FiDS : modèles FiDS

Edge Airport France

Table des matières

FiDS : modèles FiDS .....
Préambule .....
But .....
Principe .....
Définition .....
Pré-requis .....
Création d'un template .....
Principe .....
Premier template : veille .....
Créer la page html .....
Créer le CSS .....
Second template : veille messages .....
Debugger un template .....
Variables FiDS .....
Appel des variables .....
Liste des variables FIDS .....
Variables de vols .....
Variable "langages" .....
Edge Airport France .....

FiDS : modèles FiDS

Préambule

But

Permettre au service informatique des aéroports de créer ses propres modèles d'écran.

Principe

Prendre connaissance des technologies utilisées à l'affichage : HTML, CSS et JavaScript et créer un template exploitable par le serveur web.

Définition

Airport Manager FiDS exploite les standards de navigation web dont chacune des technologies a un rôle bien défini :

* **HyperText Markup Language** (HTML), langage de balisage qui établit la structure d'une page	`</head>`
	`<body>`
	`  <div id="page">`
	`  <img id="veille" src="veille.jpg" />`
	`  </div>`
	`</body>`
	`</html>`

* **Cascading Style Sheets** (CSS), langage de présentation qui permet d'appliquer un style à une page HTML	`#content #escales div{`
	`  width: 100%;`
	`  text-align: center;`
	`}`
	
	`#escales .escale {`
	`  width: 100%;`
	`  text-align: center;`
	`}`

* **JavaScript** (JS), langage de script qui permet de manipuler des informations ainsi que le contenu d'une page HTML	`function getElementsByClass( searchClass, domNode, tagName) {`
	`  if (domNode == null) domNode = document;`
	`  if (tagName == null) tagName = '*';`
	`  var el = new Array();`
	`  var tags = domNode.getElementsByTagName(tagName);`
	`  var tcl = " "+searchClass+" ";`
	`  for(i=0,j=0; i<tags.length; i++) {`
	`  var test = " " + tags[i].className + " ";`
	`  if (test.indexOf(tcl) != -1)`
	`  el[j++] = tags[i];`
	`  }`
	`  return el;`
	`}`

* **Smarty**, moteur de templates PHP qui permet d'ajouter des informations dynamiques dans des pages web tout en minimisant le code.	`{if count($flight->getSharedFlights(false)) > 0}`
	`  <div class="sharedFlights">`
	`  {foreach from=$flight->getSharedFlights(false) item=sharedFlight}`
	`  <div class="sharedFlight">`
	`  <img class="logo" src="{$sharedFlight.logo}" alt="{$sharedFlight.IATA}"/>`
	`  <span class="code">{$sharedFlight.number}</span>`
	`  </div>`
	`  {/foreach}`
	`  </div>`
	`{/if}`

Ces technologies permettent de créer un **template**, un patron de mise en page où l'on place images, textes et autres objets.

L'intérêt du template est d'y associer des **variables** qui seront lues par un **serveur web**. Ce dernier transforme ces variables en informations (texte, image…).

Le FiDS Display recevra donc un document généré du serveur web à partir du template, et à son tour traduira les pages **CSS** et **JS** liées au document.

Pré-requis

Airport Manager FiDS exploite un serveur web **Apache**. Le client ***FiDS Display*** embarque le navigateur web **Chromium**.

Le client se connecte depuis son adresse IP et envoie en paramètres le numéro d'écran ainsi que la base à questionner (Ex. : http://192.168.1.1/fids/?ecr=1&db=EAF).

Pour les tests, vous pouvez utiliser :

- **Google Chrome** pour afficher des templates.
- **Notepad++** pour éditer les templates.

