fim.

O UI5 possui duas bibliotecas principais de controles. Falaremos mais destas bibliotecas mais adiante. Você pode entender a biblioteca *sap.ui.commons* como um conjunto de controles desenhados para aplicações que serão executadas em um Desktop e a biblioteca *sap.m* sendo voltada para dispositivos móveis. Marque a primeira opção (*sap.ui.commons*). Também deixe marcado a opção "Create an initial view" e clique em *Next*.



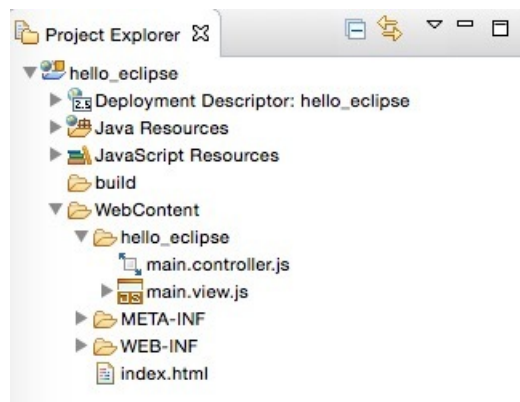
#### Informações gerais sobre o novo projeto

Como você optou por criar uma visão inicial, é hora de dar um nome a mesma. Chamaremos nossa visão de *main*. Como veremos no futuro, uma visão pode ser criada de diferentes formas (JavaScript, XML, JSON e HTML) no UI5. Usaremos a opção padrão, que é JavaScript. Clique então em *finish*. Caso você clique em *next* será exibido simplesmente um resumo do que foi escolhido durante o *wizard*.



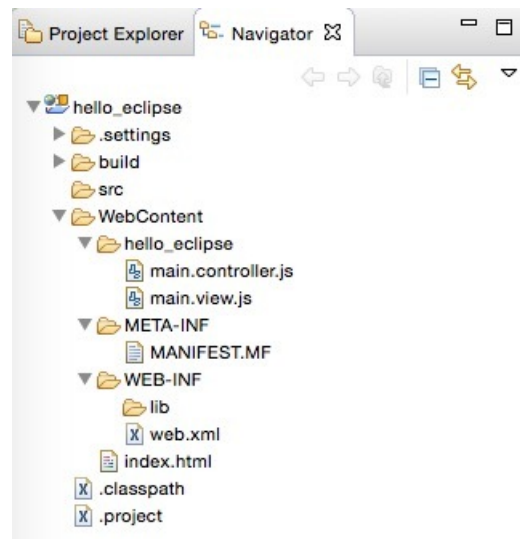
Definindo um nome para a visão inicial

O Eclipse cria uma estrutura de diretórios para aplicações UI5 semelhante a usada em aplicações Java. Por isso uma vez que você crie um projeto no Eclipse pela primeira vez é sugerido a mudança de perspectiva para Java EE. Não é obrigatório estar nesta perspectiva para desenvolver suas aplicações em UI5.



Project Explorer

Uma outra forma de ver os arquivos de seu novo projeto é incluindo a guia *Navigator*, através do menu *Windows > Show View > Navigator* no Eclipse.



Navigator

Por fim, abra o arquivo *index.html* criado dentro do diretório *WebContent* e ajuste a propriedade *src* do *bootstrap* para o caminho correto do UI5 e em seguida teste a sua aplicação (lembre-se de usar o *localhost*). Como a visão que foi gerada estará em branco, o melhor indicativo que o caminho do UI5 está correto é o fundo azul típico do tema *sap\_bluecrystal*.

## Estrutura de uma aplicação UI5 usando MVC

Assim como o Web Dynpro e muitas outras biblioteca JavaScript, o UI5 usa o conceito *MVC* (*Model-View-Controller*) para organizar uma aplicação. Em outras palavras, aplicações UI5 terão arquivos diferentes para diferentes funções:

- Arquivos descrevendo o desenho das telas que serão renderizadas pelo navegador são chamados de **Views ou Visões**
- Arquivos que fazem a ligação dos controles desenhados nas visões com a regra de negócio (*Model*) são chamados de **Controllers**
- Tipicamente, a regra de negócio (**Model ou Modelo**) de uma aplicação UI5 fica no *back end*, que pode ser representado por serviços XSJS no HANA, BAPIs no NetWeaver ou modelos *oData* gerados pelo SAP Gateway ou SAP HANA, etc. Caso você precise implementar alguma regra de negócio na camada de *front end* isso deve ser feito em arquivos JavaScript totalmente desacoplados das suas visões e *controllers*.



O termo **Controller** do paradigma MVC não será traduzido para *controle* para evitar confusões com o termo **controles de interface gráfica**, que representa o que pode ser desenhado dentro das visões como botões, campos de texto, links, etc.

Como estamos usando o Eclipse, temos a criação das visões e *controllers* simplificados. Vamos analisar o que se passa no arquivo `index.html`. Você irá notar que a única diferença entre nosso Hello World criado anteriormente e o projeto atual é o segundo bloco `<script>`, localizado diretamente depois do *bootstrap*. Algumas quebras de linhas foram feitas para facilitar a leitura do código.

`index.html`

---

```
sap.ui.localResources("hello_eclipse");
var view = sap.ui.view({
    id:"idmain1",
    viewName:"hello_eclipse.main",
    type:sap.ui.core.mvc.ViewType.JS
});
view.placeAt("content");
```

---

Ao invés de desenhar um *TextView*, um outro controle de interface gráfica está sendo criado: uma visão. Por hora não se prenda a chamada `sap.ui.localResources`. Note que no final do bloco `<script>` também há uma chamada ao método *placeAt* que irá renderizar o conteúdo da visão dentro do `<div>` principal da página.

Quando optamos por criar uma visão inicial *main* durante a criação do projeto, o Eclipse se encarregou de criar uma pasta com o mesmo nome do projeto (*hello\_eclipse*) diretamente abaixo da pasta *WebContent*. É nesta pasta que as visões são definidas. Para cada visão, dois arquivos são criados: um para definir o conteúdo da visão (*main.view.js*) e o outro é o *controller* daquela visão (*main.controller.js*).

Por hora não nos preocuparemos com o *controller*. Abra o arquivo *main.view.js*.

**WebContent/hello\_eclipse/main.view.js**

---

```
1  sap.ui.jsview("hello_eclipse.main", {
2
3      /** Specifies the Controller belonging to this View.
4       * In the case that it is not implemented, or that "null" is returned, this View\
5       does not have a Controller.
6       * @memberOf hello_eclipse.main
7       */
8      getControllerName : function() {
9          return "hello_eclipse.main";
10     },
11
12     /** Is initially called once after the Controller has been instantiated. It is \
13     the place where the UI is constructed.
14     * Since the Controller is given to this method, its event handlers can be attac\
15     hed right away.
16     * @memberOf hello_eclipse.main
17     */
18     createContent : function(oController) {
19
20     }
21
22 });
```

---

Por padrão, uma visão é definida com dois métodos:

- `getControllerName` - responsável por definir qual o *controller* da visão em questão (Apesar de não ser recomendável, um *controller* pode ser utilizado por várias visões)
- `createContent` - responsável pela definição dos controles de interface gráfica que serão renderizados como parte daquela visão.

Além dos métodos, um id também definido para a visão. Este nome nada mais é do que o diretório no qual a visão está inserida concatenado ao nome do arquivo.



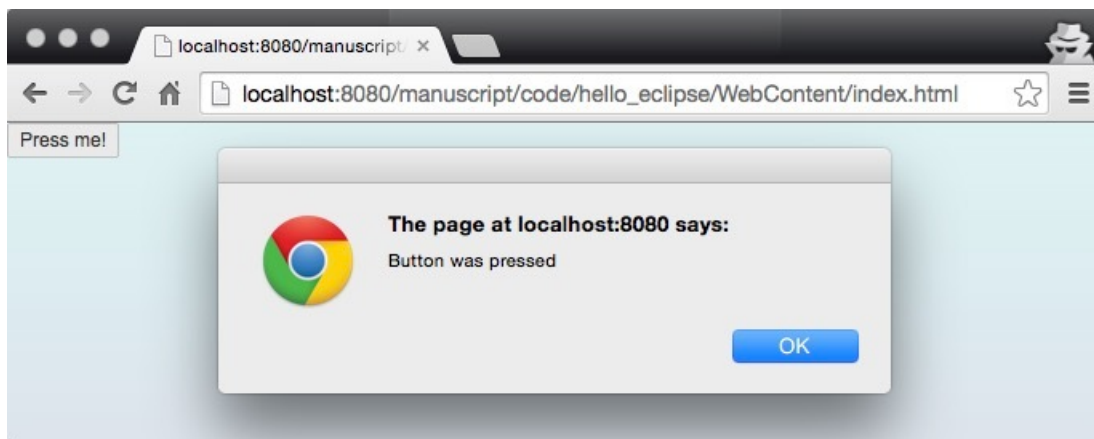
Esta é uma boa hora para olhar a documentação da classe `sap.ui.core.mvc.JSView` dentro da guia *API Reference* na documentação da SDK.

Vamos criar um botão dentro da nossa visão. Dentro do método `createContent`, insira o código abaixo:

WebContent/hello\_eclipse/main.view.js

```
createContent : function(oController) {  
    var but_press_me = new sap.ui.commons.Button({  
        text : "Press me!",  
        press : function() {  
            alert("Button was pressed");  
        }  
    });  
  
    return but_press_me;  
}
```

Teste sua aplicação. Você deve ver um botão que ao clicado exibe um *popup* no navegador.



Botão disparando alerta



Veja a documentação da classe `sap.ui.commons.Button` na SDK.

Neste exemplo, além de definirmos a propriedade *text* para o botão, usamos também seu evento *press*, que espera uma função JavaScript como argumento. Codificamos uma função que exibe o *popup* diretamente no código da função.

É muito comum em JavaScript definirmos funções *inline* assim como fizemos no argumento do evento *press*. Contudo, isso não é recomendável quando o código desta função começa a crescer. Além de dificultar uma eventual manutenção, código JavaScript responsável por fazer validações, escritas em logs e comunicação com o *Model* deve estar nos *Controllers* segundo o paradigma MVC. Ter funções definidas diretamente na visão não é abominável, mas deve ser feito com critério.

Vamos então definir uma função chamada *onPressButton* no nosso *controller* conforme abaixo.

**WebContent/hello\_eclipse/main.controller.js**

---

```
1  sap.ui.controller("hello_eclipse.main", {
2
3  /**
4   * Called when a controller is instantiated and its View controls (if available) \
5   * are already created.
6   * Can be used to modify the View before it is displayed, to bind event handlers \
7   * and do other one-time initialization.
8   * @memberOf hello_eclipse.main
9   */
10 // onInit: function() {
11
12 // },
13
14 /**
15 * Similar to onAfterRendering, but this hook is invoked before the controller's \
16 View is re-rendered
17 * (NOT before the first rendering! onInit() is used for that one!).
18 * @memberOf hello_eclipse.main
19 */
20 //      onBeforeRendering: function() {
21 //
22 //      },
23
24 /**
25 * Called when the View has been rendered (so its HTML is part of the document). \
26 Post-rendering manipulations of the HTML could be done here.
27 * This hook is the same one that SAPUI5 controls get after being rendered.
28 * @memberOf hello_eclipse.main
29 */
30 //      onAfterRendering: function() {
31 //
32 //      },
33
34 /**
35 * Called when the Controller is destroyed. Use this one to free resources and fi\
36 nalize activities.
37 * @memberOf hello_eclipse.main
38 */
39 //      onExit: function() {
40 //
41 //      }
```

```
42
43     onPressButton: function() {
44         alert("Button was pressed");
45     }
46
47 });
```

---

Note que um *controller* é definido com algumas funções comentadas. Estas funções são chamadas automaticamente em momentos específicos do tempo de vida de uma visão, por exemplo quando uma função é criada ou antes de começar o processo de renderização da mesma.

Precisamos agora usar a nossa função definida no *controller* no evento *press* do nosso botão. Volte a visão *main* e modifique a criação do botão conforme abaixo:

WebContent/hello\_eclipse/main.view.js

---

```
createContent : function(oController) {
    var but_press_me = new sap.ui.commons.Button({
        text : "Press me!",
        press : oController.onPressButton
    });

    return but_press_me;
}
```

---

**Talvez este momento seja um dos mais importantes de todo o livro.** Por isso não continue a leitura sem entender o que está acontecendo. Agora, quando o usuário clica no botão o seu evento *press* é chamado, conforme anteriormente. A diferença é que agora ao invés de definirmos a função sendo executada diretamente no código da visão, estamos fazendo uma chamada a função *onPressButton* do *controller* da visão. **Quando a função usada como argumento da propriedade *createContent* é chamada, uma referência ao *controller* da visão é passada como parâmetro.** No exemplo acima, esta referência tem o nome *oController*.

Esta é uma aplicação que usa o paradigma MVC. Lembre-se que a regra de negócio tipicamente fica no servidor e neste caso não fizemos nenhuma chamada dentro do nosso *controller*. Chegará o momento no qual faremos chamadas HTTP a servidores mas antes vamos estudar um pouco mais em detalhes os diferentes tipos de controles de interface gráfica disponíveis.

# Namespaces

JavaScript é uma linguagem orientada a objetos e fracamente tipada. Em outras palavras, variáveis podem ter seus tipos alterados em tempo de execução. Ainda, não é necessário definir um tipo inicial na declaração de uma variável. Em linguagens orientadas a objetos e tipadas como Java, C# e ABAP, classes podem ser entendidas como tipos complexos, os quais podem ser usados para a criação de referências. As referências por sua vez são usadas para a criação de objetos.

JavaScript por ser uma linguagem fracamente tipada não tem o conceito de classes (mas ainda sim é orientada a objetos). Praticamente tudo que você vê num código JavaScript é um objeto (ou parte de um).

Uma boa prática no mundo JavaScript é definir um único *Namespace* para todo o código da sua aplicação. Em outras palavras, recomenda-se criar um “objeto principal” que mantém consigo todos os outros objetos e funções responsáveis pelo código da aplicação. Isso é útil pois simplifica a manutenção e também evita conflitos entre diferentes bibliotecas sendo usadas no mesmo projeto.

**jQuery**, provavelmente a bibliotecas mais conhecida e utilizada no mundo JavaScript usa um *namespace* chamado \$ (cifrão). Toda a funcionalidade do jQuery pode ser acessada através deste objeto. Por exemplo, para saber a versão do jQuery usando sua API usa-se o código abaixo:



Além do \$, pode-se também usar o *namespace* **jQuery**, que simplesmente é uma outra referência ao mesmo objeto.

```
1 $.fn.jquery
2 // http://api.jquery.com/jquery-2/
```

Se forma semelhante, para se realizar uma requisição HTTP, a chamada abaixo pode ser usada:

```
1 $.get( "config.json" );
```

Usando o mesmo princípio, toda a biblioteca do UI5 está sob o *namespace* **sap**. Para provar que **sap** é um objeto, abra a última aplicação UI5 criada usando as ferramenta de desenvolvedor do seu navegador e digite no console:

```
1 typeof sap
```

O resultado exibido será *object*.



digitando *typeof jQuery* no console será exibido o retorno *function* (ou função). Não se preocupe, funções em JavaScript são tipos especiais de objetos que podem ser invocados.

O UI5 é uma biblioteca nada modesta. Para segregar ainda mais as diferentes funcionalidades da biblioteca, desenvolvedores da SAP optaram por criar *namespaces* dentro do *namespace sap*, cada um com sua responsabilidade.

Diretamente abaixo do *namespace sap*, temos dois objetos, conforme a tabela abaixo.

