Vývoj web aplikácii nad platformou SmartWeb

Platforma SmartWeb umožuje efektívny vývoj webových aplikácií, ktoré ako zdroj dát používajú aplikaný server D2000 resp. majú aplikanú logiku naprogramovanú v aplikanom serveri D2000. Vaka priamemu využívaniu štandardných webových technológií ako HTML, CSS a JavaScript nie je dizajn aplikácií limitovaný grafickými možnosami D2000. Znalos týchto technológií je preto nutnou požiadavkou pre vývoj aplikácií na platforme SmartWeb.

Úvod do vývoja aplikácií

Požiadavky na vývoj

- Nainštalovaný Git
- Vývojové prostredie pre HTML a JavaScript (napr. WebStorm)
- Webový prehliada (minimálne Internet Explorer 11, odporúaný Google Chrome)

Bundlovanie aplikácie

Pre aplikácie na platforme SmartWeb je možné používa konštrukcie jazyka JavaScript až do verzie ECMAScript 2017 a React komponenty používajúce JSX syntax bez ohadu na výsledný používaný webový prehliada. Toto je možné vaka tomu, že aplikácie vytvorené na platforme SmartWeb prechádzajú pri nasadení do produknej verzie procesom spracovania nazývaným bundlovanie. Tento proces má na starosti úpravu zdrojových kódov (transpilácia) tak, aby boli natívne interpretovatené najnižšou podporovanou verziou webového prehliadaa (prekladá sa do JavaScript-u verzie ES 5.0). Zárove poas procesu bundlovania dochádza k zmenšovaniu vekosti zdrojových kódov - minifikácia (odstránenie prázdnych znakov a komentárov, skrátenie názvov premenných a funkcií, ...) kvôli šetreniu prenášaných dát a tým rýchlejšiemu naítaniu webových stránok.

Konfigurácia aplikácie

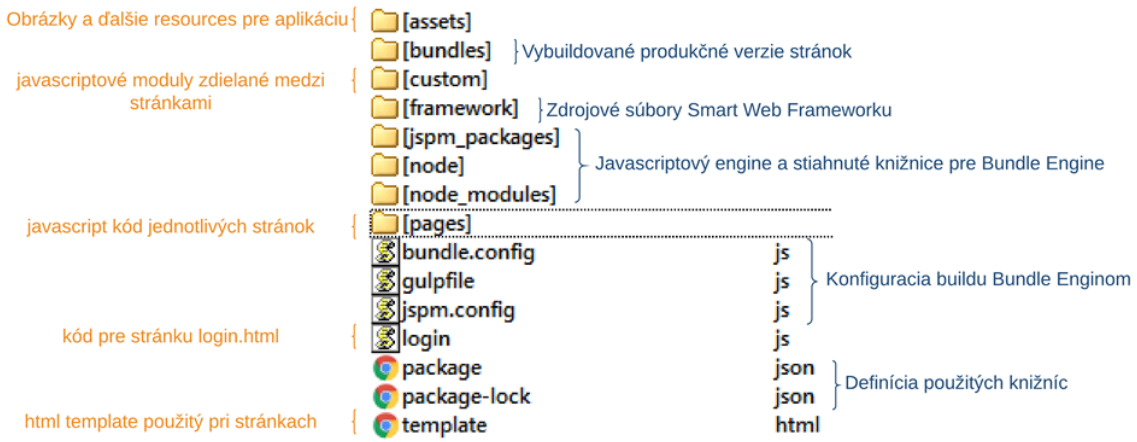
Na konfiguráciu SmartWeb aplikácie slúži súbor *smartweb.json*, v ktorom je možné povoova resp. zakazova jednotlivé súasti aplikácie, nastavi spôsob autentifikácie, obmedzi prístup k Event skriptom a RPC procedúram. Podrobný popis konfigurácie je možné nájs na stránke Konfigurácia SmartWeb aplikácie.

Štruktúra aplikácie

Nasledovný obrázok znázoruje štruktúru SmartWeb aplikácie.