Connaitre les bases des langages utilisés :

- CSS http://www.cheatography.com/davechild/cheat-sheets/css2/
- HTML http://www.cheatography.com/davechild/cheat-sheets/html4/
- JS http://www.cheatography.com/pyro19d/cheat-sheets/javascript/
- Jquery http://www.cheatography.com/i3quest/cheat-sheets/jquery/

- Smarty http://www.smarty.net/docsv2/fr/

Création d'un template

Principe

Avant de créer un template pour le FiDS, il est impératif de connaitre l'arborescence du système :

airlines

Contient les informations compagnies (logos, images...)

airport

Contient les informations par site (logos, images, css...)
s**ervices**

Contient des web services (météo, flux RSS...)

source

Contient des ressources nécessaires aux templates :

* css

* font

* img

* javascript

* jquery

* lib *

templates

Contient les templates, triés par mode

* 0 : veille

* 1 : infos bagages

* 2 : infos enregistrement

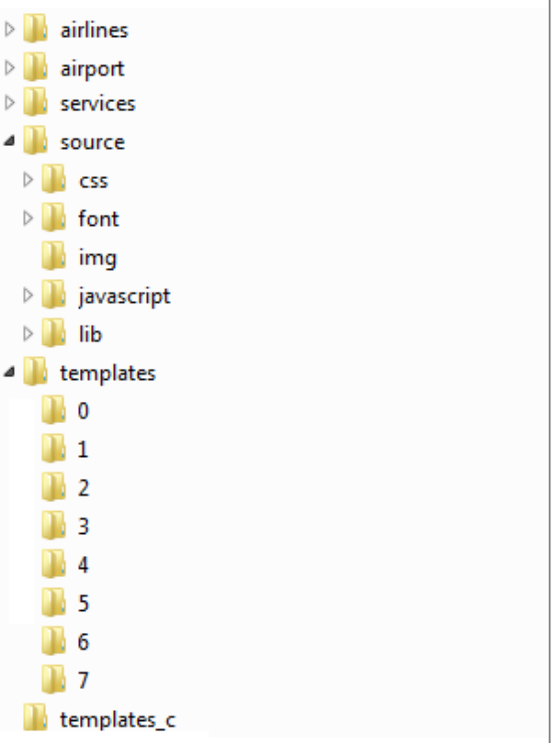
* 3 : infos embarquement

* 4 : liste arrivées

* 5 : liste departs

* 6 : sequences

* 7 : autres



- JS→ **/fids/source/javascript/**
- Jquery→ **/fids/source/javascript/jquery/**

Note :

Etant donné que les fichiers seront lus depuis un serveur web, la présence du **slash (/)** en début de chemin indique que le dossier est placé à la **racine** du serveur web.

En cas d'absence de ce caractère, le fichier sera recherché à partir du dossier où la page HTML est présente.