Nome	Função
sap.ui	Namespace com diferentes funcionalidade de JavaScript feitas pela SAP
sap.m	Conjunto de controles de interface gráfica próprios para <b>dispositivos móveis</b>

Diretamente abaixo do *namespace sap.ui*, temos outros *namespaces*, conforme tabela abaixo:

Nome	Função
sap.ui.app	Contém classes depreciadas (Application e MockServer)
sap.ui.base	Classes base de todo UI5, nada existiria se não fosse pelas classes deste <i>namespace</i>
sap.ui.commons	Conjunto de controles de interface gráfica próprios para <b>desktops</b>
sap.ui.core	Funcionalidades usadas por todo o UI5. Contém classes bases de controles de interface gráfica, componentes e MVC
sap.ui.Device	Classes para extração de dados sobre dispositivos e funcionalidades suportadas pelos mesmos
sap.ui.layout	Classes que permitem organizar o layout de controles de interface gráfica
sap.ui.model	API de <i>Data Binding</i> - A comunicação entre <i>controllers</i> e o <i>model</i> geralmente será feita por classes deste <i>namespace</i>
sap.ui.suite	Alguns controles de interface gráfica em estado experimental
sap.ui.table	Controles para criação de tabelas próprias para exibição em Desktops
sap.ui.test	Classes com finalidade de testes
sap.ui.unified	Controles unificados para Desktop e dispositivos móveis
sap.ui.ux3	Controles complexos de interface gráfica



Há outros *namespaces* com finalidades bem específicas mais afundo na estrutura acima. Todos eles são marcados com “N” na documentação da SDK. Classes são marcadas com “C”.

## Namespace sap.ui.core

The SAPUI5 Core Runtime.

Contains the UI5 jQuery plugins (jQuery.sap.\*), the Core and all its components, base class

### Namespaces & Classes

- N AccessibleRole** Defines the accessible roles for ARIA support.
- N AppCacheBuster** The AppCacheBuster is used to hook into URL relevant functions in jQuery
- N BarColor** Configuration options for the colors of a progress bar
- N BusyIndicator** Provides methods to show or hide a waiting animation covering the whole page
- N Collision** Collision behavior: horizontal/vertical.
- G Component** Base Class for Component.
- G ComponentContainer** Component Container
- G Configuration** Collects and stores the configuration of the current environment.
- G Control** Base Class for Controls.
- G Core** Core Class of the SAP UI Library.
- N CSSColor** A string type that represents CSS color values.
- N CSSSize** A string type that represents CSS size values.
- N CSSSizeShortHand** This type checks the short hand form of a margin or padding definition.
- G CustomData** Contains a single key/value pair of custom data attached to an Element.
- G DeclarativeSupport** Static class for enabling declarative UI support.

Classes e *Namespaces* na documentação da SDK

## Dependências entre *namespaces*

Alguns *namespaces* possuem classes que dependem de outras para existirem (como no caso de herança). Logo, antes que uma classe seja utilizada é importante que todas suas dependências já tenham sido carregadas previamente. Felizmente, no que tange a dependência entre *namespaces* criados pela SAP, isso é garantido.

Vejamos o *bootstrap* utilizado no projeto ‘hello\_eclipse’. Nele, carregamos a biblioteca *sap.ui.commons* para utilizarmos a classe *sap.ui.commons.Button*.

```

1 <script src="../../../../openui5-sdk/resources/sap-ui-core-all.js"
2     id="sap-ui-bootstrap"
3     data-sap-ui-libs="sap.ui.commons"
4     data-sap-ui-theme="sap_bluecrystal">
5 </script>
```

Caso este *namespace* não tivesse sido especificado no *bootstrap*, o botão não seria criado e o erro abaixo ocorreria no console do navegador.



Uncaught TypeError: Cannot read property ‘Button’ of undefined

Observando a documentação da classe *sap.ui.commons.Button*, podemos perceber que ela é uma extensão da classe abstrata *sap.ui.core.Control*, estando portanto em um outro *namespace*. Como consequência disso, quando você importa o *namespace* *sap.ui.commons* no *bootstrap*, é necessário que o *namespace* *sap.ui.core* seja carregado de antemão.

Podemos verificar quais as partes da biblioteca estão carregados usando o método *getLoadedLibraries* da classe *sap.ui.core.Core* usando a chamada abaixo:

```
1 sap.ui.getCore().getLoadedLibraries()
```

Ao executar tal chamada no console do navegador, temos a saída abaixo:

```
1 Object {  
2     sap.ui.core: Object,  
3     sap.ui.layout: Object,  
4     sap.ui.commons: Object,  
5     __proto__: Object  
6 }
```

Percebemos então que apesar de não especificar explicitamente, as dependências do *namespace* *sap.ui.commons* foram carregados na aplicação.



O UI5 utiliza-se de diversas outras bibliotecas JavaScript. Uma delas é o [Require.js](http://requirejs.org)<sup>29</sup> que tem por função simplificar a inicialização de arquivos e módulos JavaScript.

## sap.ui.Device - Capturando informações sobre dispositivo

Antes de estudarmos os dois *namespaces* mais famosos (*sap.m* e *sap.ui.commons*), vamos fazer uma pequena aplicação capaz de exibir informações sobre o sistema e o navegador sendo usados pelo usuário.



É muito comum no JavaScript a verificação de informações sobre o dispositivo uma vez que algumas funcionalidades podem não ser suportadas por navegadores específicos.

Crie um novo projeto no Eclipse com seguintes propriedades:

---

<sup>29</sup><http://requirejs.org>

Nome do projeto	device
Biblioteca principal	sap.ui.commons
Criar visão inicial?	Sim
Nome da visão inicial	device_info
Paradigma de programação	JavaScript

Em seguida, ajuste o *bootstrap* para o caminho correto do UI5.

```

1  <!DOCTYPE HTML>
2  <html>
3      <head>
4          <meta http-equiv="X-UA-Compatible" content="IE=edge">
5          <meta http-equiv='Content-Type' content='text/html; charset=UTF-8' />
6
7
8          <script src="../../../../../../openui5-sdk/resources/sap-ui-core.js"
9              id="sap-ui-bootstrap"
10             data-sap-ui-libs="sap.ui.commons"
11             data-sap-ui-theme="sap_bluecrystal">
12
13             <!-- add sap.ui.table, sap.ui.ux3 and/or other libraries to 'data-sap-ui-libs' \
14 if required -->
15
16             <script>
17                 sap.ui.localResources("device");
18                 var view = sap.ui.view({
19                     id:"iddevice_info1",
20                     viewName:"device.device_info",
21                     type:sap.ui.core.mvc.ViewType.JS
22                 });
23                 view.placeAt("content");
24             </script>
25
26         </head>
27         <body class="sapUiBody" role="application">
28             <div id="content"></div>
29         </body>
30 </html>

```

Para evitarmos de abordar assuntos que não serão explicados em detalhes por hora, nossa aplicação terá apenas um *TextView* (semelhante ao nosso Hello World). Este *TextView* será preenchido com algumas informações do dispositivo assim que a visão “device\_info” for inicializada.

Na sua visão, crie um *TextView* conforme abaixo.

**WebContent/device/device\_info.view.js**

---

```
sap.ui.jsview("device.device_info", {

    /** Specifies the Controller belonging to this View.
     * In the case that it is not implemented, or that "null" is returned, this View\
     does not have a Controller.
     * @memberOf device.device_info
     */
    getControllerName : function() {
        return "device.device_info";
    },

    /** Is initially called once after the Controller has been instantiated. It is \
    the place where the UI is constructed.
     * Since the Controller is given to this method, its event handlers can be attac\
    hed right away.
     * @memberOf device.device_info
     */
    createContent : function(oController) {
        return new sap.ui.commons.TextView("txv_device_info");
    }

});
```

---

Não podemos atribuir o valor do *TextView* estaticamente no código usando a propriedade *text* por as informações do dispositivo serem desconhecidas. Por este motivo, demos um “id” para nosso *TextView* afim de que consigamos capturá-lo posteriormente no *controller*, onde será capturado os detalhes do mesmo.

No seu *controller*, insira o código abaixo:

**WebContent/device/device\_info.controller.js**

---

```
sap.ui.controller("device.device_info", {

    /**
     * Called when a controller is instantiated and its View controls (if available) \
     are already created.
     * Can be used to modify the View before it is displayed, to bind event handlers \
     and do other one-time initialization.
     * @memberOf device.device_info
     */
```

```

onInit: function() {

    var browser_name = this.getBrowserName(sap.ui.Device.browser.name);
    var os_name = this.getOSName(sap.ui.Device.os.name);
    var system_name;

    //          Refactor: Using generic functions
    //          browser_name = this.getGenericName(
    //                                     sap.ui.Device.browser.name,
    //                                     sap.ui.Device.browser.BROWSER
    //                                     );
    //
    //          os_name = this.getGenericName(
    //                                     sap.ui.Device.os.name,
    //                                     sap.ui.Device.os.OS
    //                                     );

    system_name = this.getGenericName(
                                                true,
                                                sap.ui.Device.system
                                                );

    sap.ui.getCore()
        .byId("txv_device_info")
        .setText(
            "Browser: " + browser_name + "\n" +
            "OS: " + os_name + "\n" +
            "System Name:" + system_name
        );
},

/**
 * Similar to onAfterRendering, but this hook is invoked before the controller's \
 * View is re-rendered
 * (NOT before the first rendering! onInit() is used for that one!).
 * @memberOf device.device_info
 */
//          onBeforeRendering: function() {
//
//          },
//
/**

```

```

* Called when the View has been rendered (so its HTML is part of the document). \
Post-rendering manipulations of the HTML could be done here.
* This hook is the same one that SAPUI5 controls get after being rendered.
* @memberOf device.device_info
*/
//      onAfterRendering: function() {
//
//      },

/**
* Called when the Controller is destroyed. Use this one to free resources and fi\
nalize activities.
* @memberOf device.device_info
*/
//      onExit: function() {
//
//      }

/**
* Given a browser abbreviation, get its name
*/
getBrowserName: function(browser_short_name){
    for ( var browserName in sap.ui.Device.browser.BROWSER) {
        if (browser_short_name == sap.ui.Device.browser.BROWSER[browserName]) {
            return browserName.toLowerCase();
        }
    }
},

/**
* Given a OS abbreviation, get its name
*/
getOSName: function(os_short_name){
    for ( var osName in sap.ui.Device.os.OS) {
        if (os_short_name == sap.ui.Device.os.OS[osName]) {
            return osName.toLowerCase();
        }
    }
},

/**
* Given a enum and a value, get the object property whose value is equal that \

```

```

value
    */
    getGenericName: function(stuff_short_name, stuff_enum){
        for ( var stuffName in stuff_enum) {
            if (stuff_short_name == stuff_enum[stuffName]) {
                return stuffName.toLowerCase();
            }
        }
    }
});

```

---

Usamos a função *onInit* para carregarmos o valor do *TextView* uma vez que a visão seja carregada. Primeiro, precisamos capturar no *controller* o objeto que representa nosso *TextView*. Uma das maneiras mais fáceis de se fazer isso é usando o método *byId* do retorno da chamada *sap.ui.getCore()*.

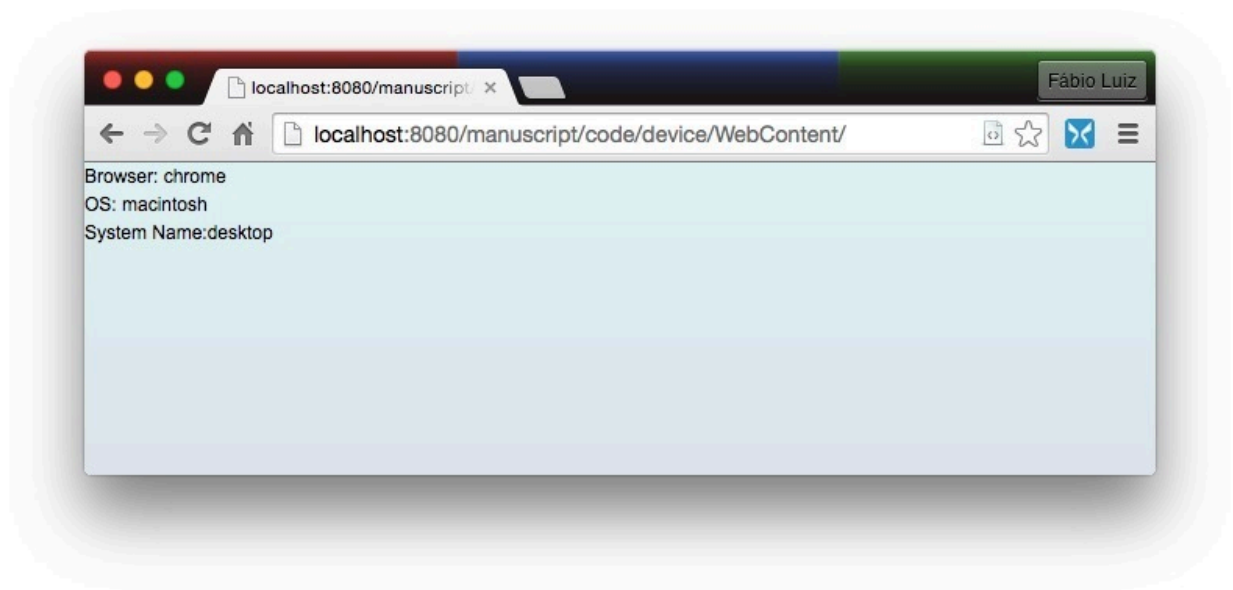
Uma vez que tenhamos a referência do *TextView*, chamamos o método *setText* que é capaz de alterar tal propriedade do controle. O argumento passado para o *setter* é o resultado de uma concatenação com as informações do dispositivo já carregadas. O código que antecede a chamada do método *setText* é um exemplo de uso do *namespace sap.ui.Device*, que é capaz de capturar informações sobre o dispositivo. As três informações que requeemos estão armazenadas nas propriedades abaixo:

- **sap.ui.Device.browser.name** - Nome abreviado do navegador
- **sap.ui.Device.os.name** - Nome abreviado do sistema operacional
- **sap.ui.Device.system** - informação do tipo de sistema sendo usado (Desktop, Celular, Tablet)

Pela maneira que o UI5 armazena informações nos objetos (de forma abreviada), foram definidos três métodos adicionais no *controller* que ajudarão a identificar mais facilmente as informações que desejamos:

- **getBrowserName** - Dado uma abreviação de navegador, retorna o nome do mesmo. Exemplo: Dado “cr”, retorna “CHROME”
- **getOSName** - Dado uma abreviação de sistema operacional, retorna o nome do mesmo. Exemplo: Dado “mac”, retorna “MACINTOSH”
- **getGenericName** - Como o código dos métodos acima é similar, foi criado um método genérico para fins de estudo - Eis um grande exemplo de reutilização de código dentro de um *controller*