Vavo sa nachádza popis súborov, ktoré vytvára programátor. Sú to najmä súbory jednotlivých stránok (adresár *pages*), stránka pre prihlasovanie užívatea (*login.js*) a HTML šablóna stránok (*template.html*). JavaScript moduly zdieané medzi stránkami sa nachádzajú v adresári *custom* a pre uloženie ostatných súborov ako napr. obrázky slúži adresár *assets*.

Vpravo sú popísané súbory generované resp. kopírované SmartWeb Bundle Enginom. Ide hlavne o vybuildované produkné verzie stránok, súbory SmartWeb frameworku a JavaScriptový engine so stiahnutými knižnicami pre bundle engine.



Externé knižnice

Vaka využitiu SystemJS a JSPM je možné v SmartWeb aplikácii využi akúkovek JavaScript knižnicu z GitHub-u a npm. Závislosti aj prípadné konflikty vo verziách budú automaticky vyriešené.

Vývoj aplikácií

SmartWeb aplikácie sa štandardne skladajú z viacerých webových stránok. Stránky môžu by napísané priamo ako HTML (prípona .html) súbory alebo ako JavaScript súbory (prípona .js), ktoré budú pomocou šablóny (*template.html*) namapované na HTML súbor (napr. stránka *index.html* vznikne kombináciou súboru šablóny *template.html* a súboru *index.js*). Súbor *template.html* predstavuje šablónu webovej stránky, pomocou ktorej je možné zadefinova opakované súasti výsledných stránok. Jediným povinným HTML elementom šablóny je element typu * * s id *"root"*. Obsah tohto elementu bude generovaný na základe konkrétneho JavaScript súboru, ktorý je mapovaný na výsledný HTML súbor. V šablóne je možné použi niekoko placeholder premenných definovaných v SmartWeb-e. Tieto budú pri naítaní stránky automaticky nahradené ich skutonou hodnotou.

- *{{{smartweb.global.context.path}}}* je url cesta (url path) ku koreovému adresáru SmartWeb servera, odvodená od názvu deploynutého"war" súboru aplikaným serverom Wildfly. (napr. /smartWeb).
- *{{{smartweb.application.root.path}}}* je url cesta (url path) ku koreovému adresáru web aplikácie, odvodená od toho ako je definovaná v konfiguranom súbore *smartweb.json.* (napr. /smartWeb/admin).
- *{{{smartweb.application.script}}}* obsahuje kód SmartWeb aplikácie
- *{{{smartweb.library.scripts}}}* obsahuje kódy použitých knižníc v SmartWeb aplikácii

Nasledovný kód predstavuje príklad súboru *template.html*, v ktorom sa definujú základné štýly, písma a ikona web stránky:

template.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>SmartWeb Demo</title>