Premier template : veille

Créer la page html

Pour commencer, créer la page **/fids/templates/0/veille_test.htm** :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN////">
<html>
<head>
<title>Airport Manager - TEST</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<link type="text/css" href="/fids/source/css/veille_test.css"
rel="stylesheet" />
<script type="text/javascript"
src="/fids/source/javascript/jquery/jquery-1.4.4.min.js"></script>
<script type="text/javascript"
src="/fids/source/javascript/jquery/jquery.functions.js"></script>
<script type="text/javascript"
src="/fids/source/javascript/veille_test.js"></script>
</head>
<body>
<div id="header">Airport Manager vous souhaite un bon voyage.</div>
<div id="content"> <img src="/fids/airport/EAF/veille.jpg" /> </div>
<div id="footer"></div>
</body>
</html>
```

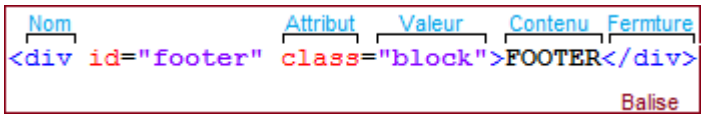
Il s'agit d'un langage de balises :

- **DOCTYPE** : permet de définir le type de document à afficher.
- **html :** balise qui contiendra le code. Aucun attribut n'est nécessaire.
- **head** : l'en-tête du document, partie invisible mais qui permet de :
  - Donner un titre au document.
  - Définir l'encodage des caractères.
  - lier une **feuille de style** (CSS).
  - lier une page **JavaScript**.
- **body** : le corps du document, partie visible par le client.

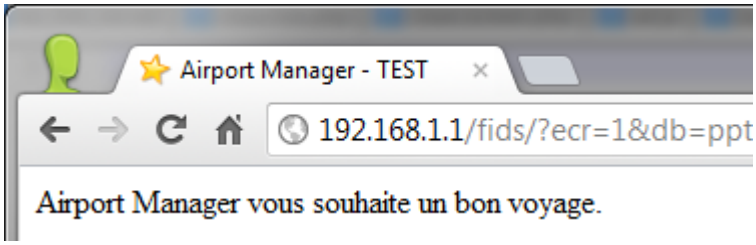
Les balises peuvent :

- avoir des attributs.
- encadrer un contenu (ex. : balises **<script></script>** et **<body></body>**).
- être auto-fermantes (ex. : balises **<meta/>** et **<link/>**).

Voici la structure basique d'une balise :



Ce document affichera un texte sur fond blanc :

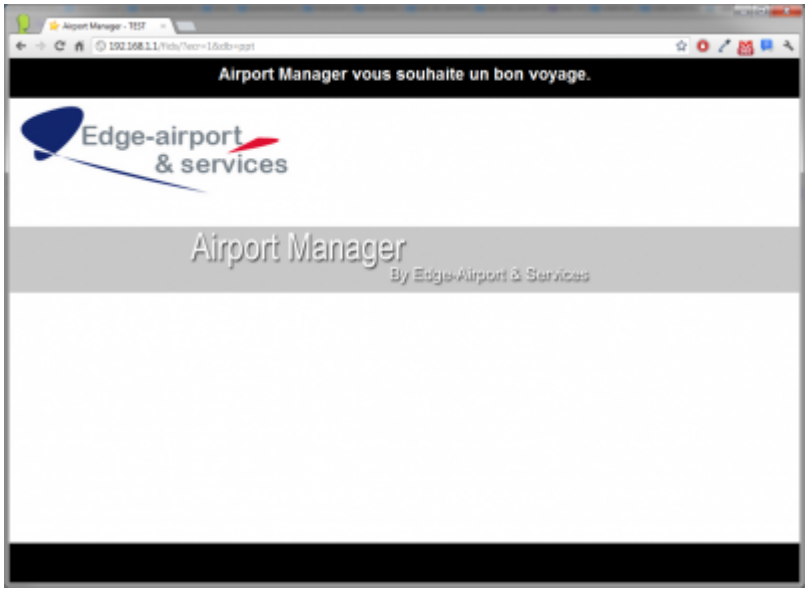


Créer le CSS

*Nous pouvons maintenant créer la feuille de style **/fids/source/css/veille_test.css** :*