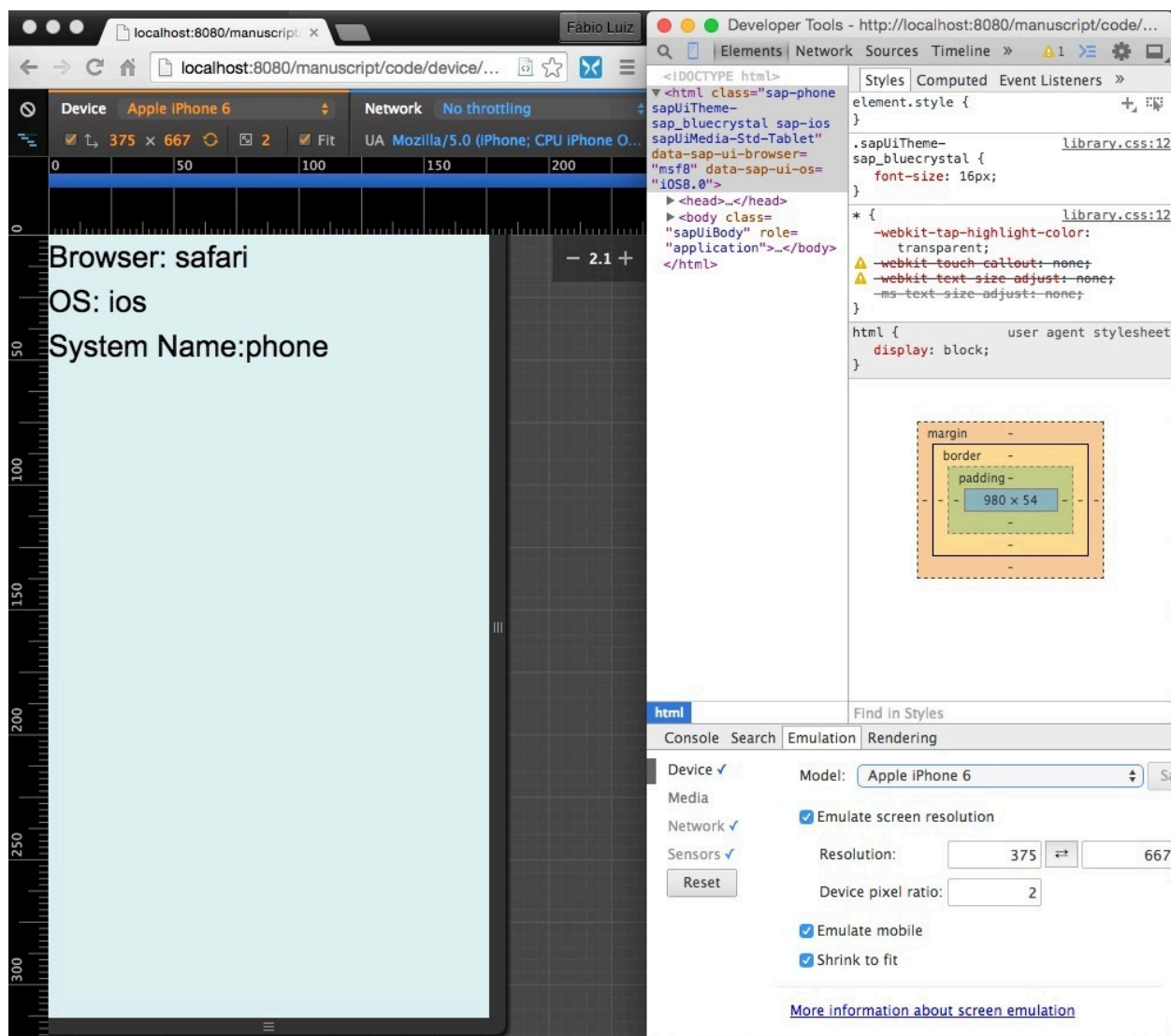
Como resultado, temos a saída abaixo:



### Informações sobre dispositivos



Use a opção de emulação do Google Chrome para simular um *smartphone* e um *tablet* executando a aplicação.



Informações sobre dispositivos

## Bibliotecas de terceiros

Os outros *namespaces* incluídos no UI5 são tão importantes e compreensivos que merecem capítulos especiais no livro. Por hora, vale mencionar também algumas bibliotecas de terceiros (i.e. não criadas pela SAP) que são usadas internamente no UI5.



Todas as bibliotecas de terceiros podem ser encontradas dentro do pacote de *runtime* ou *SDK* no caminho `resources/sap/ui/thirdparty`. (leia arquivo `readme.html` desta pasta para mais detalhes).

Biblioteca	Função	Copyright
<a href="#">Caja-HTML-Sanitizer</a> <sup>30</sup>	Biblioteca para garantir segurança ao embutir HTML, CSS e JavaScript de bibliotecas terceiras em uma aplicação	Google
<a href="#">Crossroads</a> <sup>31</sup>	Sistema de roteamento - Abstrai caminhos de navegação reduzindo assim complexidade do código	Miller Medeiros
<a href="#">D3</a> <sup>32</sup>	Criação de documentos orientado por dados (principalmente usada para gráficos)	Michael Bostock
<a href="#">DataJS</a> <sup>33</sup>	Biblioteca para criação de aplicações web centralizada em dados (vindos de JSON e oData). É a base para o <i>sap.ui.model</i>	Microsoft
<a href="#">Flexie</a> <sup>34</sup>	CSS3 <i>Flexible Box Model</i> cross-browser	Richard Herrera
<a href="#">HandleBars</a> <sup>35</sup>	Construção de <i>templates</i> semânticos	Yehuda Katz
<a href="#">Hasher</a> <sup>36</sup>	Conjunto de funções para controlar o histórico do navegador para aplicações <i>rich media</i>	Miller Medeiros
<a href="#">URI</a> <sup>37</sup>	Biblioteca para trabalhar com URIs	Rodney Rehm
<a href="#">iScroll</a> <sup>38</sup>	JavaScript <i>Scroller</i>	Matteo Spinelli

---

<sup>30</sup><https://code.google.com/p/google-caja/>

<sup>31</sup><http://millermedeiros.github.com/crossroads.js>

<sup>32</sup><http://d3js.org/>

<sup>33</sup><http://datajs.codeplex.com/>

<sup>34</sup><http://flexiejs.com/>

<sup>35</sup><http://handlebarsjs.com/>

<sup>36</sup><https://github.com/millermedeiros/hasher/>

<sup>37</sup><http://medialize.github.io/URI.js/>

<sup>38</sup><http://iscrolljs.com/>

Biblioteca	Função	Copyright
<a href="#">jQuery Mobile</a> <sup>39</sup>	Sistema de interface voltado para aplicações web responsivas executadas em dispositivos móveis	jQuery Foundation
<a href="#">JSZip</a> <sup>40</sup>	Classe para geração e leitura de arquivos .zip	Stuart Knightley
<a href="#">LESS</a> <sup>41</sup>	Pré-processados CSS (adiciona funcionalidades ao CSS)	Alexis Sellier
<a href="#">Mobify/Scooch</a> <sup>42</sup>	Carrosel de imagens ou outro conteúdo	Mobify
<a href="#">Punycode</a> <sup>43</sup>	Conversor Punycode	Mathias Bynens
<a href="#">qUnit</a> <sup>44</sup>	Testes unitários em JavaScript	jQuery Foundation
<a href="#">qUnit Reporter jUnit</a> <sup>45</sup>	Plugin de qUnit para produzir relatórios de teste em formato XML	jQuery Foundation
<a href="#">Require</a> <sup>46</sup>	Carregador de módulos e arquivos. É usado para carregar dependências no UI5	The Dojo Foundation
<a href="#">Signals</a> <sup>47</sup>	Sistema de eventos e mensagens em JavaScript	Miller Medeiros
<a href="#">Sinon</a> <sup>48</sup>	Espiões, <i>stubs</i> e <i>mocks</i> de teste em JavaScript	Christian Johansen

---

<sup>39</sup><http://jquerymobile.com>

<sup>40</sup><http://stuartk.com/jszip>

<sup>41</sup><http://lesscss.org>

<sup>42</sup><https://mobify.github.io/scooch/>

<sup>43</sup><https://github.com/bestiejs/punycode.js>

<sup>44</sup><http://qunitjs.com>

<sup>45</sup><https://github.com/jquery/qunit-reporter-junit>

<sup>46</sup><http://github.com/jrburke/requirejs>

<sup>47</sup><http://millermedeiros.github.com/js-signals/>

<sup>48</sup><https://github.com/cjohansen/Sinon.JS>

Biblioteca	Função	Copyright
<a href="#">SwipeView</a> <sup>49</sup>	Carrosséis infinitos para dispositivo móveis	Matteo Spinelli
<a href="#">vkBeautify</a> <sup>50</sup>	Minificador e formatador ( <i>pretty printer</i> ) de código XML, JSON, CSS e SQL	Vadim Kiryukhin
<a href="#">Scroller</a> <sup>51</sup>	Efeito de <i>panning</i> e <i>zoom</i> para DOM e Canvas	Zinga

Note que nem todas estas bibliotecas são carregadas automaticamente no *bootstrap*. Você pode importar qualquer uma delas usando uma tag `<script>` convencional ou utilizar a chamada abaixo:

#### Importando biblioteca D3

---

```
jQuery.sap.require("sap.ui.thirdparty.d3");
```

---

A vantagem da chamada acima é que ela pode ser usada para criação de dependências entre objetos. Caso você queira implementar um controle de interface gráfica customizado que necessite o d3 inicializado, seu controle só será carregado uma vez que o d3 o seja previamente.



Veja a documentação do método `jQuery.sap.require`.

As bibliotecas de terceiros vistas no capítulo anterior também são entregues em versões minificadas. Caso precise depurá-las, use o mesmo princípio aplicado no *bootstrap*.

#### Importando biblioteca D3

---

```
jQuery.sap.require("sap.ui.thirdparty.d3-dbg");
```

---



---

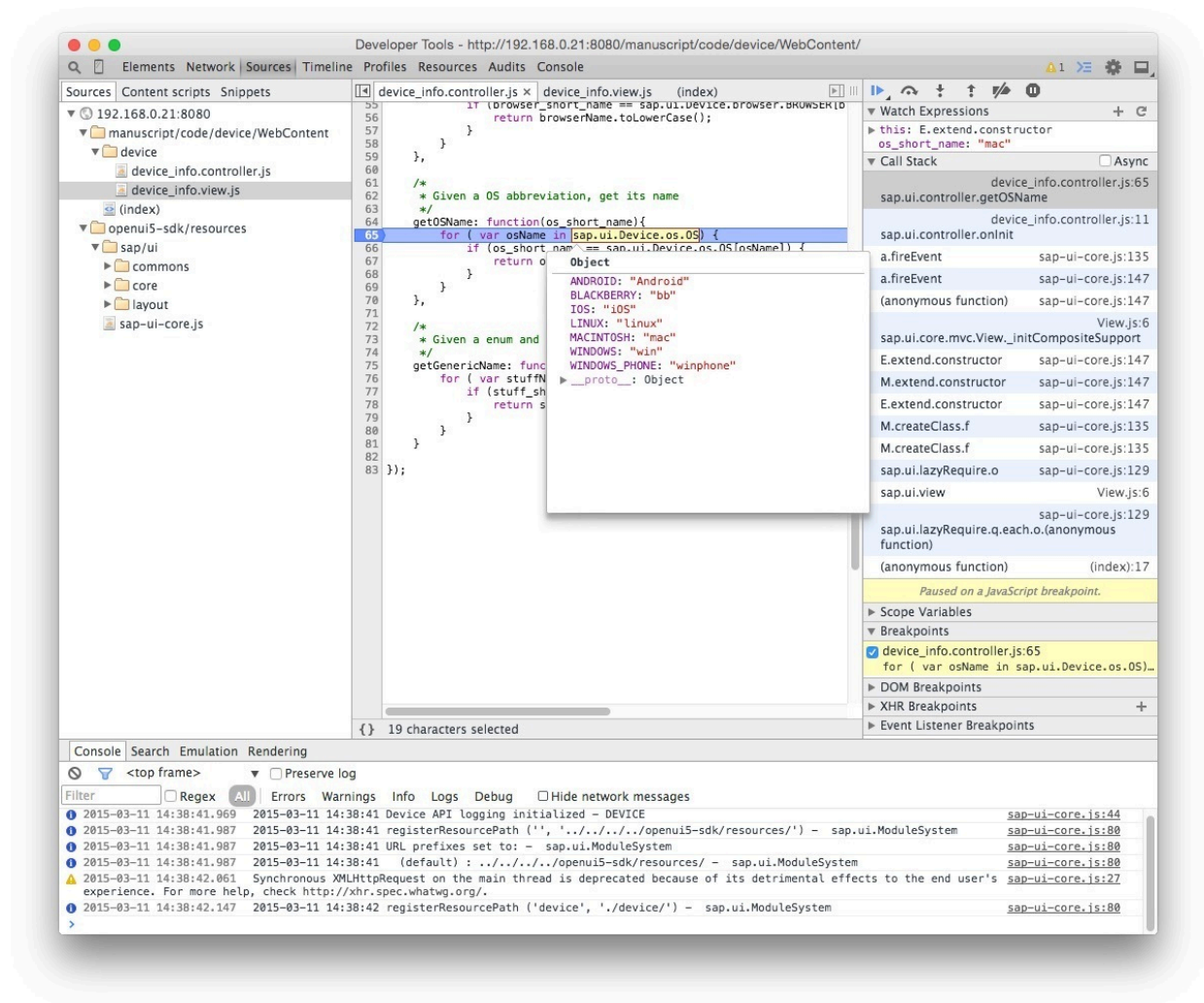
<sup>49</sup><http://cubiq.org/swipeview>

<sup>50</sup><http://www.eslinstructor.net/vkbeautify/>

<sup>51</sup><http://zynga.github.io/scroller/>

# Depurando um projeto em UI5

O código que escrevemos em UI5 é processado no *front end*. Podemos depurar nossa aplicação usando as ferramentas de desenvolvedor do Google Chrome ou Firefox por exemplo.

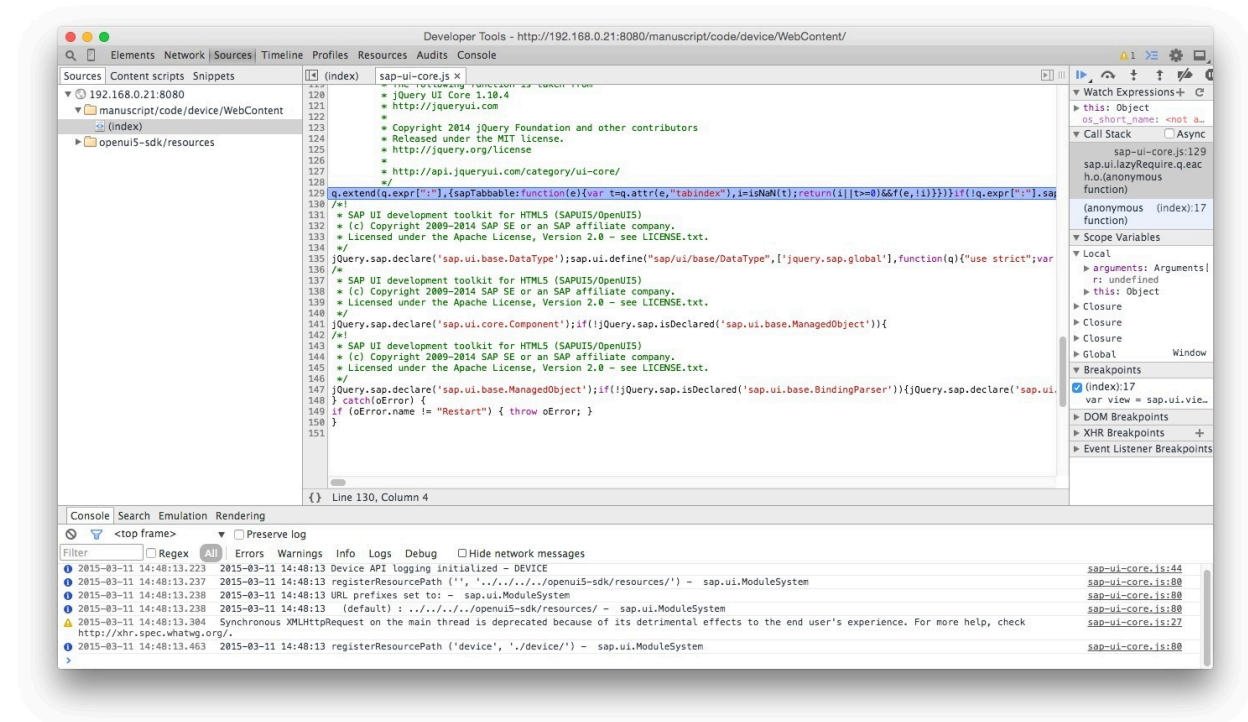


## Depurando *controller* usando o Google Chrome

Desenvolvedores ABAP estão acostumados a depurar código da SAP em alguns de seus produtos como o SAP ECC, SAP GRC, SAP CRM, etc. Esta é uma tarefa que poucos adoram mas que muitas vezes é necessária para identificar a causa raiz de algum incidente ou até mesmo desvendar como o sistema faz algum tipo de lógica que (tomara) possa ser reutilizado em uma aplicação customizada.

