BSP + jQuery Mobile, ou a aventura de SAP-webdev – parte I: Preparação e login

31/03/2015 10:00

FALAE SEUS BANDO DE CONSULTOR DEGENERADO! Tudo certo com vocês? Sim, eu estou de volta só agora em 2015 porque finalmente eu tenho algum conteúdo bacana pra colocar aqui. Porém antes de começar a historinha do *post*, um breve:

Aviso! | *¡Advertencia!* | *Warning!* | *Warnung!*

Esta série de *posts*, apesar de possuir conteúdo ABAP, está focada em **interface *web***, portanto o leitor deve esperar neste conteúdo referências a assuntos como:

- **HTML5**;
- **CSS**;
- **JavaScript** / **jQuery**;
- e **desenvolvimento *web*** em geral.

Introdução

Recentemente, eu fui envolvido num projeto onde o cliente quer modernizar a interface de algumas transações, que são executadas atualmente via SAP Console. Essas transações servem de suporte pra uma operação de coleta de materiais, e devido à essa natureza os operadores utilizam PDAs com Windows NT (sim, essas coisas ainda existem e estão sendo usadas) que acessam o SAP. A ideia do projeto é migrar as transações pra aplicações BSP e depois, aos poucos, substituir os dispositivos que os operadores usam por outros mais novos, com sistemas mais modernos (Android ou iOS).

"Mas por que usar BSP?", talvez seja a dúvida que surja neste momento. Talvez, porque ao parar pra pensar, BSP **é a única alternativa viável**. A única versão de WebDynpro que suporta integração móvel é a versão Java, e é de conhecimento comum que a performance de aplicações WebDynpro é muito inferior quando comparada à performance de outras tecnologias de interface *web* (esse *post* aqui, apesar da propaganda óbvia, ilustra o problema), o que é um ponto crucial quando decidimos o que usar numa aplicação móvel. Por esse mesmo motivo, riscamos as transações Easy Web da lista de candidatos. Não há a disponibilidade de usar Gateway, porque o ambiente não tem os componentes disponíveis e só a instalação e configuração dos mesmos daria um projeto à parte. Portanto, devido à flexibilidade necessária, nos resta apenas o BSP pra fazer o *backend*. O *framework* do BSP é praticamente o mesmo desde o R/3 4.7, o que facilita tanto o entendimento para a construção como as manutenções posteriores.

Com o *backend* decidido, o que usar de *frontend*? Como a aplicação vai rodar em navegadores / dispositivos mais novos, não faz sentido usar o ultrapassado HTMLB. Qual das trocentas alternativas de bibliotecas pra *frontend* é a melhor? Essa decisão, no meu caso, também já estava quase tomada – o cliente já tinha algumas aplicações BSP com interfaces implementadas em jQuery / jQuery Mobile em uso, portanto achei sensato usar essas mesmas bibliotecas para a aplicação nova. Seria interessante considerar o uso de UI5 aqui, porém eu achei que a curva de complexidade se tornaria muito íngreme para implementar uma lógica de interface que, nas aplicações que já existiam, era bem simples. E além disso eu não estou completamente à vontade ainda para desenvolver em UI5, então eu acabei pesando pro lado que eu conhecia melhor.

Resolvi, então, usar esse projeto como base para essa série de *posts*, onde eu pretendo explicar as técnicas que eu usei para desenvolver uma aplicação BSP com interface em jQuery Mobile. Observem bem que nessa última frase eu usei a palavra **técnicas**, porque eu não pretendo mostrar aqui a construção completa de uma aplicação – se você precisa de algo assim, existem alguns *posts* no SCN como esse aqui e esse aqui também ou quem sabe esse no Sourceforge onde ao final você terá uma aplicação construída.

Enfim, com as ferramentas escolhidas, já é possível começar a desenvolver.