```
//body
{
/* fond d'écran*/
background-color: black; /* fond noir */
/* police */
font-family: Arial; /* police "Arial" */
color: white; /* police blanche*/
text-align: center; /*centre le texte*/
font-weight: bold; /* police en gras*/
font-size: 25px; /* hauteur 25 pixels*/
overflow: hidden;
}
img
{
width: 100%;
}
```

Une fois que la feuille de style est définie, la page html affiche :



Une relation va automatiquement être créée entre la balise "**<body>**" dans le document HTML et la règle **body** de la feuille de style. Le navigateur va interpréter les **déclarations** du **sélecteur** concerné.

Voici la structure d'une règle CSS :

Note :

En CSS, il existe 4 types de sélecteurs :

- * (étoile), qui force toutes les balises à appliquer sa déclaration.
- Les sélecteurs précédés du caractère **#** (dièse) s'appliquent pour **une** balise ayant l'attribut **id** du même nom (ex. : le sélecteur **#header** s'appliquera pour la balise

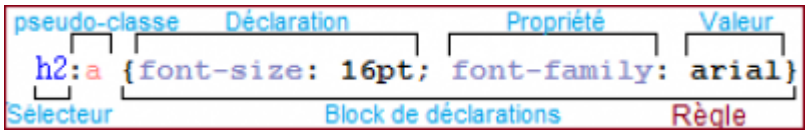
).

- Les sélecteurs précédés du caractère **.** (point) s'appliquent pour **toute** balise ayant l'attribut **class** du même nom (ex. : le selécteur**.section** s'appliquera pour les balises

).

- * Les autres règles s'appliqueront pour les balises ayant le même nom (ex. : le sélecteur **body** s'appliquera pour la balise <**body**></**body**>).

Il est possible de concaténer les sélecteurs afin d'affiner le style de balises filles.

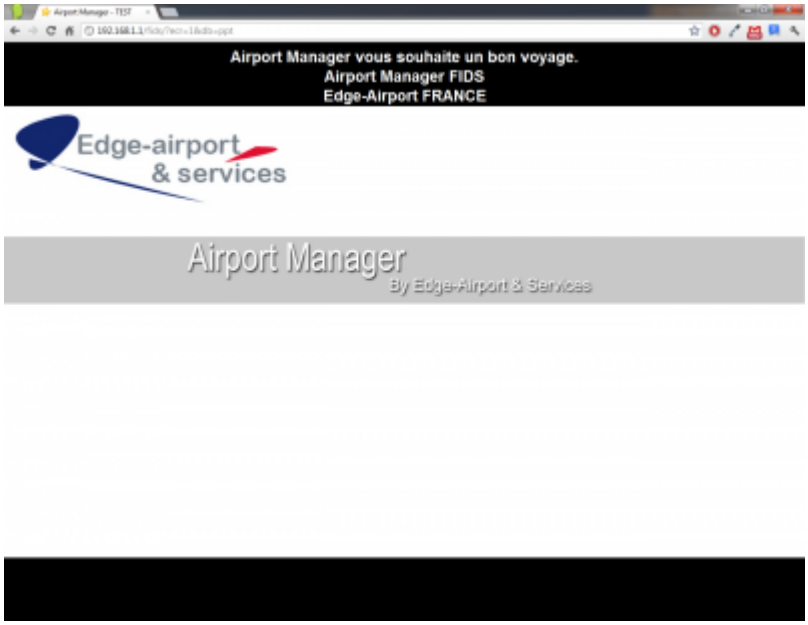


Second template : veille messages

Airport Manager FiDS met à disposition une série d'informations qui peuvent être affichées depuis les templates. Dans ce cas présent nous ferons défiler les variables **MSG1, MSG2** et **MSG3** fournies par le système.

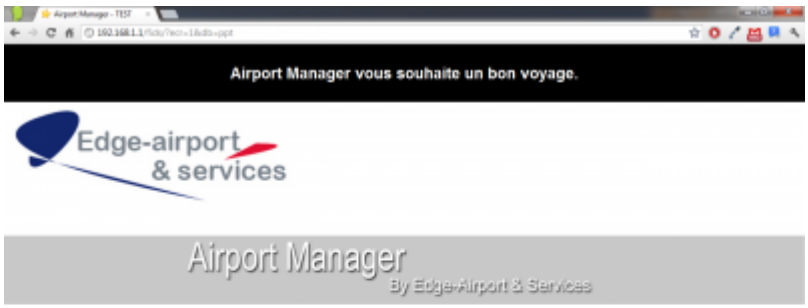
```
<!DOCTYPE HTML PUBLIC "-W3CDTD HTML 4.01 TransitionalEN">
<html>
<head>
<title>Airport Manager - TEST</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<link type="text/css" href="/fids/source/css/veille_test.css"
rel="stylesheet" />
<script type="text/javascript"
src="/fids/source/javascript/jquery/jquery-1.4.4.min.js"></script>
<script type="text/javascript"
src="/fids/source/javascript/jquery/jquery.functions.js"></script>
<script type="text/javascript"
src="/fids/source/javascript/veille_test.js"></script>
</head>
<body>
<div id="header">Airport Manager vous souhaite un bon voyage.</div>
<div id="content"></div>
<div id="footer">
**<div id="messages">
<div id="message1" class="messages">{$MSG1}</div>
<div id="message2" class="messages">{$MSG2}</div>
<div id="message3" class="messages">{$MSG3}</div>
</div>
</div>
</body>
</html>
```

Le corps de la page s'est enrichi d'une section "**messages**" composée de sous-section "**messages**" :



Sans modifier le CSS, les textes s'affichent en-dessous du premier texte car seul le corps de texte (la balise body) a été définie dans la feuille de style.