    <link rel='shortcut icon' type='image/x-icon' href='{{{smartweb.application.root.path}}}/custom/img/favicon.
ico'/>
    <link href="{{{smartweb.application.root.path}}}/assets/fonts/font-awesome/font-awesome.css" rel="
stylesheet">
    <link href="{{{smartweb.application.root.path}}}/assets/fonts/font-open-sans/open-sans.css" rel="stylesheet"
>
    <style>
        .gray-bg {
            background-color: #f3f3f4;
        }
    </style>
    {{{smartweb.library.scripts}}}
</head>
<body class="fixed-nav top-navigation gray-bg">
    <div id="root"/>
    {{{smartweb.application.script}}}
</body>
</html>
```

Pre každý adresár pod koreovým adresárom SmartWeb aplikácie je možné definova vlastný súbor šablóny (vždy nazvaný *template.html*). Ak sa v adresári s JavaScript kódom stránky nenachádza súbor šablóny, tak sa automaticky hadá v adresárovej štruktúre vyššie, až kým sa šablóna nenájde.

Všetky stránky definované pod adresárom */pages* vyžadujú autentifikáciu. Ak užívate nie je autentifikovaný, je automaticky presmerovaný na stránku */login.html*, ktorá obsahuje prihlasovací formulár a realizuje autentifikáciu. Predvolená stránka, na ktorú je užívate presmerovaný po zadaní adresy SmartWeb aplikácie, je */pages/index.html*. Stránky definované mimo adresára */pages* nevyžadujú autentifikáciu.

Ladenie aplikácie

Pretože stránky pri ich nasadení do produknej verzie prebiehajú poas bundlovania procesom transpilácie a minifikácie, je ich vo výslednej forme veami ažké ladi. Na ladenie bol preto vytvorený špeciálny vývojový (ladiaci) režim, kedy je stránka do webového prehliadaa zaslaná vo svojej pôvodnej podobe a potrebné úpravy kódu ako transpilácia sa realizujú až následne priamo vo webovom prehliadai. Tento režim (ak je povolený) sa aktivuje klávesovou skratkou *CTRL+ALT+D* resp. doplnením parametra *?DEV* za názov naítavanej stránky. Naítanie stránky v ladiacom režime trvá dlhšie, pretože všetky zdojové kódy sú naítavané v ich pôvodnej podobe bez optimalizácie a navyše sú potom ešte v prehliadai pomocou JavaScriptu transformované pre zabezpeenie kompatibility s webovým prehliadaom.

Zmeny na stránke vidia len prihlásení užívatelia, ktorí k stránke pristupujú cez ladiaci režim. Práca ostatných užívateov s produknou verziou stránky/aplikácie nie je nijak ovplyvnená.

Rebundlovanie aplikácie

Po odladení zmien aplikácie pre ich nasadenie do produkcie nie je potrebné reštartova webový server. Ak je to v konfigurácii povolené, tak pomocou klávesovej skratky *CTRL+ALT+R* resp. doplnením parametra *?REBUNDLE* za názov stránky sa spustí rebundlovanie aplikácie na serveri.

Prístupy k vývoju webových aplikácií

Platforma SmartWeb umožuje vysokú flexibilitu z hadiska vývoja webových aplikácií. Programátor má na výber, ktoré rozhranie a ktoré súasti SmartWeb platformy na prístup k D2000 použije.

Vlastné UI (užívateské rozhranie) a REST API

V tomto prípade má programátor úplnú vonos z hadiska tvorby UI webovej aplikácie. Na komunikáciu s D2000 použije REST API, ktoré umožuje vola D2000 RPC a SBA procedúry a naítava dáta z archívov.

Nasledovný kód ukazuje príklad stránky bez použitia SmartWeb Frameworku, len s využitím D2000 REST API. Na zabránenie útoku typu cross-site request forgery, je potrebné do každého REST dotazu doplni CSRF token. Pomocou AJAX dotazu je cez REST API volaná RPC procedúra *Sum* na Evente *E.SmartWebTutorial*. Procedúra má tri parametre - prvé dva, vstupné, sú odovzdané hodnotou a tretí, návratový parameter typu *real*, bude vložený do atribútu s názvom *vysledok*. Podrobný popis serializácie dát je dostupný na stránke Serializácia dát medzi klientom a API rozhraniami.

example2.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>SmartWeb Demo</title>
    <script src="https://code.jquery.com/jquery-3.3.1.min.js" integrity="sha256-FgpCb/KJQlLNfOu91ta32o
/NMZxltwRo8QtmkMRdAu8=" crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/js-cookie@2/src/js.cookie.min.js"></script>
</head>
<body class="">
<div id="root">
    <h1>Stranka bez využitia SmartWeb Frameworku 2</h1>
    <div>
        <input id="inputA" type="text" value="10">
        <span style="background-color: rgb(238, 238, 238);"> + </span>
        <input id="inputB" type="text" value="20">
        <span>
            <button id="button1" type="button"> = </button>
        </span>
        <input id="inputC" type="text" class="form-control" disabled="" value="">
    </div>
</div>
<script language="JavaScript">

    function loadData(valueA, valueB) {
        // Vytvorí resp. naíta unikátny CSRF token na zabránenie cross-site request forgery
        var csrfToken = Cookies.get('CSRF-TOKEN');

        $.ajax({
            url: `/smartWeb/api/web/rest/v0/d2/rpc/E.SmartWebTutorial/Sum?CSRF-TOKEN=${csrfToken}`,
            type: 'POST',
            data: JSON.stringify([
                valueA, valueB, {type: 'real', returnAs: 'vysledok'}
            ]),
            async: false,
            cache: false,
            contentType: "application/json; charset=utf-8",
            dataType: "json",
            processData: false,
            success: function (data, status, jqXHR) {
                $('#inputC').val(data.vysledok.value);
            },
            error: function (jqXHR, status) {
                // error handler
                console.log(jqXHR);
                alert('fail' + status.code);
            }
        });
    }

    $(document).ready(function () {
        $('#button1').on('click', function () {
            var valueA = parseFloat($('#inputA').val());
            var valueB = parseFloat($('#inputB').val());
            loadData(valueA, valueB);
        });
    });
</script>
</body>
</html>
```