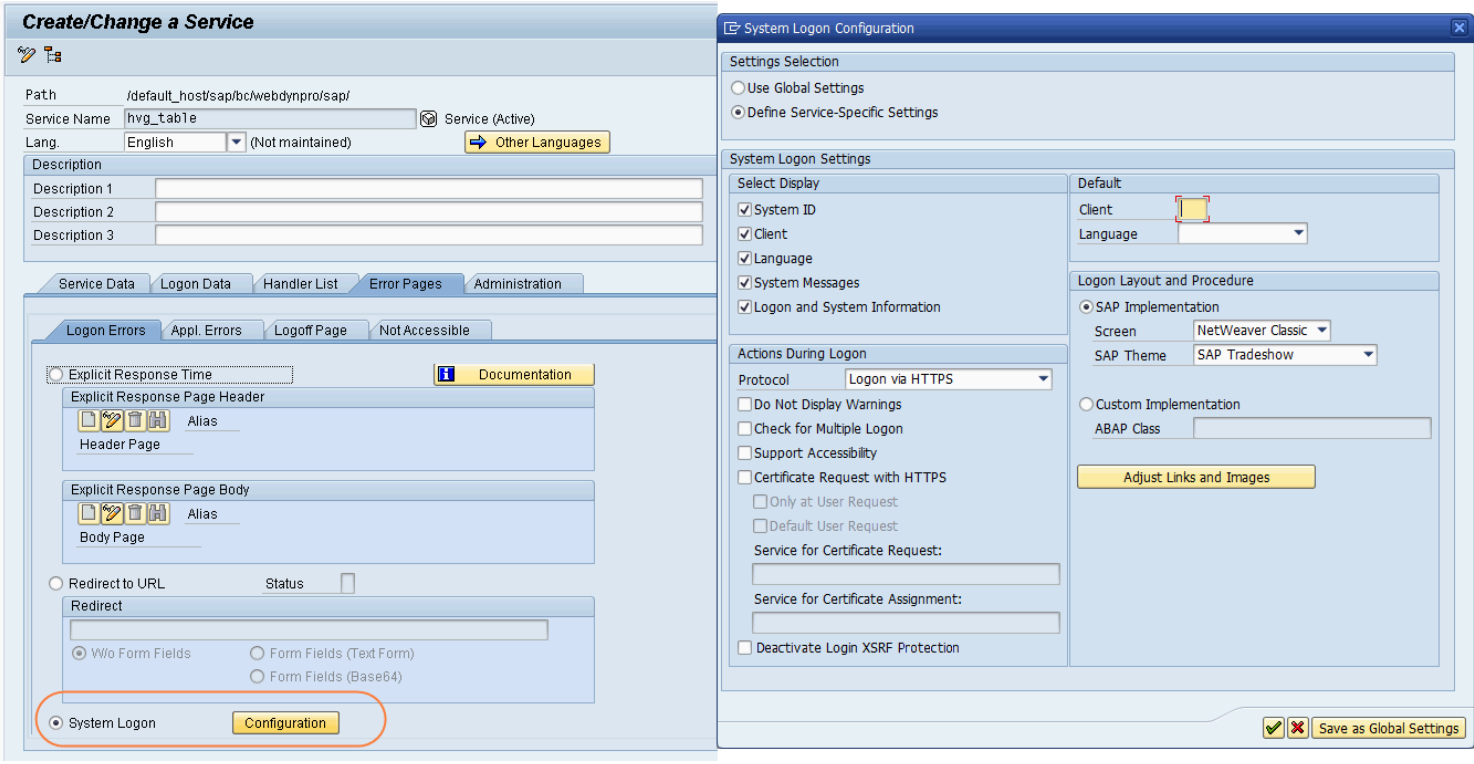
Preparando os recursos

Como estamos trabalhando com BSP, este é o momento onde deve-se importar os recursos necessários à aplicação para o repositório MIME. A documentação diz que os objetos MIME disponíveis para todas as aplicações BSP ficam na pasta**/SAP/BC/BSP/[namespace]/PUBLIC** (onde**[namespace]** pode ser tanto**SAP** como algum *namespace* do cliente), portanto esse é o local onde devemos importar as bibliotecas de jQuery (**UPDATE:** essa informação não está errada, mas leia o *edit* no fim desse *post*). No caso do projeto, as bibliotecas de jQuery e de jQuery Mobile já estavam presentes no repositório, assim como o tema jQuery Mobile personalizado do cliente. Eu acabei precisando importar apenas mais alguns *plugins* específicos para algumas tarefas, como o DateBox para criar campos de seleção de data mais consistentes do que os do jQuery Mobile, e o Number para formatação de campos com dados numéricos (vou falar mais desses *plugins* nos próximos *posts*).