```
* {padding: 0;margin: 0;}
html, body{width: 100%;height: 100%;}
img {width: 100%;height: 100%;}
body {
background-color: black; ///* fond noir *//  /
///* police *//  /
font-family: Arial; ///* police "Arial" *//  /
color: white; ///* police blanche*//  /
text-align: center; ///*centre le texte*//  /
font-weight: bold; ///* police en gras*//  /
font-size: 25px; ///* hauteur 25 pixels*//  /
overflow: hidden;
}
#header {
width: 100%;
height: 92px;
line-height: 92px;
}
#content {
width: 100%;
height: 738px;
}
#footer {
width: 100%;
height: 92px;
}
#footer #messages {
width: 100%;
text-align: center;
}
#footer #messages .message {
width: 100%;
text-align: center;
}
```





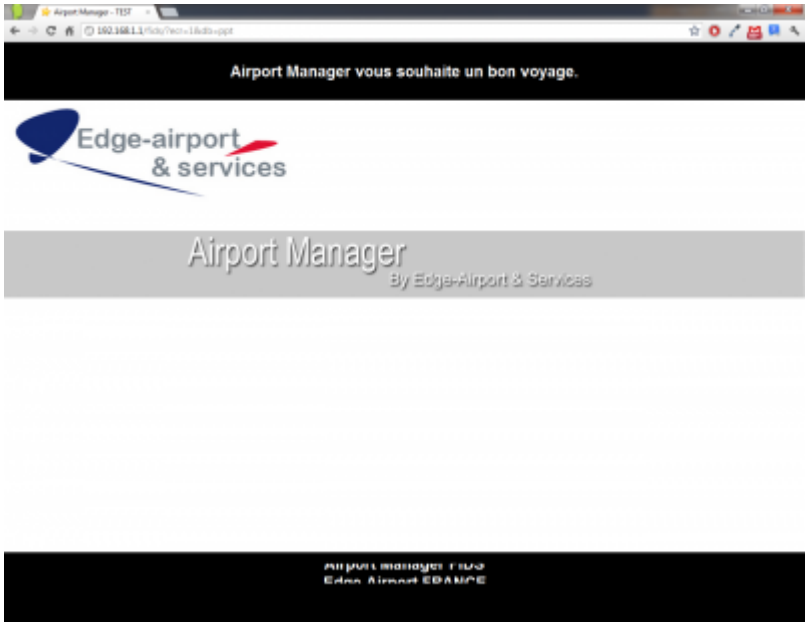
Les balises appliquent leur règle CSS et les messages sont bien en bas de page, mais ne défilent pas encore. Cette partie sera effectuée par le **javascript**, plus précisément par une librairie nommée **jQuery**.

jQuery permet d'écrire du javascript en utilisant le système de sélecteurs en CSS, il est donc plus facile d'appliquer des animations sur une balise. Cette librairie est extensible et permet d'intégrer des fonctions personnalisées.

Un appel à cette librairie aura été fait au préalable (ligne 7) ainsi qu'à des fonctions nécessaires (ligne 8). Nous allons donc pouvoir écrire le fichier **veille_test.js :**