Também é possível depurar o código fonte do UI5 mas isso geralmente envolve uma pequena

mudança no código do *bootstrap*. Como via de regra as partes da biblioteca UI5 são carregadas a partir de arquivos minificados, fica difícil saber o que está se passando internamente.



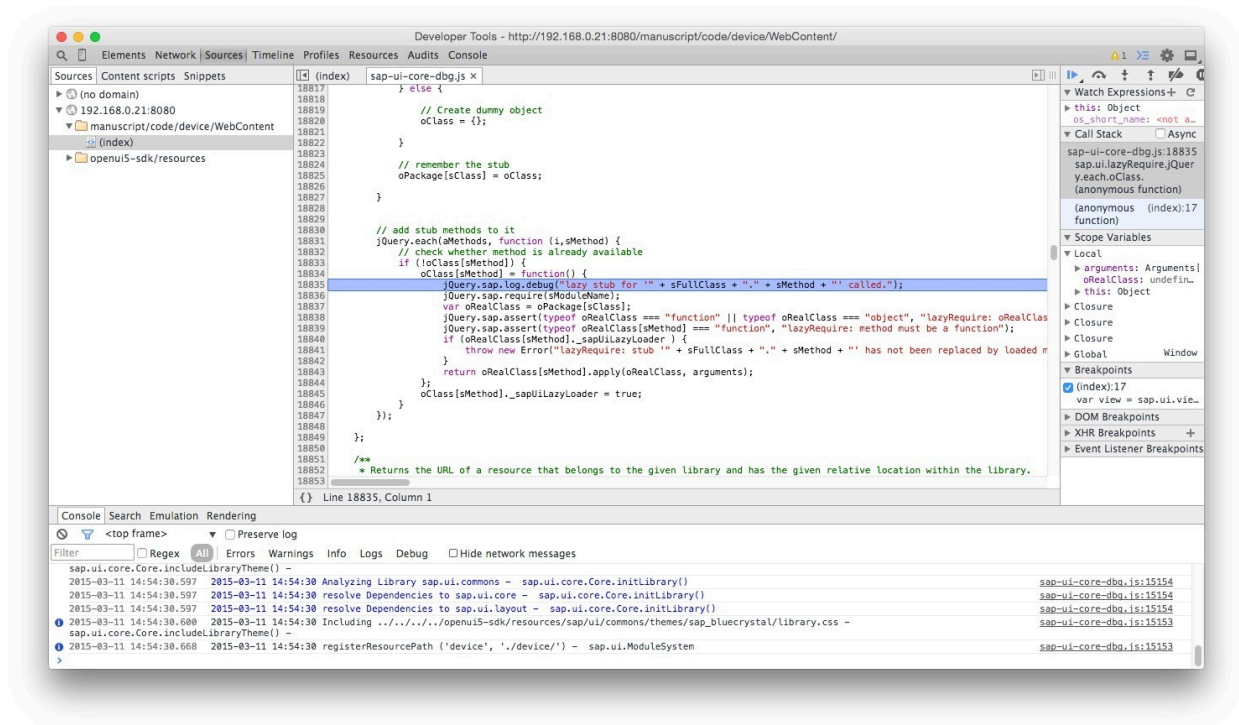
### Depurando *sap.ui.core* minificado - Péssima ideia



Muito código está na mesma linha (note a barra de rolagem horizontal). Isso otimiza e muito o tempo que o arquivo *sap-ui-core* é baixado, mas impossibilita que ele seja compreendido. O arquivo original possui mais de **40 mil** linhas enquanto a versão minificada possui apenas 150.

Os arquivos não-minificados do UI5 tem o sufixo **-dbg** em seu nome. Logo, basta alterar o nome do arquivo no *bootstrap* e poderemos entender em mais detalhes o que acontece internamente na biblioteca.

```
1 <script src="../../openui5-sdk/resources/sap-ui-core-dbg.js"
2       id="sap-ui-bootstrap"
3       data-sap-ui-libs="sap.ui.commons"
4       data-sap-ui-theme="sap_bluecrystal">
5 </script>
```



### Depurando *sap.ui.core* não-minificado - Boa ideia

Agora que você já sabe como depurar, que tal praticar estudando a função *getGenericName* do projeto *device* no depurador?

# Controles de UI

Controles de UI ou controles de interface gráfica são os elementos que o usuário usa como referência ou interage em uma aplicação: campos de entrada, rótulos, botões, botões de rádio, *checkboxes*, e muitos outros. Estes na verdade são apenas alguns exemplos dos controles mais simples disponíveis. O UI5 possui 7 diferentes tipos de controles:

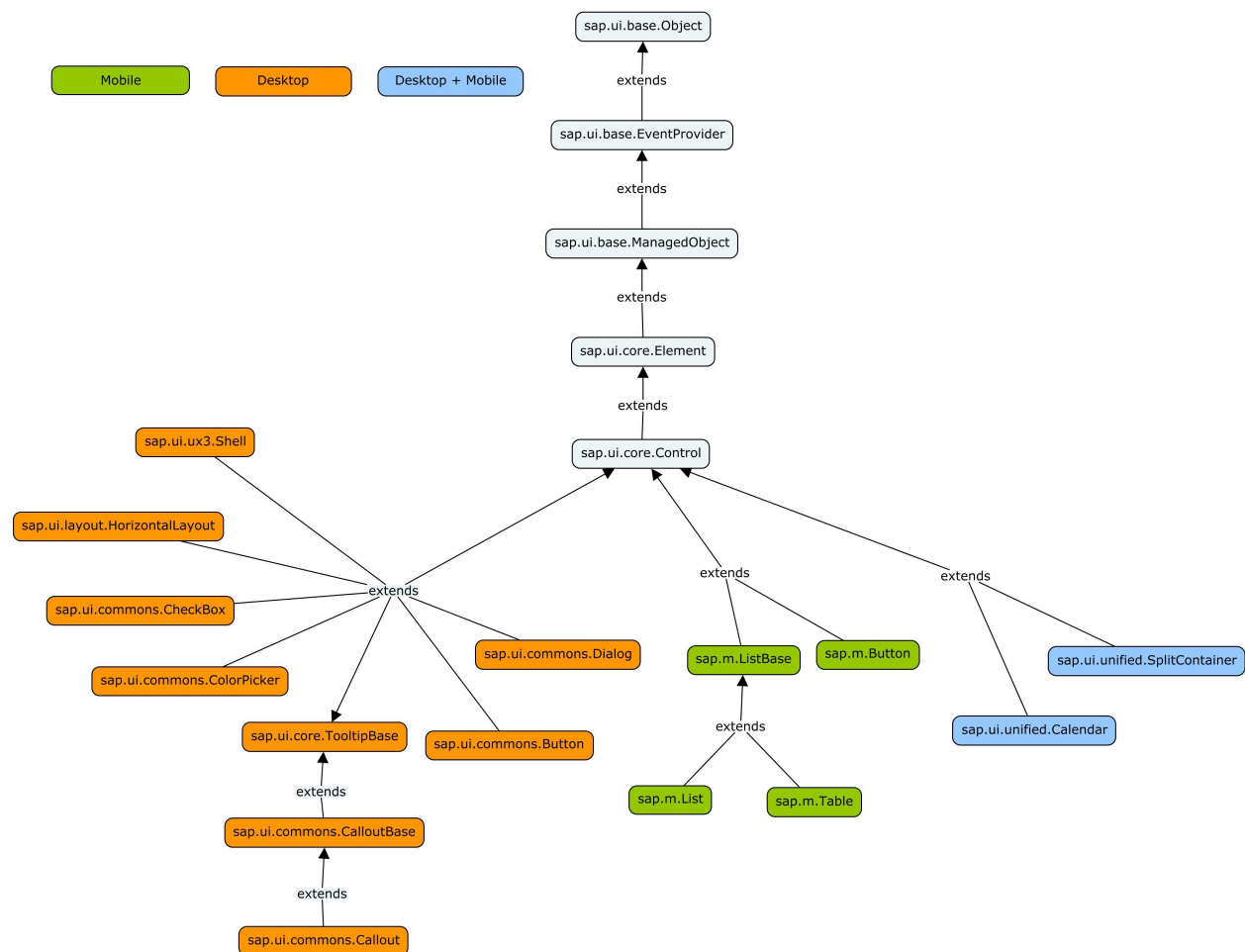
- **Controles simples** - Label, Button, Link, TextView, entre outros
- **Armazenadores de valores (ou Value Holders)** - ComboBox, ListBox, CheckBox, RadioButtonGroup, entre outros
- **Leiautes** - Formas de organizar outros controles: HorizontalLayout, VerticalLayout, MatrixLayout, entre outros
- **Complexos** - ColorPicker, Table, Menu, entre outros
- **Dialogs** - Dialog e MessageBox
- **UX3** - Semelhantes a controles complexos: QuickView, CollectionInspector, Shell, entre outros
- **Outros** - Callout, RichTooltip, LocalBusyIndicator

## Herança de controles

Todos os controles pré-definidos no UI5 direta ou indiretamente herdam da classe abstrata **sap.ui.core.Control**. O método *placeAt* utilizado na nossa primeira aplicação está definido nesta classe. Logo, todos os controles tem acesso a este método sendo possível inseri-los no DOM manualmente.

Alguns controles foram criados voltados para *Desktops* onde a entrada de informações é mais simples pelo tamanho dos monitores, praticidade do teclado e precisão do mouse. A grande maioria destes controles está do *namespace sap.ui.commons*. Outros controles foram criados para dispositivos móveis, onde a área de tela é menor e a interação do usuário não é tão precisa. Estes controles fazem parte do *namespace sap.m*. Mais recentemente, foram desenvolvidos controles capazes de se adaptar para *Desktops* ou dispositivos móveis conforme necessidade. Os últimos estão no *namespace sap.ui.unified*.

Abaixo é exibido um diagrama com a base da herança até algumas classes de controle de diferentes tipos.



## Controles de interface gráfica

## Elementos versus Controles

Elementos são partes de um controle complexo. Observando a imagem acima, vemos a classe *sap.ui.core.Element* sendo herdada pela classe *sap.ui.core.Control*. Por não serem completos, elementos não podem ser renderizados por si só.



Veja o construtor da classe na documentação da *SDK*.

Um exemplo de elemento é o *sap.ui.table.Column*, que representa uma coluna de um objeto do tipo *sap.ui.table.Table*, que por sua vez é um controle.

## ManagedObject

A classe mãe da *sap.ui.core.Element* representa um objeto “manipulável”. Estamos falando da classe *sap.ui.baseManagedObject* (veja a imagem da herança das classes).

Esta classe é de extrema importância pois ela define os tipos de propriedades que cada objeto abaixo de si na herança pode possuir (incluindo os controles de interface gráfica).

## Tipos de propriedades de um ManagedObject

Ao criar um objeto do tipo *sap.ui.baseManagedObject*, temos a opção de definir suas propriedades, que são divididas em quatro diferentes tipos:

- Simples
- Agregações
- Associações
- Eventos

Vamos usar alguns controles de interface gráfica para exemplificar os diferentes tipos de propriedades de um objeto do tipo *sap.ui.baseManagedObject*.



Veja o construtor da classe na documentação da *SDK*.

## Propriedades Simples

Propriedades simples possuem um e somente um valor com tipo simples (números, booleanos, e strings). Como exemplo, temos a propriedade *text* da classe *sap.ui.commons.TextView*.

text:Exemplo de propriedade simples

```
var hello = new sap.ui.commons.TextView({
    text: 'Hello World',
});
```

## Propriedades de Agregação

Propriedades de Agregação são composições de um conjunto de outros objetos. Estes conjuntos podem ter um limite de um ou mais objetos dependendo da propriedade em questão. Como exemplo, temos a propriedade *content* da classe *sap.ui.core.mvc.View*. Quando criamos uma visão do tipo JavaScript pelo Eclipse, esta propriedade é preenchida pelo método *createContent*.

### Exemplo de propriedade de agregação

---

```
createContent : function(oController) {  
    return new sap.ui.commons.TextView("txv_device_info");  
}
```

---

Outra maneira de preencher este tipo de propriedade é diretamente na instânciação de um novo objeto. Veja o seguinte exemplo. Nele, temos um controle do tipo *sap.ui.commons.RadioButtonGroup* que possui a propriedade de agregação *items*. Esta propriedade recebe um array de objetos do tipo *sap.ui.core.Item*.

### items:Exemplo de agregação preenchida na instânciação

---

```
var rad_group =  
    new sap.ui.commons.RadioButtonGroup({  
        items : [  
            new sap.ui.core.Item({  
                text : "Item 1",  
            }},  
            new sap.ui.core.Item({  
                text : "Item 2",  
            })  
        ]  
    });
```

---

## Propriedades de Associação

Alguns controles estão associados a outros. Diferentemente de agregações que podem aceitar um conjunto de objetos, uma propriedade do tipo associação somente pode ser preenchida por um e somente um objeto.



Associações múltiplas não são suportadas **ainda**. Isso pode ser feito no futuro.

Um exemplo simples de propriedade do tipo associação pode ser encontrado na classe *sap.ui.commons.Label*. Esta classe possui a propriedade *labelFor* que faz a associação do *Label* com algum outro (e somente **um** outro) controle qualquer.

### labelFor:Exemplo de propriedade de associação

---

```
var lab_name = new sap.ui.commons.Label({
    text : "Name",
    labelFor : txf_name
});

var txf_name = new sap.ui.commons.TextField();
```

---

## Propriedades de eventos

Muitos objetos são capazes de disparar eventos. Controles que possuem eventos são aqueles nos quais o usuário realizará alguma ação como um clique com o mouse ou toque com o dedo (no caso de dispositivos *touch*).

Propriedades de eventos são aquelas cujo valor são funções.

Um bom exemplo de uma propriedade de evento é encontrado na classe *sap.ui.commons.Button*. Esta classe possui a propriedade *press*, que recebe uma função. A chamada a esta função será feita quando o evento ocorrer. No caso, quando o usuário clica no botão.

### press:Exemplo de propriedade de evento

---

```
var but_press_me = new sap.ui.commons.Button({
    text : "Press me!",
    press : function() {
        alert("Button was pressed");
    }
});
```

---

Lembre-se que não é ideal definir funções com bastante código fonte nas visões pois isso fere o paradigma MVC.



Um bom uso de funções definidas em visões são funções anônimas que funcionam como *wrappers* de funções parametrizáveis definidas no *controller*.

press: Chamada de evento em método do *controller*

---

```
var but_press_me = new sap.ui.commons.Button({
    text : "Press me!",
    press : function() {
        alert("Button was pressed");
    }
    press : oController.onPressButton
});
```

---



Esteja sempre atento ao valor da referência *this* dentro das funções. Uma das maiores dificuldades de iniciantes em JavaScript é assumir valores desta referência indevidamente.

# Controles Simples

Controles simples são amplamente utilizados em aplicações. A maioria deste tipo de controle não permite entrada do usuário, servindo apenas para exibição de alguma informação (e.g. controles *TextViews* e *Image*). Outros tem o objetivo de dar ao usuário o poder de realizar alguma ação (e.g. controles *Button* e *Link*).



Uma forma interativa de navegar por todos os controles simples é através da página [#content/Controls/SimpleControls/index.html](#) na documentação da *SDK*.



Controles Simples

Para fins de estudo dos controles simples, recomenda-se criar um novo projeto no Eclipse chamado *controls\_simple* sem uma visão inicial. Como ainda não estudamos os controles de layout (*sap.ui.layout*), vamos organizar nossos controles usando tags `<div>` no bloco `<body>`.