Vlastné UI a (Comet) D2Api

alšou možnosou pri vývoji webovej aplikácie je použitie rozhrania D2Api, ktoré je implementované pomocou knižnice CometD. Toto rozhranie, na rozdiel od REST rozhrania, umožuje aj tzv. server push komunikáciu, o znamená, že zo servera môžu by na klienta kedykovek odoslané dáta bez toho, aby si ich klient vopred vypýtal. Táto forma komunikácie je výhodná najmä v prípade, ke je žiadané na webovej stránke v reálnom ase publikova zmeny hodnôt v D2000.

React/SmartWeb komponenty a (Comet) D2Api

Tento prípad naplno využíva vlastnosti a možnosti SmartWeb Frameworku. SmartWeb Bundle Engine sa stará závislosti knižníc a správne usporiadanie modulov pred ich volaním, rieši spätnú kompatibilitu JavaScript kódov ich transpiláciou a optimalizuje vekos kódu. Volania D2Api sú zabalené v triede *Data Container*, ktorá sa stará o automatickú inicializáciu pripojenia cez D2Api. Navyše sú k dispozícii predpripravené React komponenty, podporujúce responzívny dizajn pomocou Bootstrap frameworku.

Nasledovný príklad renderuje stránku *objects.html*. Trieda *ObjectsPage* je React komponent definujúci vzhad a správanie stránky. *DataContainer* je špeciálny React komponent, ktorý realizuje pripojenie na D2000 a do stránky propaguje property *this.props.d2*, ktorá predstavuje inštanciu triedy *D2Api* slúžiacu na komunikáciu s D2000. Stránka využíva predpripravené komponenty *ValueComponent* na získavanie aktuálnych hodnôt D2000 objektov - v tomto prípade minúta (objekt *Min*) a sekunda (objekt *Sec*).

objects.js

```
import React from 'react'
import {DataContainer, PageHeader, ValueComponent} from '../custom/components'

DataContainer.renderComponent(class ObjectsPage extends React.Component {
    constructor() {
        super();
    }

    render() {
        return (
            <div className="container-fluid">
                <PageHeader {...this.props}/>
                <div className="wrapper wrapper-content page-heading">
                    <div className="row">
                        <div className="col-xs-8 text-right">
                            <span>Aktuálna minúta z D2000:</span>
                        </div>
                        <div className="col-xs-4 text-right">
                            <ValueComponent d2={this.props.d2}
                                            datasource="Min"
                                            numberFormat="0"
                                            missingFormattedValue="?"/>
                        </div>
                    </div>
                    <div className="row">
                        <div className="col-xs-8 text-right">
                            <span>Aktuálna sekunda z D2000:</span>
                        </div>
                        <div className="col-xs-4 text-right">
                            <ValueComponent d2={this.props.d2}
                                            datasource="Sec"
                                            numberFormat="0"
                                            missingFormattedValue="?"/>
                        </div>
                    </div>
                </div>
            </div>
        );
    }
});
```