Customizando o *login*

Até começar neste projeto, eu nunca havia desenvolvido muito em BSP fora de um ZHELLO_WORLD. Minha experiência com desenvolvimento *web* dentro do SAP abrangia somente WebDynpro ABAP, e em todos os projetos onde eu trabalhei com essa tecnologia até hoje o *deploy* das aplicações sempre foi feito dentro de um Portal (EP). Esse tipo de implantação facilita alguns aspectos do desenvolvimento. Um deles é a autenticação de usuários: na maioria das vezes, uma aplicação WDA implantada num EP não precisa se preocupar com *login*, *logoff* ou sessões de usuários, porque esses elementos são tratados pelo próprio portal.

Mesmo assim, em alguns momentos eu já precisei customizar o serviço ICF de algumas aplicações WDA. Só que com esse projeto eu acabei descobrindo que os serviços tem muito mais opções do que apenas "Ativar / Desativar".



Configurando páginas de erro na SICF

As telas acima mostram um exemplo de configuração de um serviço do ICF – mais especificamente da configuração de páginas de erro do serviço. Estas páginas se chamam assim porque são as respostas definidas no serviço a erros específicos que podem ocorrer durante o uso da aplicação: erros de *login* (**HTTP 401**), erros em geral da aplicação (**HTTP 500**), erros ao não encontrar algum recurso no servidor (**HTTP 404**) e páginas de resposta para *logoff*.

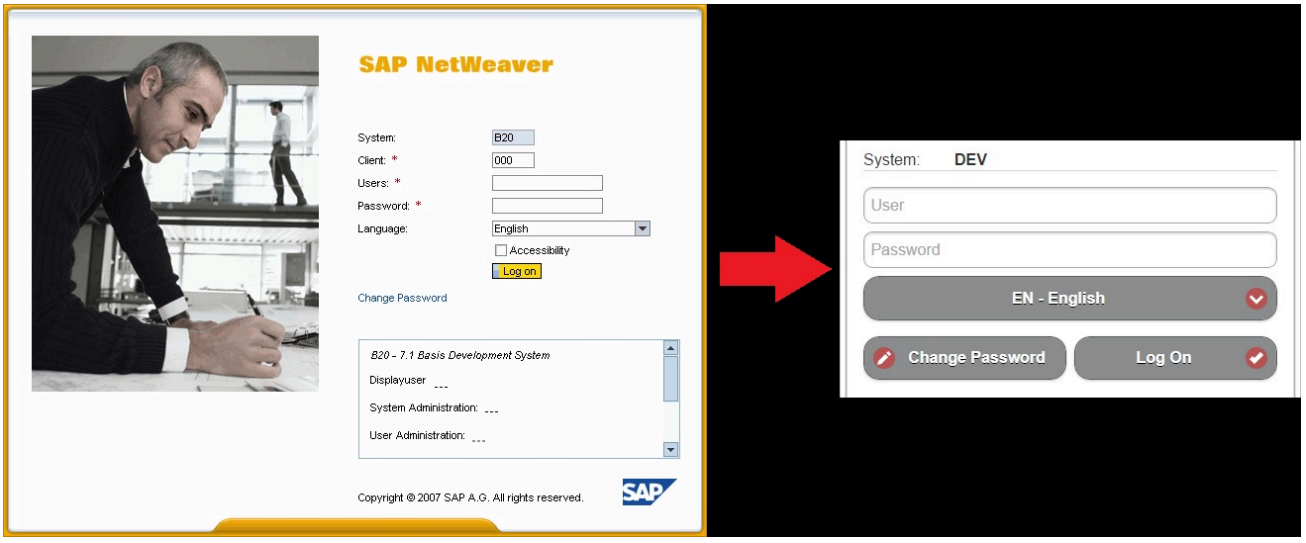
Todas essas páginas podem ser customizadas para um serviço de uma aplicação BSP, e é interessante que se faça isso. Porém, a tela de *login* talvez seja a mais importante de todas no caso de aplicações móveis porque, a não ser que se trate de algum caso mais raro onde dispositivos móveis fazem uso de Single Sign-On, a tela de *login* sempre será a primeira a ser exibida quando o usuário acessa o *link* da aplicação. Dessa maneira, é importante que a página de *login* seja tão responsiva quanto as páginas da aplicação.

O que temos a fazer aqui é criar uma classe filha da**CL_ICF_SYSTEM_LOGIN** e indicar essa classe dentro da configuração ***System Logon*** do serviço na SICF, através de ***Error Pages*** > ***Logon Errors*** > selecionar ***System Logon*** e clicar no botão ***Configuration*** > na tela nova, selecionar ***Define Service-Specific Settings*** > no grupo ***Logon Layout and Procedure***, selecionar ***Custom Implementation*** e indicar em ***ABAP Class*** o nome da classe gerada.