No seu arquivo *index.html* faça as seguintes modificações:

1. Ajuste o caminho do *sap-ui-core.js* no *bootstrap*. Certifique-se de usar o *namespace sap.ui.commons*.
2. Modifique o conteúdo da tag `<body>` conforme abaixo:

## index.html

---

```
<body class="sapUiBody" role="application">
  <div>
    <span>TextView</span>
    <div id="divTextView"></div>
  </div>

  <br><br>

  <div>
    <span>FormattedTextView</span>
    <div id="divFormattedTextView"></div>
  </div>

  <br><br>

  <div>
    <span>Label</span>
    <div id="divLabel"></div>
  </div>

  <br><br>

  <div>
    <span>HTML</span>
    <div id="divHtml"></div>
  </div>

  <br><br>

  <div>
    <span>Button</span>
    <div id="divButton"></div>
  </div>

  <br><br>

  <div>
    <span>ToggleButton</span>
    <div id="divToggleButton"></div>
  </div>
```

```

<br><br>

<div>
    <span>Link</span>
<div id="divLink"></div>
</div>

<br><br>

<div>
    <span>Image</span>
<div id="divImage"></div>
</div>

<br><br>

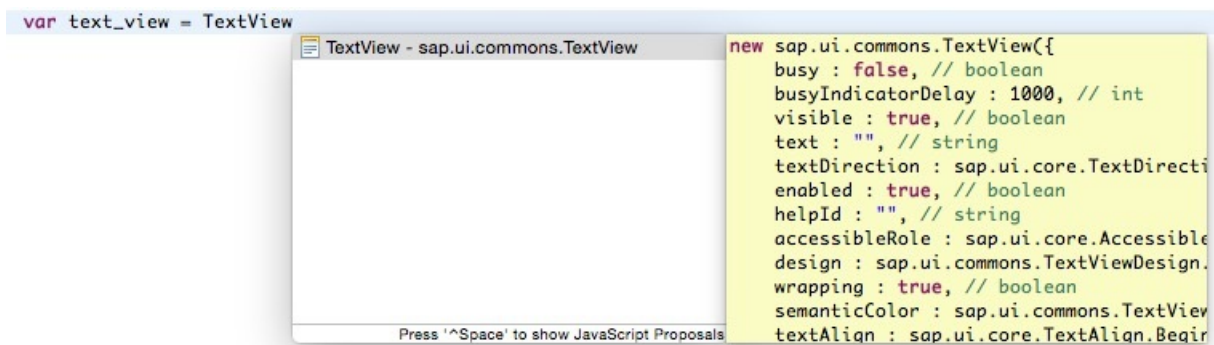
<div>
    <span>ImageMap</span>
<div id="divImageMap"></div>
</div>

</body>

```

1. Use o bloco <script> seguinte ao *bootstrap* para inserir todos os códigos de exemplos citados a partir de agora.

O *SAPUI5 Tools* instalado no seu Eclipse pode te ajudar a criar controles. Para isso, digite o nome de um controle e pressione “Control + Espaço” (Windows e Mac) para ter uma lista de sugestões.



Code completion - Lista de possibilidades

Então, escolha o controle desejado e aperte “Enter” para que o Eclipse gere o código base para você.

```

var text_view = new sap.ui.commons.TextView({
  busy : false, // boolean
  busyIndicatorDelay : 1000, // int
  visible : true, // boolean
  text : "", // string
  textDirection : sap.ui.core.TextDirection.Inherit, // sap.ui.core.TextDirection
  enabled : true, // boolean
  helpId : "", // string
  accessibleRole : sap.ui.core.AccessibleRole.Document, // sap.ui.core.AccessibleRole
  design : sap.ui.commons.TextViewDesign.Standard, // sap.ui.commons.TextViewDesign
  wrapping : true, // boolean
  semanticColor : sap.ui.commons.TextViewColor.Default, // sap.ui.commons.TextViewColor
  textAlign : sap.ui.core.TextAlign.Begin, // sap.ui.core.TextAlign
  width : undefined, // sap.ui.core.CSSSize
  tooltip : undefined, // sap.ui.core.TooltipBase
  customData : [ new sap.ui.core.CustomData({
    key : undefined, // string
    value : undefined, // any
    writeToDom : false, // boolean, since 1.9.0
    tooltip : undefined, // sap.ui.core.TooltipBase
    customData : [], // sap.ui.core.CustomData
    dependents : []
  // sap.ui.core.Control, since 1.19
  }) ], // sap.ui.core.CustomData
  dependents : [], // sap.ui.core.Control, since 1.19
  ariaDescribedBy : [], // sap.ui.core.Control
  ariaLabelledBy : []
// sap.ui.core.Control
})

```

Code completion - Resultado

## TextView

O controle *TextView* é capaz de exibir um texto sem formatação HTML em múltiplas linhas.

TextView

---

```

var text_view = new sap.ui.commons.TextView({
  text : "This book was written for special folks who are interested in:",
  wrapping : true,
  width:"50%",
  semanticColor : sap.ui.commons.TextViewColor.Critical,
  textAlign : sap.ui.core.TextAlign.Center,
  tooltip : 'This is a tooltip',
});

```

---

```
text_view.placeAt("divTextView");
```

---

As propriedades preenchidas são explicas abaixo:

- *text* - Define o texto do controle;

- *wrapping* - Permite se o texto pode ser quebrado em diversas linhas
- *width* - Largura do controle. No caso, 50% do tamanho do *parent*, que no caso é o `<div>divTextView`
- *semanticColor* - Uma formatação predefinida pelo UI5 (veja a classe estática *sap.ui.commons.TextViewColor* para conhecer os valores válidos)
- *textAlign* - Posicionamento do texto (a esquerda, direita, centralizado)
- *tooltip* - Caixa de informação padrão do navegador exibida quando o mouse é posicionado sobre o controle



Teste a sua aplicação e verifique o *TextView* criado. Mude o valor de algumas propriedade e veja qual o efeito causado. Repita este processo para cada um dos controles que serão vistos mais adiante.

## FormattedTextView

O controle *FormattedTextView* permite exibir um texto com formatações HTML em múltiplas linhas

### FormattedTextView

---

```
var some_html_text =
    "<ul>" +
        "<li>HTML</li>" +
        "<li>CSS</li>" +
        "<li>Javascript</li>" +
        "<li>SAP</li>" +
    "</ul>";

var formatted_text_view = new sap.ui.commons.FormattedTextView({
    htmlText : some_html_text
});

formatted_text_view.placeAt("divFormattedTextView");
```

---

As propriedades preenchidas são explicadas abaixo: \* *htmlText* string contendo o conteúdo do controle

## Label

O controle *Label* permite rotular um outro controle associado.

## Label

---

```
var label =  
    new sap.ui.commons.Label({  
        visible : true,  
        text : "Country",  
        required : true,  
        requiredAtBegin : true,  
        labelFor : text_view  
    });
```

```
label.placeAt("divLabel");
```

---

As propriedades preenchidas são explicadas abaixo:

- *visible* - Define se o controle é visível ao usuário
- *text* - Texto do controle
- *required* - Adiciona uma marcação (\*) indicando que o preenchimento do controle associado é obrigatório
- *requiredAtBegin* - Controla a posição da marcação de obrigatoriedade
- *labelFor* - Indica qual o controle associado ao *Label*

## HTML

O controle *HTML* permite pode ser entendido como um *wrapper* para um bloco de *tags* que é capaz de ser inserido em um arquivo HTML (no exemplo abaixo, um SVG). Este controle é muito útil quando se quer trabalhar com bibliotecas terceiras que manipulam o DOM, como jQuery e D3.

### HTML

---

```
var flag =  
    "<svg>" +  
        "<rect width='200' height='100' fill='#008000' />" +  
        "<polygon points='100,10 190,50 100,90 10,50' fill='#FFFF00' />" +  
        "<circle cx='100' cy='50' r='30' fill='#0000FF' />" +  
        "<line x1='70' y1='45' x2='130' y2='55' stroke='white' stroke-width='5' />" +  
    "</svg>";  
  
var html = new sap.ui.core.HTML({  
    content : flag,  
});  
  
html.placeAt("divHtml");
```

---

As propriedades preenchidas são explicadas abaixo:

- *content* - Conteúdo que pode ser embutido em HTML

## Button

O controle *Button* cria um botão. Como este é o primeiro controle no qual usaremos uma propriedade de evento, que tal um exemplo mais complexo?

### Button

---

```
var handler =
{
    onPressed: function(os, browser){
        alert(
            "You are using a " + browser +
            " installed on a " + os + " system"
        );
    }
};

var button = new sap.ui.commons.Button({
    text : "What am I using?",
    icon: "sap-icon://sys-monitor",
    press :
    [
        function(oEvent) {
            var control = oEvent.getSource();
            console.log(
                "Button pressed: " +
                control.getText()
            );
            this.onPressed(
                sap.ui.Device.os.name,
                sap.ui.Device.browser.name
            );
        },
        handler
    ]
});

button.placeAt("divButton");
```

---

As propriedades preenchidas são explicadas abaixo:

- *text* - Texto do controle
- *icon* - Ícone do controle
- *press* - Função chamada quando o usuário clica no botão. Além da função, é possível definir um objeto *listener* explicitamente



Esteja sempre atento ao valor da referência *this* dentro das funções. Uma das maiores dificuldades de iniciantes em JavaScript é assumir valores desta referência indevidamente.

No caso acima, a função chamada pelo evento encapsula outra função que apesar de ser chamada com a palavra reservada *this*, está num contexto diferente. Este é o mesmo procedimento que fazemos ao usar uma função definida no *controller* dentro de uma visão.



Ícones não são imagens mas sim fontes!!!. Para saber mais sobre ícones leia a documentação oficial do SAPUI5 [Using Icon Font in SAPUI5](#)<sup>52</sup>

## ToggleButton

O controle *ToggleButton* é um botão com estados “ligado” e “desligado”.

ToggleButton

```
var toggle_button =
    new sap.ui.commons.ToggleButton({
        text : "Off",
        lite : false,
        style : sap.ui.commons.ButtonStyle.Reject,
        pressed : false,
        press :
        [
            function(oEvent) {
                var control = oEvent.getSource();
                if (this.getPressed() === false) {
                    this.setStyle(sap.ui.commons.ButtonStyle.Accept);
                    this.setText("On");
                } else {
```

<sup>52</sup>[https://help.sap.com/saphelp\\_uiaddon10/helpdata/en/21/ea0ea94614480d9a910b2e93431291/content.htm](https://help.sap.com/saphelp_uiaddon10/helpdata/en/21/ea0ea94614480d9a910b2e93431291/content.htm)

```
        this.setStyle(sap.ui.commons.ButtonStyle.Reject);
        this.setText("Off");
    }
},
toggle_button
]
});

toggle_button.placeAt("divToggleButton");
```

---

As propriedades preenchidas são explicadas abaixo:

- *text* - Conteúdo do controle (que no exemplo é alterado posteriormente)
- *lite* - Muda o formato da exibição do botão quando o mesmo está “desligado”
- *style* - Uma formatação predefinida pelo UI5 (veja a classe estática *sap.ui.commons.ButtonStyle* para conhecer os valores válidos)
- *pressed* - Estado inicial do botão. Este estado é alterado automaticamente quando o usuário clica no botão
- *press* - Função a ser chamada quando o usuário clica no controle

Note que neste caso definimos o listener da função como sendo o próprio objeto do controle. Logo, quando usamos *this* na função estamos usando a referência ao objeto *ToggleButton*.

## Link

O controle *Link* representa um dos principais conceitos da web: um *hyperlink*.

Link

---

```
var link =
    new sap.ui.commons.Link({
        text : "Find out more",
        href : "//hanabrasil.com.br",
        target : "_blank"
    });

link.placeAt("divLink");
```

---

As propriedades preenchidas são explicadas abaixo:

- *text* - Texto do controle
- *href* - Referência que é chamada quando o link é clicado
- *target* - Define onde o link deve ser aberto (“\_blank” significa “nova aba/janela”);



Cuidado com a propriedade *href*! Caso a referência seja para um recurso externo use // antes no valor a ser usado.

## Image

O controle *Image* exibe uma imagem que pode ser carregada usando uma URI relativa ou absoluta. É possível carregar imagens de servidores remotos (em um outro domínio).

### Image

---

```
var image =  
    new sap.ui.commons.Image({  
        src : "http://openui5.org/images/OpenUI5_new_big_side.png"  
    });  
  
image.placeAt("divImage");
```

---

As propriedades preenchidas são explicadas abaixo:

- *src* - URI da imagem

## ImageMap

O controle *ImageMap* permite definir várias áreas clicáveis em uma imagem. Veja o exemplo abaixo

## ImageMap

---

```
var image_ui5_logos =
    new sap.ui.commons.Image({
        // image size: 217x131
        src : "http://abap101.com/wp-content/uploads/2015/02/sapui5-openui5.png",
    });

var image_map =
    new sap.ui.commons.ImageMap({
        name : "map1", // string
        areas :
        [
            new sap.ui.commons.Area({
                shape : "rect",
                coords : "0,76,217,131" ,
                href : "https://sapui5.hana.ondemand.com/sdk/",
                alt : "SAPUI5",
            }),
            new sap.ui.commons.Area({
                shape : "rect",
                coords : "0,0,217,75" ,
                href : "https://openui5.hana.ondemand.com/",
                alt : "OpenUI5",
            })
        ]
    });

image_ui5_logos.setUseMap("map1");
image_ui5_logos.placeAt("divImageMap");
image_map.placeAt("divImageMap");
```

---

As propriedades preenchidas são explicadas abaixo:

- *name* - Nome da área (é usada na propriedade *useMap* do controle *Image*)
- *areas* - Áreas da imagem

Para cada área, as seguintes propriedades foram preenchidas:

- *shape* - Forma da área
- *coords* - Coordenadas da área
- *href* - Referência a ser chamada quando a área é clicada
- *alt* - Texto alternativo exibido caso a imagem não seja carregada

## Outros controles simples

Praticamente esgotamos a lista de controles simples. O único controle que não usamos como exemplo é *ScrollBar*. Não temos nada contra ele... apenas não se prenda em decorar o funcionamento de cada controle. Ao contrário, aprenda o mecanismo dos controles. Todos eles são representados por propriedades dos 4 diferentes tipos que já estudamos. Este livro não visa ser um substituto para a documentação oficial da *SDK*. O UI5 conta com centenas de controles diferentes, e ainda é possível criar os seus próprios controles!

Também não usamos todas as propriedades de cada um dos controles explicados (somente as mais comuns). Sempre veja na documentação o que cada controle é capaz de fazer antes de usá-lo.



Crie mais um `<div>` no seu arquivo *index.html* e dentro dele crie um *ScrollBar* preenchendo pelo menos três de suas propriedades.

# Controles de Value Holders

Controles *value holders* (ou armazenadores de valor) também são amplamente utilizados em aplicações. A principal diferença deste conjunto de controles frente aos controles simples é que nos primeiros é permitida a entrada de informações por parte do usuário. O modo como esta entrada é feita varia de controle para controle. Em alguns o meio principal de entrada é o teclado (e.g. controles *TextFields* e *SearchField*). Outros usam o *mouse* como a principal forma de entrada (e.g. controles *ListBox* e *CheckBox*).



Uma forma interativa de navegar por todos os controles *value holders* é através da página [#content/Controls/ValueHolders/index.html](#) na documentação da *SDK*.



Controles *Value Holders*

Para fins de estudo dos controles *value holders*, recomenda-se criar um novo projeto no Eclipse chamado *controls\_value\_holders* sem uma visão inicial. Como ainda não estudamos os controles de leiaute (*sap.ui.layout*), vamos organizar nossos controles usando tags `<div>` no bloco `<body>`.