```
$().ready(function() { ///* attend que le document soit prêt*//  /

///*initialise les repertoires*//  /

js_path = '/fids/source/javascript/';

jquery_path = js_path+'jquery/';

///* importe l'extension de défilement*//  /

$.includeJS(jquery_path+'jquery.cycle.all.min.js');

///* lance la fonction de défilement sur la balise id="messages"*//  /

$('#messages').cycle({

fx: 'scrollUp',

speed: 2000

});

});
```

La présence de jQuery se reconnait dans l'utilisation de la syntaxe **$(selecteur).uneFonction()**. Ici, ces quelques lignes permettent, lorsque le navigateur a fini de charger la page, de lancer un défilement (cycle) sur les messages avec un effet de déroulement qui prend 2 secondes par page.

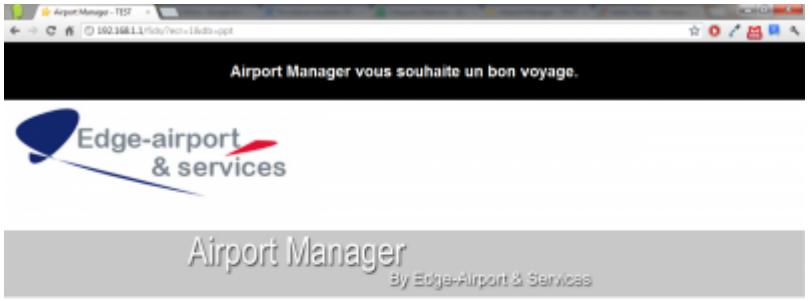


Le résultat final, pris lors du déroulement entre les messages 1 et 2

Debugger un template

Google Chrome intègre un mode debugger dans le navigateur. Il permet de s'assurer de la gestion des ressources (mauvais chemin d'un fichier, page introuvable…), du bon fonctionnement des scripts JS et de la bonne interprétation des règles CSS.

Il est disponible en pressant la touche **F12** ou dans la **paramètres > Outils > Outils de développement**. Une fenêtre apparaitra sous la page affichée





La documentation de cette fenêtre est disponible à cette adresse :

https://developers.google.com/chrome-developer-tools/docs/overview?hl=fr#window

Les onglets Elements et Console permettront :

- de parcourir le code HTML du template généré par le serveur.
- de parcourir les règles CSS pour chaque balise.
- de parcourir les erreurs retournées par Javascript.

Variables FiDS

Un template peut contenir des variables afin d'en personnaliser l'affichage.

Appel des variables

Les variables sont interprétées côté serveur. Pour les intégrer, une syntaxe bien précise doit être respectée dans le document HTML.