A**CL_ICF_SYSTEM_LOGIN** tem alguns métodos com o nome**HTM_***, que são responsáveis por criar o *markup* das páginas de *login* – os métodos que merecem destaque entre esses são o**HTM_LOGIN**, que é o responsável propriamente por criar a página de *login*, e o**HTM_CHANGE_PASSWD**, que constrói a tela para mudança de senha. A lógica implementada nessas páginas é bem simples: o usuário deve preencher seu ID e senha e clicar em um botão que disparará um evento específico através de uma função JavaScript pré-definida. Para isso, todos os métodos**HTM_*** possuem um parâmetro *importing* com o nome**IV_JAVASCRIPT**, que contém as funções predefinidas para os botões, além é claro de um parâmetro *returning* chamado**RV_HTML**, que é a *string* com o HTML gerado. No caso de uma tela construída com jQuery Mobile, podemos concatenar o conteúdo do parâmetro**IV_JAVASCRIPT** no *markup* logo após importar as bibliotecas jQuery. Os métodos também possuem um outro parâmetro *importing* chamado**IV_HIDDEN_FIELDS**, que traz um *markup* já construído, contendo os campos que NÃO estão selecionados no grupo ***Select Display*** na configuração (veja na imagem acima), ou seja, campos que devem ficar escondidos. O valor desse parâmetro pode ser inserido no início do *form* que será montado pelo método.

Além dos métodos, a classe também possui diversas **constantes** que definem os atributos**id** e**name** das *tags* de campos e botões. Esta página do *help* da SAP fornece uma referência para o uso dessas constantes. E podemos encontrar, ainda nessa mesma classe, uma série de outros **atributos** contendo textos que são construídos automaticamente para textos de *labels* e botões, de modo a facilitar a construção de *layout* – como, por exemplo, o atributo**M_TXT_LABEL_USER**, que serve de *label* para o campo de ID do usuário, ou o atributo**M_TXT_BUTTON_LOGIN**, que contém texto para o botão que realiza o *login*.

Felizmente, existe uma referência *standard* de como construir uma tela de *login* e também uma tela de mudança de senha. Essa referência é a classe**CL_ICF_EXAMPLE01_LOGIN**, que constrói as páginas em HTMLB. Partindo da implementação dessa classe, basta ter em conta os pontos onde as funções JavaScript são chamadas e reconstruir o *markup* usando os elementos do jQuery Mobile.



Enxugando a tela de logon com jQuery Mobile

Algumas dicas sobre a construção das telas de *login*:

- **Deixe a construção do HTML de retorno para um único comando no final do método.** Os métodos da**CL_ICF_EXAMPLE01_LOGIN** misturam diversos**CONCATENATE**s ao longo do código incluindo as *tags* HTML e construindo os textos das variáveis ao mesmo tempo, o que torna difícil de ver o resultado completo que será retornado. Em vez disso, é melhor construir todas as variáveis no começo do método e depois, como último comando, construir todo o HTML no parâmetro de retorno. Isso facilita a visualização do *markup*, de modo que você não precise ter que executar o método apenas para poder ver o retorno. BONUS POINTS pra quem fizer isso usando *string templates* em vez de**CONCATENATE** (se for possível, é claro).
- **Não esqueça das mensagens!** O atributo**M_LOGMESSAGES** é uma tabela que contém as mensagens relativas ao *login*; se algum erro for encontrado, a tela de *login* será chamada novamente e esse atributo voltará preenchido. É possível identificar o tipo de cada mensagem (erro, aviso, informação) comparando o valor do campo**M_LOGMESSAGES-SEVERITY** com as constantes definidas na classe**CL_BSP_MESSAGES**. A partir disso, cabe a você definir como exibir cada mensagem – eu particularmente gostei da implementação simples usando um *collapsible* que foi feita nesse *post*.
- **Desligue o Ajax nos *forms*.** O jQuery Mobile trata automaticamente os *submits* em *forms* usando Ajax, tentando carregar o resultado de um**POST** dentro de um novo** ** na mesma página. Não podemos deixar isso acontecer nas páginas de *login*; para isso, basta um**data-ajax="false"** na declaração de cada *form* e está tudo resolvido.
- **Limite os mandantes e idiomas.** Normalmente, você não quer que o usuário faça *login* em alemão no mandante 000 para usar a sua aplicação – portanto, limite os *inputs* de mandante (tabela**T000**) e de idioma (tabelas**T002** e**T002T**) usando *drop-downs*, dependendo do ambiente. Algumas funções relevantes para tratar os idiomas:**SYSTEM_INSTALLED_LANGUAGES**, que retorna os idiomas que estão instalados no sistema, e**RSRA_GET_DEFAULT_LANGUAGE**, que retorna o idioma padrão de *login* do sistema.