No seu arquivo *index.html* faça as seguintes modificações:

1. Ajuste o caminho do *sap-ui-core.js* no *bootstrap*. Certifique-se de usar o *namespace sap.ui.commons*.
2. Modifique o conteúdo da tag `<body>` conforme abaixo. Por questões de simplicidade o código abaixo somente contém *divs* para os dois primeiros controles.

index.html

---

```
<body class="sapUiBody" role="application">
    <div>
        <span>TextField</span>
        <div id="divTextField"></div>
    </div>

    <br><br>

    <div>
        <span>TextArea</span>
        <div id="divTextArea"></div>
    </div>

    <!-- Siga o mesmo padrão para cada um dos outros controles -->
</body>
```

---

1. Use o bloco `<script>` seguinte ao *bootstrap* para inserir todos os códigos de exemplos citados a partir de agora.

## TextField

O controle *TextField* representa um simples campo para entrada de valores. Ao contrário do *TextView*, o texto de um *TextField* é armazenado na propriedade *value* (e não *text*).

TextField

---

```
var text_field_first_name =
    new sap.ui.commons.TextField("txf_first_name",{
        value : "Fabio",
        editable : true,
        required : true,
        width : "30%",
        maxLength : 20,
        design : sap.ui.core.Design.Standard,
        name : "name",
        tooltip : "First Name",
        change :
        [
            function(oEvent) {
                console.log("change");
            }
        ]
    });
```

```
        var txf_last_name = sap.ui.getCore().byId("txf_last_name");
        var full_name =
            this.getValue() +
            " " +
            txf_last_name.getValue();

        var txf_full_name = sap.ui.getCore().byId("txf_full_name");
        txf_full_name.setValue(full_name);
    },
    text_field_first_name
]
});

text_field_first_name.placeAt("divTextField");

var text_field_last_name =
    new sap.ui.commons.TextField("txf_last_name",{
        width : "30%",
        design : sap.ui.core.Design.Monospaced,
        placeholder : "Last Name",
        liveChange :
        [
            function(oEvent) {
                console.log("liveChange");
                var txf_first_name = sap.ui.getCore().byId("txf_first_name");
                var full_name =
                    txf_first_name.getValue() +
                    " " +
                    oEvent.getParameter("liveValue");

                var txf_full_name = sap.ui.getCore().byId("txf_full_name");
                txf_full_name.setValue(full_name);
            },
            text_field_last_name
        ]
    });

text_field_last_name.placeAt("divTextField");

var text_field_full_name =
    new sap.ui.commons.TextField("txf_full_name",
    {
```

```
        width: "30%",  
        value: text_field_first_name.getValue(),  
        enabled : true,  
        editable : false,  
    });  
  
text_field_full_name.placeAt("divTextField");
```

---

As propriedades preenchidas são explicadas abaixo:

- *value* - Preenchimento do campo
- *editable* - Define se o campo está aberto para entrada de valores
- *enabled* - Não permite foco no elemento
- *required* - Caso o tema suporte, indica se o campo é obrigatório
- *width* - Largura do campo. Aceita qualquer unidade de medida CSS
- *maxLength* - Cumprimento máximo do valor
- *design* - Padrão ou monoespaçado
- *name* - Atributo *name* do HTML. Usado em chamadas *POST* a servidores web
- *tooltip* - Atributo *title* do HTML. Exibe uma pequena caixa com texto quando o cursor é repousado sobre o controle
- *placeholder* - Texto de referência exibido quando o controle está sem valor
- *change* - Evento que indica que valor foi alterado. É chamado quando o foco sai do controle
- *liveChange* - Qualquer caracter que seja inserido ou deletado do controle dispara este evento. Ao contrário do evento *change*, quando este evento é disparado a propriedade *value* não está atualizada. É necessário capturar o parâmetro *liveValue* usando o único argumento da função.

No caso acima, três *TextFields* foram criados. Os dois primeiros permitem a entrada de valores enquanto o último é atualizado com o resultado da concatenação dos seus antecessores. Quando o valor do primeiro campo é alterado e o usuário tira o foco do mesmo, o último *TextField* é atualizado usando o evento *change*. Já qualquer alteração feita no segundo campo é imediatamente refletida no último graças ao evento *liveChange*.

Note que ambos eventos permitem definir um objeto *listener* diferente do controle em si (segundo item do array passado nos eventos). Quando este objeto não é passado, a referência *this* aponta para o próprio controle em si. Quando um objeto *listener* é usado a referência *this* é ajustada para tal objeto.

A classe *TextField* possui algumas subclasses que representam campos de entrada com finalidades um pouco mais específicas. Atualmente há cinco subclasses: *ComboBox*, *DatePicker*, *PasswordField*, *TextArea* e *ValueHelpField*.

## TextArea

O controle *TextArea* representa um campo para entrada com várias linhas de texto.

### TextArea

---

```
var text_area =  
    new sap.ui.commons.TextArea({  
        maxLength : 140,  
        name : "bio",  
        cols : 45,  
        rows : 4,  
        wrapping : sap.ui.core.Wrapping.Hard,  
        cursorPos : 0,  
        explanation : "Some explanation",  
    });  
  
text_area.placeAt("divTextArea");
```

---

As propriedades preenchidas são explicadas abaixo:

- *maxLength* - Tamanho máximo do texto em caracteres
- *name* - Atributo *name* do HTML. Usado em chamadas *POST* a servidores web
- *cols* - Número de colunas
- *rows* - Número de linhas
- *wrapping* - Define como a quebra de texto é realizada
- *cursorPos* - Posicionamento do cursor
- *explanation* - Explicação caso ajuda-rápida esteja ligada

## PasswordField

O controle *PasswordField* representa um campo de senha, no qual o que é digitado não pode ser visto no valor do campo.

### PasswordField

---

```
var password_field =  
    new sap.ui.commons.PasswordField("pwd_field",{  
        required : true,  
        placeholder : "Password"  
    });  
  
password_field.placeAt("divPasswordField");
```

---

As propriedades preenchidas são explicadas abaixo:

- *required* - Caso o tema suporte, indica se o campo é obrigatório
- *placeholder* - Texto de referência exibido quando o controle está sem valor

## ValueHelpField

O controle *ValueHelpField* tem o funcionamento muito próximo a um *parameter* criado em ABAP. Um botão para a inserção de valores com algum tipo de ajuda é anexado ao campo texto. Esteticamente este botão é idêntico ao *matchcode* no ABAP. Em relação a codificação necessária, diferente do que é feito em ABAP, este controle de UI5 espera que a função de ajuda seja codificada a parte. Logo, não há um *popup* criado automaticamente quando o usuário clica no botão de ajuda.

### ValueHelpField

---

```
var value_help_field = new sap.ui.commons.ValueHelpField({  
    value : "",  
    valueHelpRequest :  
        function(oEvent) {  
            this.setValue("Value after help");  
        }  
});  
  
value_help_field.placeAt("divValueHelpField");
```

---

As propriedades preenchidas são explicadas abaixo:

- *value* - Preenchimento do campo
- *valueHelpRequest* - Função que é chamada quando o usuário clica no botão de ajuda de valores (“*matchcode*”)

## DatePicker

O controle de tela *DatePicker* é usado para inserção de datas. Em relação a sua aparência este é similar ao *ValueHelpField* pois contém um botão a direita do campo de entrada. Todavia, este botão por padrão exibe um pequeno calendário para a inserção de datas quando clicado.

O usuário pode inserir manualmente datas no formato AAAAMMDD (ano - mês - dia). Quando o foco é retirado do controle o seu valor é automaticamente ajustado de acordo com a propriedade *locale*.

### DatePicker

---

```
var yesterday = new Date();
yesterday.setDate(yesterday.getDate() - 1);
var date_picker = new sap.ui.commons.DatePicker({
    locale : 'pt-BR',
    yyyyymmdd : yesterday.toISOString().slice(0,10).replace(/-/g, ""),
});

date_picker.placeAt("divDatePicker");
```

---

As propriedades preenchidas são explicadas abaixo:

- *locale* - Define o idioma e país. Esta propriedade especifica o formato da data caso *data binding* não seja usado
- *yyyyymmdd* - Data inserida no controle, no formato AAAAMMDD.

No exemplo acima, o controle *DatePicker* é carregado automaticamente com a data do dia anterior ao dia atual. Como um objeto *Date* em JavaScript guarda a informação de hora e fuso-horário, é necessário fazer um tratamento para capturar apenas a data do mesmo antes de preencher a propriedade *yyyyymmdd* do controle.

## ComboBox

O controle *ComboBox* define um campo de entrada com uma lista de valores possíveis para uso.

## ComboBox

---

```
var combo_box = new sap.ui.commons.ComboBox({
    name : "country",
    placeholder : "Country",
    maxPopupItems : 4,
    displaySecondaryValues : true,
    selectedKey : "brazil",
    // selectedItemId : "item_default",
    items :
    [
        new sap.ui.core.ListItem({
            text : "Brazil",
            enabled : true,
            key : "brazil",
            additionalText : "South America",
        }),

        new sap.ui.core.ListItem("item_default",{
            text : "Canada",
            enabled : true,
            key : "canada",
            additionalText : "North America",
        }),

        new sap.ui.core.ListItem({
            text : "Germany",
            enabled : false,
            key : "germany",
            additionalText : "Europe",
        })
    ],
});

combo_box.placeAt("divComboBox");
```

---

As propriedades preenchidas são explicadas abaixo:

- *name* - Atributo *name* do HTML. Usado em chamadas *POST* a servidores web
- *placeholder* - Texto de referência exibido quando o controle está sem valor
- *maxPopupItems* - Número máximo de itens exibidos quando o *ComboBox* é aberto

- *displaySecondaryValues* - Cada item pode estar associado a um valor secundário. Esta propriedade define se estes valores são mostrados a direita dos itens
- *selectedKey* - Cada item pode estar associado a uma chave. Esta propriedade define o item selecionado de acordo com sua chave
- *selectedItemId* - Cada item pode conter um *id*. Esta propriedade define o item selecionado de acordo com este identificador
- *items* - Lista de itens do *ComboBox*

No caso acima os itens do *ComboBox* foram declarados internamente na própria criação do controle, o que não permite reuso. Caso fosse necessário criar vários controles do tipo *ComboBox* com exatamente os mesmos itens, a propriedade *listBox* (explicada mais adiante) seria mais indicada.

Os itens de um *ComboBox* podem ser carregados a partir de alguma classe de modelo (JSON, XML e oData). Neste caso o uso de *data binding* seria obrigatório.

Cada item do *ComboBox* é um objeto do tipo *sap.ui.core.ListItem*. As propriedades preenchidas para estes elementos são explicadas abaixo:

- *text* - Texto do item
- *enabled* - Define se o item pode ser escolhido ou não
- *key* - Chave que representa o item
- *additionalText* - Texto adicional do item. Este texto é exibido caso a propriedade *displaySecondaryValues* do *ComboBox* for verdadeira.

Além de ser subclasse de *TextField*, a classe *ComboBox* é a superclasse dos controles *AutoComplete* e *DropDownBox*.

## AutoComplete

O controle *AutoComplete* é semelhante ao campo de busca do Google pois permite fazer buscas instantâneas a uma lista de valores possíveis.

### AutoComplete

---

```
var auto_complete = new sap.ui.commons.AutoComplete({
    items:
    [
        new sap.ui.core.ListItem({
            text: "Fabio",
            key: "1",
        }),
        new sap.ui.core.ListItem({
```

```
        text: "Feliciana",
        key: "2",
    }},
    new sap.ui.core.ListItem({
        text: "Marcel",
        key: "3",
    }},
    new sap.ui.core.ListItem({
        text: "Marcelo",
        key: "4",
    }},
    new sap.ui.core.ListItem({
        text: "João",
        key: "5",
    })
],
});

auto_complete.placeAt("divAutoComplete");
```

---

As propriedades preenchidas são explicadas abaixo:

- *items* - Itens carregados enquanto o usuário digita sobre o controle

O controle *AutoComplete* possui ainda o evento *suggest* que é automaticamente disparado quando o usuário faz alguma alteração do campo que atualiza a lista de sugestões.

## DropDownBox

O controle *DropDownBox* pode ser entendido como um *ComboBox* misturado a um *ValueHelpField*. Ele possui uma lista de valores e também pode acionar um função de ajuda de preenchimento, que deve ser codificada a parte.

## DropDownBox

---

```
var dropdown = new sap.ui.commons.DropDownBox({
    name : "role",
    searchHelpEnabled : true,
    searchHelpText : "More",
    searchHelpAdditionalText : "F4",
    displaySecondaryValues : true,
    maxHistoryItems : 3,
    items :
        [
            new sap.ui.core.ListItem({
                text : "Intern",
                key : "intern",
            }),
            new sap.ui.core.ListItem({
                text : "Analyst",
                key : "analyst",
            }),
            new sap.ui.core.ListItem({
                text : "Developer",
                key : "developer",
            }),
            new sap.ui.core.ListItem({
                text : "Manager",
                key : "manager",
            }),
            new sap.ui.core.ListItem({
                text : "Tester",
                key : "tester",
            })
        ],

    searchHelp : function(oEvent) {
        var control = oEvent.getSource();
        // open search help form/dialog/view
        alert("Search Help");
        this.setSelectedKey("tester");
    }
});

dropdown.placeAt("divDropDownBox");
```

---

As propriedades preenchidas são explicadas abaixo:

- *name* - Atributo *name* do HTML. Usado em chamadas *POST* a servidores web
- *searchHelpEnabled* - Define se uma função de ajuda deve ser exibida. Ao contrário do *ValueHelpField*, o *matchcode* é exibido como o primeiro item da lista de possibilidades
- *searchHelpText* - Texto exibido para o item que representa a função de ajuda
- *searchHelpAdditionalText* - Texto adicional para o item que representa a função de ajuda
- *displaySecondaryValues* - Cada item pode estar associado a um valor secundário. Esta propriedade define se estes valores são mostrados a direita dos itens
- *maxHistoryItems* - O controle *DropDownBox* armazena um histórico de valores escolhidos localmente. Esta propriedade estabelece o número máximo de entradas neste histórico
- *items* - Lista de itens do *DropDownBox*
- *searchHelp* - Função de ajuda para preenchimento

## ListBox

O controle *ListBox* representa uma lista simples de valores. Ao contrário dos elementos citados anteriormente como o *ComboBox*, *AutoComplete* e *DropDownBox*, um controle do tipo *ListBox* é capaz de permitir a seleção de mais de um item.

### ListBox

---

```
var list_box = new sap.ui.commons.ListBox({
    allowMultiSelect : true,
    visibleItems : 4,
    items :
        [
            new sap.ui.core.Item({
                text : "Car",
                key : 'car',
            }),
            new sap.ui.core.Item({
                text : "Flight",
                key : 'flight',
            }),
            new sap.ui.core.Item({
                text : "Subway",
                key : 'subway-train',
            })
        ]
});
```

```

        ],
    });

list_box.placeAt("divListBox");

```

---

As propriedades preenchidas são explicadas abaixo:

- *allowMultiSelect* - Define se o usuário pode selecionar mais de um item
- *visibleItems* - Define quantos item são exibidos
- *items* - Itens do *ListBox*, cada um do tipo *sap.ui.core.Item* ou alguma classe filha a esta

O controle *ListBox* ainda pode ser usado quando é necessário reutilizar itens de uma lista em mais de um controle, como em dois *DropDownBox* conforme o exemplo abaixo.