```
<div>** {$uneVariable**} </div>
```

Les accolades impliquent que tout ce qui sera à l'intérieur va être interprété côté serveur. Le caractère **$** (dollar) précède toutes les variables.

Les accolades peuvent aussi contenir des fonctions natives afin d'effectuer un traitement plus profond des variables comme des conditions ou des boucles.

{**if** $flight→countEscales() > 0} */*si le vol contient une escale** /

{**foreach** from=$flight→getEscales() item=escale name=ligne} */* pour chaque escale que contient le vol** /

/ {$escale→getAirportName()} */* affiche l'aéroport escale** /

{/**foreach**}

{/**if**}

Liste des variables FIDS

VARIABLE	DESCRIPTION	UTILISATION
Tous Modes		
MSG1	Messages personnalisés	$MSG1
MSG2	$MSG2	
MSG3	$MSG3	
VIDEO	Chemin d'une vidéo	$VIDEO
SOC	Code Aéroport	$SOC
CLASS	Classe de l'écran	$CLASS
ECRFILE	Fichier lié à l'écran (image, séquence…)	$ECRFILE
RSS	url d'un flux RSS	$RSS
TerminalTitle	Nom du terminal où se situe l'écran	$TerminalTitle

languages	Tableau qui contient les termes les plus courant en aeroportuaire dans les langues définies par l'administrateur	
Check-In, Boarding et Bagage		
flights	Liste des vols selectionnés	$flights
flight1	Vols sélectionnés	$flight1
flight2	$flight2	
flight3	$flight3	
airline_class	Classe de la compagnie	$airline_class
IMAGEBG	Image de fond de la compagnie	$IMAGEBG
Liste Arrivée/Départ		
flights	Liste des vols du jour	$flights
nblines	Nombre de lignes autorisées à afficher par l'écran	$nblines

Variables de vols

Les variables de vols ont une utilisation différente des autres variables. Elles sont composées de variables imbriquées mais aussi de **méthodes** à appeler pour récupérer une information spécifique d'un vol. Voici la liste des méthodes :

METHODE	DESCRIPTION	UTILISATION
getId()	Identifiant abstrait du vol	$flight→getId()
getCLI()	Code compagnie du vol	$flight→getCLI()
getLocalPlannedDate()	Date locale du vol	$flight→getLocalPlannedDate()
getLocalPlannedTime()	Heure locale du vol	$flight→getLocalPlannedTime()
getLastCheckInTime()	Heure limite d'enregistrement	$flight→getLastCheckInTime()
getLogo()	Chemin du logo de la compagnie	$flight→getLogo()
getNum()	Numéro de vol	$flight→getNum()
getSharedFlights()	Tableau des vols en "codeShare". Cette méthode accepte un paramètre qui définit si le vol principal est intégré dans le tableau. Si aucun paramètre n'est entré, le système interprètera que ce paramètre est à **true.**	$flight→getSharedFlights(true) Ou $flight→getSharedFlights(false)
getAirportName()	Nom de l'aéroport de destination ou en provenance	$flight→getAirportName()
getAirportNames()	Liste du nom de l'aéroport de destination ou en provenance dans les langues définies par l'administrateur.	$flight1→getAirportNames()
countEscales()	Nombre d'escales	$flight→countEscales()
getEscales()	Tableau d'escales	$flight→getEscales()
getGate()	Code porte	$flight→getGate()
getInfos()	Tableau des informations de vol	$flight→getInfos()

Variable "langages"

La variable langages est une matrice à deux dimensions :

- La première dimension stocke la langue (français, anglais, espagnol, arabe…)

- La deuxième dimension stocke un mot ou une expression dans la langue définie au-dessus.

Code pour afficher une expression dans toutes les langues disponibles :

```
{foreach from=$languages item=language} ///*pour chaque langue*//  /

<p>{$language.time}</p> ///*afficher l'expression "time"*//  /

{/foreach}
```

La liste des mots qu'elle contient est extensible. Chaque aéroport peut ajouter des expressions qui seront nécessaires au système de télé-affichage. Voici une liste non exhaustive :

- arrival
- domesticArrival
- internationalArrival
- nationalArrival
- businessArrival
- cargoArrival
- departure
- domesticDeparture
- internationalDeparture
- nationalDeparture
- businessDeparture
- cargoDeparture
- date
- time
- flight
- flightNumber
- destination
- baggage
- baggageClaim
- gate
- status
- origin
- airport
- stopover
- via
- boarding
- checkIn

From:
https://edgeairport.alwaysdata.net/wiki/ - **Documentation Embross (ex Edge Airport)**

Permanent link:
https://edgeairport.alwaysdata.net/wiki/doku.php?id=guides:utilisation:fidspage:modelefids&rev=1495805753

Last update: **26/05/2017 15:35**

Edge Airport France

Airport Manager Solutions

Phone: +33 553 801 366

Service commercial : contact@edge-airport.com

Support technique : support@edge-airport.com

Edge Airport France SAS au capital de 150 000 €

RCS Bergerac 529 125 346 Les Lèches TVA : FR53529125346 / EORI : FR52912534600039

Tel : +33(0)553 801 366 contact@edge-airport.com www.edge-airport.com