Assim fecho o primeiro *post* dessa série. No próximo, vou começar a falar da interface. Dúvidas, indagações ou sugestões? Deixem nos comentários aí abaixo ou mandem um e-mail.

Edit

Quero agradecer ao comentarista Max, que me apontou um problema de arquitetura dos recursos que eu tinha ignorado aqui no começo do guia.

A tela de *login* trabalha, obviamente, sem autorização do usuário. Portanto, se tentarmos acessar um recurso específico para aplicações BSP, como é o caso quando se usa o diretório/SAP/BC/BSP/[namespace]/PUBLIC, incorreremos no erroHTTP 401 – Unauthorized ao tentar acessar os recursos CSS e JavaScript. A solução (meio gambiarrenta, admito) é subir as bibliotecas de jQuery e jQuery Mobile TAMBÉM em um outro diretório que possa ser acessado sem autorização, como por exemplo o/SAP/PUBLIC/BC/UR, onde ficam as bilbiotecas de *Unified Rendering*, responsáveis pela tela de *login standard*. Se você não é muito fã de ter duas bibliotecas iguais em lugares diferentes no repositório, basta usar somente o diretório público (sem esquecer de usar as referências certas, claro).

Comentários

Wagner Duarte — *25/05/2015 18:29*

Leo ótimo Post! Como você citou, eu também costumo desenvolver em Webdynpro aplicações p/ web, e não conheço bem o BSP, nessa sua nova aventura, você deve ter aprendido bastante de BSP, tem alguns links interessantes sobre BSP ?
Também achei interessante a analise sobre UI5, essa dúvida é natural, pouca gente usa UI5 no Brasil para Web sem o contexto do Hana.

Leo Schmidt — *26/05/2015 11:22*

Valeu Wagner!

Sinceramente, eu tenho pouca documentação sobre BSP, e eu espero explicar bem como eu fiz uso dele no terceiro post dessa série (aliás o segundo sai hoje). Eu acho que a referência do help da SAP, pra mim, foi suficiente, porque o uso do framework de BSP que eu fiz foi mais trivial, já que não é por via dele que eu

> resolvi coisas como navegação, renderização, atualização de tela, etc., e sim diretamente pelo jQuery / jQuery Mobile.
>
> Eu gosto bastante da ideia do UI5 e eu ainda vou fazer alguns posts sobre isso, mas como eu disse ali é um framework que eu não conheço muito bem ainda. Apesar disso eu sei que dá pra fazer muita coisa com ele, e não necessariamente no HANA como vc disse, afinal é frontend.

> **Mauricio Cruz** — *27/05/2015 08:51*
>
> Acho bem legal usar o UI5 e etcs, mas a diversão toda dessa separação backend e frontend é que você pode usar o que você quiser. Naturalmente a SAP vai puxar o uso do UI5 por ser a biblioteca padrão do Fiori, mas até aí whatever, a gente usa o que a gente quiser \o/

Rodrigo F. Morais — *31/03/2015 14:20*

Grande Léo, ótimo post...estou navegando por esses mares de mobile...como não há mais suporte ao SAPConsole a mudança é necessária, deixo a dica do SAP ITS Mobile, tem bastante material na SCN, normalmente é usado para conversão das transações de WM, porém já tive várias experiencias com outras soluções para aplicações de browser ainda mais com sua dica de jQuery...segue um documento interessante com alguns exemplos...
ITS Asug

Abraços!

Thiago — *31/03/2015 13:26*

Ótimo post. Recentemente caí no mesmo dilema de ter que desenvolver uma aplicação web e. responsiva. Não manjava nada de MVC, JavaScript, etc... Acho que só assim para termos noção de que o mundo web não é só restrito ao “mundo dotnet/java”, e que nós de ABAP podemos sim fazer tão bem feito quanto eles.