#### ListBox reutilizada

---

```

var list_box_country = new sap.ui.commons.ListBox({
    items:
        [
            new sap.ui.core.Item({
                text: "Brazil",
                key: "brazil"
            }),
            new sap.ui.core.Item({
                text: "Canada",
                key: "canada"
            }),
            new sap.ui.core.Item({
                text: "Germany",
                key: "germany"
            })
        ]
});

// From
var label_country_from = new sap.ui.commons.Label({
    text : "From",
    labelFor : dropdown_country_from
});

var dropdown_country_from = new sap.ui.commons.DropDownBox({
    listBox : list_box_country,

```

```
});

// To
var label_country_to = new sap.ui.commons.Label({
    text : "To",
    labelFor : dropdown_country_from
});

var dropdown_country_to = new sap.ui.commons.DropdownBox({
    listBox : list_box_country,
});

label_country_from.placeAt("divListBoxReused");
dropdown_country_from.placeAt("divListBoxReused");
label_country_to.placeAt("divListBoxReused");
dropdown_country_to.placeAt("divListBoxReused");
```

---

A propriedade *listBox* de ambos *DropdownBox* foi preenchida com a referência ao mesmo controle do tipo *ListBox*. Assim, todos os itens deste controle são usados por ambos *DropdownBox*. Veja também que não é necessário renderizar o *ListBox* usando o método *placeAt*.

## InPlaceEdit

O controle *InPlaceEdit* é um dos meus preferidos. Ele representa um campo de texto que só pode ser editado caso o usuário clique no mesmo. Enquanto isso não é feito o controle esteticamente é semelhante a um *TextView*.

### InPlaceEdit

```
var in_place_edit = new sap.ui.commons.InPlaceEdit({
    valueState: sap.ui.core.ValueState.Success,
    content: new sap.ui.commons.TextField({
        value: "Focus on me!"
    })
});

in_place_edit.placeAt("divInPlaceEdit");
```

---

As propriedades preenchidas são explicadas abaixo:

- *valueState* - Marcador para o estado do controle
- *content* - Controle que é criado internamente quando o usuário clica no *InPlaceEdit*. Os controles válidos são: *TextField*, *ComboBox*, *DropdownBox* and *Link*.

## SearchField

Como seu nome já diz, o controle *SearchField* é apropriado para campos que tem uma função de busca. Um botão para tal finalidade é acrescido a campo texto. É ainda possível implementar sugestões para este campo. Para isso é necessário definir um *SearchProvider* que implementa o protocolo [OpenSearch](http://www.opensearch.org/Home)<sup>53</sup>.

### SearchField

---

```
var search_field = new sap.ui.commons.SearchField({
    enableListSuggest: true,
    showListExpander: true,
    enableClear: true,
    search: function(o){
        alert("Searching for " + o.getParameter("query"));
    },
});

search_field.placeAt("divSearchField");
```

---

As propriedades preenchidas são explicadas abaixo:

- *enableListSuggest* - Habilita a lista de sugestões
- *showListExpander* - Exibe lista de resultados expandida
- *enableClear* - Habilita o botão de limpar
- *search* - Função chamada quando o usuário realiza uma pesquisa

## CheckBox

O controle *CheckBox* representa uma única caixa para marcação. Não há segredo no uso deste controle.

---

<sup>53</sup><http://www.opensearch.org/Home>

## CheckBox

---

```
var check_box =
    new sap.ui.commons.CheckBox({
        checked : false,
        text : "I accept the terms of use",
        name : "terms",
        change :
            function(oEvent) {
                var control = oEvent.getSource();
                if (this.getChecked()) {
                    this.setValueState(sap.ui.core.ValueState.Success);
                } else {
                    this.setValueState(sap.ui.core.ValueState.Error);
                }
            }
    });

check_box.placeAt("divCheckBox");
```

---

As propriedades preenchidas são explicas abaixo:

- *checked* - Define se a caixa está marcada ou não
- *text* - Texto de referência da caixa
- *name* - Atributo *name* do HTML. Usado em chamadas *POST* a servidores web
- *change* - Evento disparado toda vez que o usuário marca ou desmarca o *CheckBox*

## TriStateCheckBox

O controle *TriStateCheckBox* representa um *CheckBox* com três valores: marcado, desmarcado e *misto*. Este tipo de *CheckBox* é útil quando há uma opção “Marcar/Desmarcar tudo” em uma aplicação. O estado *misto* não pode ser definido pelo usuário clicando no controle mas somente via chamadas no código.

### TriStateCheckBox

---

```
var tri_state_check_box =
    new sap.ui.commons.TriStateCheckBox("tri",{
        selectionState : sap.ui.commons.TriStateCheckBoxState.Mixed,
        text : "Select all/none",
        change :
            function(oEvent) {
                if (this.getSelectionState() ==
                    sap.ui.commons.TriStateCheckBoxState.Checked) {

                    sap.ui.getCore().byId("chk_1").setChecked(true);
                    sap.ui.getCore().byId("chk_2").setChecked(true);
                    sap.ui.getCore().byId("chk_3").setChecked(true);
                    sap.ui.getCore().byId("chk_4").setChecked(true);

                }
                else{
                    sap.ui.getCore().byId("chk_1").setChecked(false);
                    sap.ui.getCore().byId("chk_2").setChecked(false);
                    sap.ui.getCore().byId("chk_3").setChecked(false);
                    sap.ui.getCore().byId("chk_4").setChecked(false);

                };
            }

    });

var mixed = function(o){
    sap.ui.getCore()
        .byId("tri")
        .setSelectionState(
            sap.ui.commons.TriStateCheckBoxState.Mixed
        )
};

var checkboxes = [
    new sap.ui.commons.CheckBox("chk_1", {
        text: "CheckBox1",
        change: mixed,
    }),
    new sap.ui.commons.CheckBox("chk_2", {
        text: "CheckBox2",
        change: mixed,
    }),
    new sap.ui.commons.CheckBox("chk_3", {
```

```

        text: "CheckBox3",
        change: mixed,
    }},
    new sap.ui.commons.CheckBox("chk_4", {
        text: "CheckBox4",
        change: mixed,
    })
];

tri_state_check_box.placeAt("divTriStateCheckBox");

for (var current = 0; current < checkboxes.length; current++) {
    checkboxes[current].placeAt("divTriStateCheckBox");
}

```

---

As propriedades preenchidas são explicas abaixo:

- *selectionState* - Estado da seleção (marcado, desmarcado e misto)
- *text* - Texto do *TriStateCheckBox*
- *change* - Evento disparado toda vez que o usuário marca ou desmarca o *TriStateCheckBox*

Note que é necessário implementar a lógica de marcação/desmarcação em massa. No exemplo acima o controle *TriStateCheckBox* marca ou desmarca todos os *CheckBox* de acordo com seu próprio valor. Enquanto isso a marcação/desmarcação de qualquer um dos *CheckBox* simplesmente muda o valor do *TriStateCheckBox* para misto.

## RadioButton

O controle *RadioButton* representa um único botão de rádio. Este controle deve ser usado quando o número de opções a serem criadas é conhecido de antemão.

### RadioButton

---

```

var radio_button1 = new sap.ui.commons.RadioButton({
    text : "Inbound",
    groupName : "group_bound",
    selected: true,
    select : [ function(oEvent) {
        var control = oEvent.getSource();
    }, this ]
});

```

```
var radio_button2 = new sap.ui.commons.RadioButton({
    text : "Outbound",
    groupName : "group_bound",
    select : [ function(oEvent) {
        var control = oEvent.getSource();
    }, this ]
});

radio_button1.placeAt("divRadioButton");
radio_button2.placeAt("divRadioButton");
```

---

As propriedades preenchidas são explicadas abaixo:

- *text* - Texto do *RadioButton*
- *groupName* - Nome do grupo. *RadioButton* no mesmo grupo são desmarcados automaticamente quando um dos controles é marcado
- *selected* - Define se o controle está selecionado
- *select* - Evento disparado sempre que o usuário seleciona o *RadioButton*

## RadioButtonGroup

O controle *RadioButtonGroup* representa um conjunto de *RadioButton*. A vantagem deste controle em relação ao *RadioButton* é que seus itens podem ser associados a um modelo via *data binding*, o que viabiliza a criação de botões de rádio de acordo com conteúdo em formato JSON, XML e oData.

### RadioButtonGroup

---

```
var radio_button_group = new sap.ui.commons.RadioButtonGroup({
    columns : 3,
    items : [
        new sap.ui.core.Item({
            text : "Answer 1",
        })],
});

for (var i = 2; i < 10; i++) {
    radio_button_group.addItem(
        new sap.ui.core.Item({
            text : ("Answer " + i)
        })
    )
}
```

```
        );  
    }  
    radio_button_group.placeAt("divRadioButtonGroup");  
}
```

---

As propriedades preenchidas são explicadas abaixo:

- *columns* - Define quantas colunas serão usadas para renderização dos botões de rádio
- *items* - Itens do *RadioButtonGroup*

## Slider

O controle *Slider* define uma barra vertical ou horizontal representando um intervalo possível de valores e uma marcação que pode ser arrastada pelo usuário para a entrada de um valor

Slider

---

```
var slider_value = new sap.ui.commons.TextField({  
    enabled: false,  
    width: '3em',  
});  
  
var slider = new sap.ui.commons.Slider({  
    width: "30%",  
    min: 0,  
    max: 10,  
    value: 5,  
    smallStepWidth: 0.5,  
    stepLabels: true,  
    labels: ["0", "2.5", "5", "7.5", "10"],  
    liveChange:  
    [  
        function(o){  
            this.setValue(o.getParameter("value"));  
        },  
        slider_value  
    ],  
});  
slider_value.setValue(slider.getValue());  
  
slider.placeAt("divSlider");  
slider_value.placeAt("divSlider");
```

---

As propriedades preenchidas são explicadas abaixo:

- *width* - Largura da barra
- *min* - Valor mínimo
- *max* - Valor máximo
- *value* - Valor atual
- *smallStepWidth* - Tamanho mínimo de um “passo”. Um passo de 0.5 significa que entre os valores 0 e 10 há 20 passos.
- *stepLabels* - Define se os passos são rotulados
- *labels* - Rótulos a serem exibidos. O controle faz automaticamente o alinhamento de acordo com a quantidade de elementos neste array.
- *liveChange* - Evento que é disparado sempre que o valor do *Slider* é alterado

## RangeSlider

O controle *RangeSlider* é uma especialização do *Slider* na qual o usuário define dois valores (mínimo e máximo) entre um intervalo de valores possíveis.

### RangeSlider

---

```
var range_slider = new sap.ui.commons.RangeSlider({
    height: "300px",
    vertical: true,
    min: 0,
    max: 10,
    value: 2.5,
    value2: 7.5,
    stepLabels: true,
    labels: ["0", "2.5", "5", "7.5", "10"],
});
```

```
range_slider.placeAt("divRangeSlider");
```

---

As propriedades preenchidas são explicadas abaixo:

- *height* - Altura da barra
- *vertical* - Define que o *Slider* é vertical
- *min* - Valor mínimo
- *max* - Valor máximo
- *value* - Valor mínimo atual

- *value2* - Valor máximo atual
- *stepLabels* - Define se os passos são rotulados
- *labels* - Rótulos a serem exibidos. O controle faz automaticamente o alinhamento de acordo com a quantidade de elementos neste array.

## RatingIndicator

O *RatingIndicator* é um excelente controle para coletar algum tipo de *feedback* do usuário de forma rápida. Este controle desenha estrelas que correspondem a alguma nota indicada pelo usuário.

### RatingIndicator

---

```
var rating_indicator = new sap.ui.commons.RatingIndicator({  
    maxValue: 7,  
    visualMode: sap.ui.commons.RatingIndicatorVisualMode.Half ,  
});  
  
rating_indicator.placeAt("divRatingIndicator");
```

---

As propriedades preenchidas são explicadas abaixo:

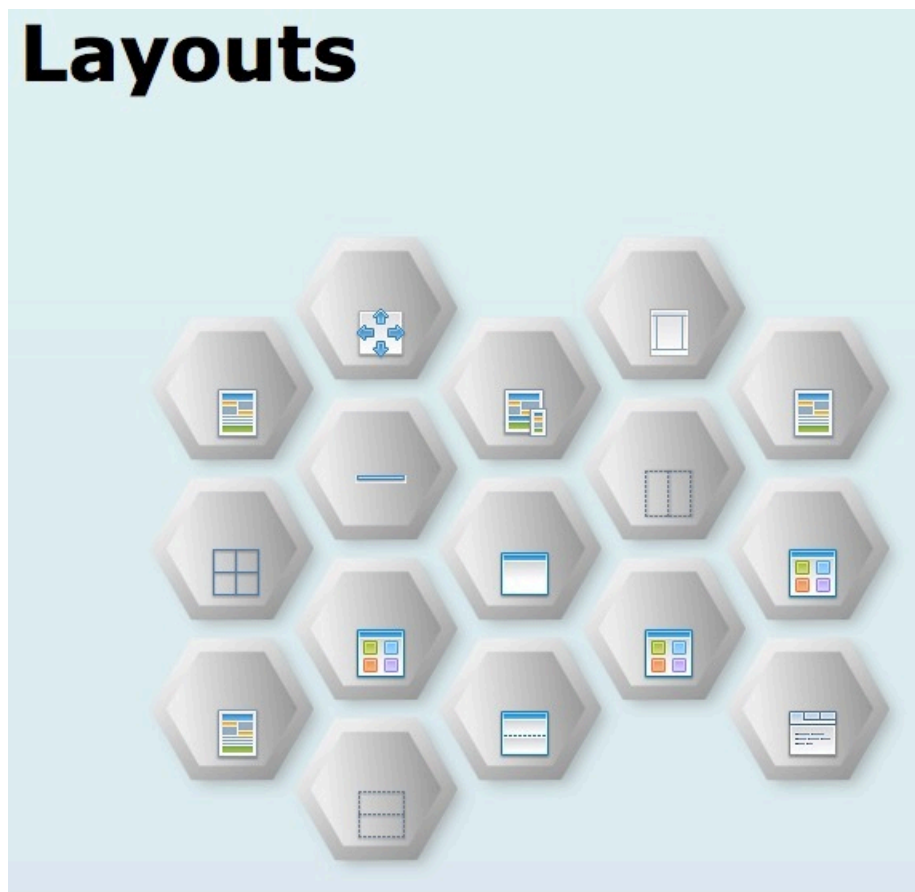
- *maxValue* - Número de estrelas que representam o valor máximo da nota
- *visualMode* - Definir como o arredondamento da nota é realizado

# Controles de Leiaute (Layout)

Controles de leiaute permitem organizar a posição e ordem de outros controles. Até o momento criamos controles diretamente abaixo de tags `<div>`. Todavia, não é necessário (nem recomendável) organizar o leiaute de uma aplicação UI5 usando somente tags de HTML e CSS. Ao invés disso, cria-se controles de leiaute os quais serão responsáveis pela organização de seu próprio conteúdo.



Uma forma interativa de navegar por todos os controles simples é através da página [#content/Controls/Layout/index.html](#) na documentação da *SDK*.



Controles de Leiaute



Alguns controles de leiaute usados neste capítulo pertencem ao *namespace* `sap.ui.layout`.

Para fins de estudo dos controles de leiaute, recomenda-se criar um novo projeto no Eclipse chamado *controls\_layout* com uma visão própria para cada controle de leiaute a ser estudado. Ao criar o projeto use a biblioteca *sap.ui.commons*. E dê o nome da sua visão inicial de *vertical\_layout*.

No seu arquivo *index.html* faça as seguintes modificações:

1. Ajuste o caminho do *sap-ui-core.js* no *bootstrap*. Certifique-se de usar o *namespace sap.ui.commons*.
2. Desta vez, não altere o conteúdo da tag `<body>`.
3. Dentro da tag `<script>` após o *bootstrap*, modifique o *id* da visão inicial para *idview* e conforme aprender novos layouts, modifique o atributo *viewName* conforme abaixo.

*index.html*

---

```
<!DOCTYPE HTML>
<html>
  <head>
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta http-equiv='Content-Type' content='text/html; charset=UTF-8' />

    <script src="../../../../openui5-sdk/resources/sap-ui-core.js"
            id="sap-ui-bootstrap"
            data-sap-ui-libs="sap.ui.commons"
            data-sap-ui-theme="sap_bluecrystal">
    </script>
    <!-- add sap.ui.table, sap.ui.ux3 and/or other libraries to 'data-sap-ui-\
libs' if required -->

    <script>
      sap.ui.localResources("controls_layout");

      var view = sap.ui.view({
        id:"idview",
        viewName:"controls_layout.vertical_layout",
        // viewName:"controls_layout.horizontal_layout",
        // viewName:"controls_layout.horizontal_divider",
        // viewName:"controls_layout.xxx",
        type:sap.ui.core.mvc.ViewType.JS
      });

      view.placeAt("content");
    </script>

  </head>
```

```
<body class="sapUiBody" role="application">
  <div id="content"></div>
</body>
</html>
```

---



Ao alterar o atributo *viewName*, não esqueça de comentar as outras ocorrências do mesmo.

## Vertical Layout

O leiaute *Vertical Layout* permite organizar controles um abaixo do outro.

Dentro da sua visão *vertical\_layout.view.js*, altere o método *createContent* conforme abaixo.

### Vertical Layout

---

```
createContent: function(oController)
{
    var text_field = new sap.ui.commons.TextField(
    {
        value: "Fabio",
    });

    var vertical_layout = new sap.ui.layout.VerticalLayout(
    {
        content: [
            new sap.ui.commons.Label(
            {
                text: "Name",
                labelFor: text_field
            }),
            text_field
        ]
    })

    vertical_layout.addContent(
        new sap.ui.commons.CheckBox(
        {
            text: "Writting this book during Easter?",
            value: true,
```

```
    })
  );

  var link = sap.ui.commons.Link(
  {
    text: "hanabrasil.com.br",
    href: "http://www.hanabrasil.com.br",
    target: "_blank"
  });

  vertical_layout.insertContent(link, 2);

  return vertical_layout;
}
```

---

As propriedades preenchidas são explicadas abaixo:

- *content* - Conteúdo do layout. Cada controle posto no array é adicionado em ordem, um abaixo do outro. Note que é possível criar um controle *on the fly* ou passar uma referência a um controle já existente

Alguns métodos úteis do controle também foram usados \* *addContent* - Adiciona um controle por último no leiaute. \* *insertContent* - Insere um controle na posição especificada pelo segundo parâmetro. A posição inicia em zero.

## Horizontal Layout

O leiaute *Horizontal Layout* permite definir controles um a direita do outro. É possível controlar também a quebra de linha dos controles caso a largura do navegador não comporte todos o conteúdo horizontal.

Crie uma nova visão chamada *horizontal\_layout* e modifique o método *createContent* conforme abaixo.

Antes de testar a sua nova visão, não esqueça de alterar o atributo *viewName* no arquivo *index.html*.

### Horizontal Layout

---

```
createContent: function(oController)
{
    var text_field = new sap.ui.commons.TextField(
    {
        value: "Fabio",
    });

    var horizontal_layout = new sap.ui.layout.HorizontalLayout(
    {
        allowWrapping: true,
        content: [
            new sap.ui.commons.Label(
            {
                text: "Name",
                labelFor: text_field
            }),

            text_field
        ]
    })

    horizontal_layout.addContent(
        new sap.ui.commons.CheckBox(
        {
            text: "Writting this book during Easter?",
            value: true,
        })
    );

    var link = sap.ui.commons.Link(
    {
        text: "hanabrasil.com.br",
        href: "http://www.hanabrasil.com.br",
        target: "_blank"
    });

    horizontal_layout.insertContent(link, 2);

    return horizontal_layout;
}
```

---

As propriedades preenchidas são explicadas abaixo:

- *allowWrapping* - Define se há quebra de linha caso a largura do navegador não seja suficiente para exibir todos controles
- *content* - Conteúdo do layout. Cada controle posto no array é adicionado em ordem, um a direita do outro. Note que é possível criar um controle *on the fly* ou passar uma referência a um controle já existente

## Horizontal Divider

Desde os tempos mais primórdios do HTML, a tag `<hr>` era usada para separar partes de uma página através de uma linha horizontal. O controle *Horizontal Divider* renderiza uma linha do tipo com exatamente a mesma tag. Logo, este controle não possui conteúdo em si mas ajuda na estruturação de um leiaute.

Em uma nova visão chamada *horizontal\_divider*, insira o código que segue.

### Horizontal Divider

---

```
createContent: function(oController)
{
    var main_layout = new sap.ui.layout.VerticalLayout(
    {
        content: [
            new sap.ui.commons.TextView(
            {
                text: "Area 1"
            }),

            new sap.ui.commons.HorizontalDivider(
            {
                type: sap.ui.commons.HorizontalDividerType.Area,
                height: sap.ui.commons.HorizontalDividerHeight.Large
            }),

            new sap.ui.commons.TextView(
            {
                text: "Area 2 - Page 1"
            }),

            new sap.ui.commons.HorizontalDivider(
            {
                type: sap.ui.commons.HorizontalDividerType.Page,
```

```
        height: sap.ui.commons.HorizontalDividerHeight.Ruleheight
    })),

    new sap.ui.commons.TextView(
    {
        text: "Area 2 - Page 2"
    })
    ]
});

return main_layout;
}
```

---

As propriedades preenchidas são explicadas abaixo:

- *type* - Tipo de divisor: *Area* ou *Page*. No tema *blue\_crystal* a diferença entre estes dois tipos é meramente estética e sutil. O divisor de área é uma linha horizontal simples com apenas uma cor enquanto o divisor de página possui cores diferentes para o topo e rodapé da borda. (Isso é feito através das propriedades *border-top-color* e *border-bottom-color* do CSS).
- *height* - Altura do divisor incluindo as margens superior e inferior. Possibilidades: *Small*, *Medium*, *Large* e *Ruleheight*. O último não insere margens.

## Panel

O controle *Panel* é um agrupador de controles com algumas especificidades porém que não organiza seu conteúdo. Ele contém uma barra superior que pode conter um título e um conjunto de botões. Esta barra pode ser contraída pelo usuário clicando no botão *collapse* caso ele seja exibido. Logo abaixo desta barra o conteúdo do *Panel* é exibido.

Em uma nova visão chamada *panel\_layout*, insira o código que segue.

### Panel

---

```
createContent: function(oController)
{

var panel_layout = new sap.ui.commons.Panel(
{
    width: "auto",
    height: "auto",
    scrollLeft: 0,
    scrollTop: 0,
```

```
    applyContentPadding: true,
    collapsed: false,
    areaDesign: sap.ui.commons.enums.AreaDesign.Plain,
    borderDesign: sap.ui.commons.enums.BorderDesign.Box,
    showCollapseIcon: true,
    // text: "Employee",
    content: [
        new sap.ui.layout.VerticalLayout(
            {
                content: [
                    new sap.ui.commons.Label(
                        {
                            labelFor: "txf_name",
                            text: "Name",
                        }
                    ),
                    new sap.ui.commons.TextField("txf_name")
                ]
            }
        ),
        title: new sap.ui.core.Title(
            {
                text: "Employee",
                icon: "sap-icon://employee"
            }
        ),
        buttons: [
            new sap.ui.commons.Button(
                {
                    text: "Create"
                }
            ),
            new sap.ui.commons.Button(
                {
                    text: "Change"
                }
            ),
            new sap.ui.commons.Button(
                {
                    text: "Delete"
                }
            )
        ]
    ]
});

return panel_layout;
```

---

```
}
```

As propriedades preenchidas são explicadas abaixo:

- *width* - Largura do *Panel*. Use “auto” para que ele se ajuste de acordo com seu conteúdo
- *height* - Altura do *Panel*. Use “auto” para que ele se ajuste de acordo com seu conteúdo
- *scrollLeft* - Caso o *Panel* possua uma rolagem horizontal, é possível controlar a posição da rolagem usando esta propriedade
- *scrollTop* - Caso o *Panel* possua uma rolagem vertical, é possível controlar a posição da rolagem usando esta propriedade
- *applyContentPadding* - Aplica um *padding* aos controles internos. Use esta opção para não deixar os controles colados a esquerda do *Panel*
- *collapsed* - Define se o *Panel* está contraído ou não
- *areaDesign* - Desenho da área interna
- *borderDesign* - Desenho da borda do *Panel*
- *showCollapseIcon* - Define se o botão para contração (*collapse*) do *Panel* é exibido
- *text* - Texto usado como título caso a propriedade *title* não seja usada
- *content* - Conteúdo do *Panel*. Use outro leiaute internamente para organizar o conteúdo
- *title* - Título do *Panel*
- *button* - Barra de botões do *Panel*

## Border Layout

O leiaute *Border Layout* cria um *container* dividido em 5 partes: *Top* (Topo), *Begin* (a esquerda), *Center* (Meio), *End* (direita) e *Bottom* (rodapé).

### Border Layout

---

```
createContent: function(oController)
{
    var border_layout = new sap.ui.commons.layout.BorderLayout(
    {
        width: "500px",
        height: "500px",
        top: new sap.ui.commons.layout.BorderLayoutArea(
        {
            overflowX: "visible",
            overflowY: "visible",
            contentAlign: "center",
```

```
        content: [
            new sap.ui.commons.Image(
                {
                    src: "http://openui5.org/images/OpenUI5_new_big_side.png"
                })
        ]
    )),
    begin: new sap.ui.commons.layout.BorderLayoutArea(
    {
        overflowX: "visible",
        overflowY: "visible",
        contentAlign: "center",
        content: [
            new sap.ui.commons.Link(
                {
                    text: "Begin"
                })
        ]
    )),
    center: new sap.ui.commons.layout.BorderLayoutArea(
    {
        overflowX: "visible",
        overflowY: "visible",
        contentAlign: "center",
        content: [
            new sap.ui.commons.TextArea(
                {
                    value: "Center",
                    rows: 20
                })
        ]
    )),
    end: new sap.ui.commons.layout.BorderLayoutArea(
    {
        overflowX: "visible",
        overflowY: "visible",
        contentAlign: "center",
        content: [
            new sap.ui.commons.CheckBox(
                {
                    text: "End"
                })
        ]
    })
```

```
    ]
  }},
  bottom: new sap.ui.commons.layout.BorderLayoutArea(
  {
    overflowX: "visible",
    overflowY: "visible",
    contentAlign: "center",
    content: [
      new sap.ui.commons.Button(
      {
        text: "Bottom",
        width: "100%"
      })
    ]
  })
});

return border_layout;
}
```

---

As propriedades preenchidas são explicadas abaixo:

- *width* - Largura do leiaute
- *height* - Altura do leiaute
- *top* - Controles que fazem parte do topo do *container*
- *begin* - Controles que fazem parte da esquerda do *container*
- *center* - Controles que fazem parte do centro do *container*
- *end* - Controles que fazem parte da direita do *container*
- *bottom* - Controles que fazem parte do rodapé do *container*

## Matrix Layout

O leiaute *Matrix Layout* permite definir um conjunto de controles em forma de matriz, com linhas e colunas alinhadas cuja dimensão é NxM. A diferença de um leiaute *Matrix* para as matrizes que aprendemos na escola é que no caso do UI5 um item da matriz pode ocupar mais de uma linha ou coluna. Para nossa alegria também não precisamos descobrir uma determinante.

## Matrix Layout

---

```
createContent: function(oController)
{
    var matrix_layout = new sap.ui.commons.layout.MatrixLayout(
    {
        width: "500px",
        // height: "500px",
        layoutFixed: true,
        columns: 3,
        widths: ["10%", "20%", "70%"],
        rows: []

    });

    this.myCreateRow(matrix_layout);
    this.myCreateRow(matrix_layout);
    this.myCreateRow(matrix_layout);

    return matrix_layout;
},

current_value: 1,

myCreateCell: function(row, value)
{
    var cell = new sap.ui.commons.layout.MatrixLayoutCell(
    {
        backgroundDesign: sap.ui.commons.layout.BackgroundDesign.Fill1,
        colSpan: 1,
        hAlign: sap.ui.commons.layout.HAlign.Center,
        padding: sap.ui.commons.layout.Padding.End,
        rowSpan: 1,
        separation: sap.ui.commons.layout.Separation.SmallWithLine,
        vAlign: sap.ui.commons.layout.VAlign.Middle,
        content: [
            new sap.ui.commons.TextView(
            {
                text: value,
            })
        ]
    });
};
```

```
        row.addCell(cell);
    },

    myCreateRow: function(matrix)
    {
        var row = new sap.ui.commons.layout.MatrixLayoutRow(
        {
            height: "100px",
            cells: []
        });

        matrix.addRow(row);

        this.myCreateCell(row, this.current_value++);
        this.myCreateCell(row, this.current_value++);
        this.myCreateCell(row, this.current_value++);
    }
}
```

---

As propriedades preenchidas são explicadas abaixo:

- *width* - Largura do leiaute
- *height* - Altura do leiaute (caso não seja especificado pode-se usar a altura das linhas)
- *layoutFixed* - controla se a largura das colunas é definida pelo leiaute ou pelo conteúdo das colunas
- *columns* - Número de colunas do leiaute
- *widths* - Um array com a largura das colunas
- *rows* - Array representando as linhas do leiaute. Cada linha é uma instância do tipo *MatrixLayoutRow*

## MatrixLayoutRow

Define uma linha de um *MatrixLayout*. Uma linha pode ser entendida como nada mais nada menos que um conjunto de células do tipo *MatrixLayoutCell*.

As propriedades preenchidas são explicadas abaixo:

- *height* - Altura da linha
- *cells* - Conjunto de células

## MatrixLayoutCell

Uma célula de um *MatrixLayout* pode ocupar mais de uma linha ou coluna no leiaute. Cada célula pode por sua vez possuir zero ou mais controles.

As propriedades preenchidas são explicadas abaixo:

- *backgroundDesign* - Define o plano de fundo da célula
- *colSpan* - Quantidade de colunas que a célula ocupa no leiaute
- *hAlign* - Alinhamento horizontal do conteúdo da célula
- *padding* - Espaçamento interno da célula
- *rowSpan* - Quantidade de linhas que a célula ocupa no leiaute
- *separation* - Seperador entre células
- *vAlign* - Alinhamento vertical
- *content* - Conteúdo da célula (array de controles)

## Splitter

O controle *Splitter* separa horizontalmente ou verticalmente dois conjuntos de controles.

### Splitter

---

```
createContent: function(oController)
{
    var splitter_layout = new sap.ui.commons.Splitter(
    {
        splitterOrientation: sap.ui.core.Orientation.Vertical,
        splitterPosition: "50%",
        minSizeFirstPane: "20%",
        minSizeSecondPane: "10%",
        width: "600px",
        // height: "300px",
        showScrollBars: false,
        splitterBarVisible: true,
        firstPaneContent: [
            new sap.ui.commons.Image(
            {
                src: "http://openui5.org/resources/OpenUI5_text_right_small.png"
            })
        ],
        secondPaneContent: [
            new sap.ui.commons.Image(
```

```
        {
            src: "//openui5.org/resources/OpenUI5_text_below_small.png"
        })
    ]
});

return splitter_layout;
}
```

---

As propriedades preenchidas são explicadas abaixo:

- *splitterOrientation* - Orientação do divisor: vertical ou horizontal
- *splitterPosition* - Posição inicial (ou atual) do *splitter*, use 50% para inicializar o divisor no meio dos dois painéis.
- *minSizeFirstPane* - Tamanho mínimo do primeiro painel (a esquerda ou em cima)
- *minSizeSecondPane* - Tamanho mínimo do primeiro painel (a direita ou em baixo)
- *width* - Largura do *Splitter*
- *height* - Altura do *Splitter*. Ajusta-se de acordo com seu conteúdo
- *showScrollBars* - Caso o tamanho do *Splitter* não comporte seu conteúdo, é possível exibir exibir barras de rolagem
- *splitterBarVisible* - Caso o usuário não possa redimensionar os painéis, a barra do *Splitter* pode ser definida como oculta
- *firstPaneContent* - Conteúdo do primeiro painel
- *secondPaneContent* - Conteúdo do segundo painel