Documentation de la classe `tb`[1]

9 avril 2008

Fabien Dagnat[2]

Cette classe a été écrite pour uniformiser et faciliter la production de documents pédagogiques. Elle permet de produire différentes formes de documents : des cours (transparents), des sujets de travaux pratiques, des fiches programmes, ...

1 Modes et variables

Son principe de fonctionnement repose sur deux notions :

1. les modes,
2. les variables.

1.1 Modes

Un *mode* est une option de la classe. La définition d'un mode (macro `\tb@DEFINE@MODE`) `toto` provoque la définition d'un booléen `tb@mode@toto` initialement positionné à faux. La liste des modes possibles est mémorisée (macro `\tb@MODE@LIST`) et chaque mode définit une option pour la classe. Si cette option est présente, le booléen devient vrai. Pour chaque mode, un environnement de même nom est défini, si le mode est actif son contenu est traité sinon il n'est pas exécuté.

Plusieurs utilisations des modes sont faites dans la classe :
– Choix du type du document
– Définition des parties du document à inclure
– Positionnement par défaut de valeurs (les variables)
La liste des modes définis par défaut est :
– `correction` : permet de définir deux sous document à l'intérieur du document, un pour le sujet, un pour la correction (voir section 3).

[1] valable pour la version 0.9 du 23/07/2007
[2] `fabien.dagnat@telecom-bretagne.eu`

– `mastere`, `majeure`, `mineure`, `fip`, `master`, `fc`, `option` : définissent les différentes formations à l'école en positionnant la valeur de certaines variables (voir figure 1.2).
– `cours`, `devoir`, `ficheprog` : les grands types de documents (voir section 2).
– `gl`, `po`, `conc`, `base`, `avance` : définissent des matières au sein des formations.

1.2 Variables

Une variable `titi` est une valeur à laquelle sont associées des macros :
– `\tb@var@titi` : la valeur (initialement « titi to define »),
– `\titi` : une macro prenant un argument permettant de mettre à jour la valeur (macro précédente),
– `\debugtiti` : qui permet d'afficher le contenu actuel de la variable.
Les variables sont définies par la macro `\tb@DEFINE` et mémorisée dans une liste qui est stockée dans la macro `\tb@VAR@LIST`. La commande `\debug` permet d'appeler la macro d'affichage (`\debugNOM`) sur toutes les variables.

La liste des variables définies par défaut est :
– `type` : le nom de la formation,
– `matiere` : le nom de la matière,
– `code` : le code du module (par exemple INF302)
– `numero` : le numéro du document (par exemple TP3)
– `titre` : le titre du document
– `titrecourt` : une version courte du titre (pour les entêtes)
– `soustitre` : le sous titre
– `soustitrecourt` : une version courte du sous titre (pour les entêtes)
– `version` : la version du document
Chaque document, selon ses options, positionne par défaut la valeur de certaines de ces variables. Tout d'abord, le `numero` doit toujours être fournis excepté pour les sujets de devoir (`devoir`) et les fiches programmes (`ficheprog`) pour lesquels la variable vaut respectivement *Examen* et *Fiche Programme*. Les autres valeurs par défaut sont décrites dans la figure 1.2.

Le positionnement spécifique de ces variables se fait évidemment à travers la macro qui porte leur nom. Pour la gestion des titre et sous titre, les macros usuelles (`title` et `subtitle`) de LaTeXsont redéfinies pour travailler avec les variables correspondantes. Elles prendront en paramètre optionnel la version courte du titre : `\titre[un titre court]{un titre long}`. Si le premier argument est absent la valeur du titre court est égale à celle du titre long.

1.3 Booléens

En plus des modes, les booléens suivant sont définis[3] :
– `tb@boolean@inCorrection` : variable interne pour contrôler la forme d'affichage des corrections (**ne pas y toucher !**)
– `tb@boolean@handout` : indique si le document est destiné à l'impression (se positionne par l'option `handout`)
– `tb@boolean@withAuthors` : contrôle l'apparition des auteurs sur la page de titre ; par défaut à vrai pour les cours, les fiches programmes et les documents pour le master (se manipule par les macros `\withAuthors` et `\withoutAuthors`)

[3] Ils doivent être uniquement modifiés en entête du document.

option(s)	`type`	`matiere`	`code`
mastere	Mastère	Base en Mod. et Prog.	ISIC302
master	MR2A Informatique	Module MMPFPSD	{}
option	3e année, Filière SLR	Bloc IDL	{}
fc	Formation Continue		{}
fc & base	Formation Continue	Introduction à Java	{}
fc & avance	Formation Continue	Java Avancé	{}
fip	FIP	Programmation Objet	INF103
majeure	Majeure Informatique		
majeure & conc	Majeure Informatique	Concurrence et Distribution	INF411
majeure & po	Majeure Informatique	Programmation Objet	INF303
majeure & gl	Majeure Informatique	Génie Logiciel	INF302
majeure & po & gl	Majeure Informatique	Génie Logiciel & Programmation Objet	INF302/303
mineure	Mineure Informatique		
mineure & po	Mineure Informatique	Programmation Objet	INF203
mineure & gl	Mineure Informatique	Génie Logiciel	INF202
mineure & po & gl	Mineure Informatique	Génie Logiciel & Programmation Objet	INF202/203

FIG. 1 – Le positionnement par défaut des variables selon les options.

- `tb@boolean@withDate` : contrôle l'apparition de la date sur la page de titre ; par défaut à vrai uniquement pour les devoirs (se manipule par les macros `\withDate` et `\withoutDate`)
- `tb@boolean@withCorrection` : permet de forcer l'apparition ou la disparition de correction[4] (se manipule par les macros `\withCorrection` et `\withoutCorrection`)
- `tb@boolean@withSection` : dans les cours, ajoute au début de chaque section un transparent d'avancement contenant la liste des sections avec la section courante mise en valeur (se manipule par les macros `\withSection` et `\withoutSection`)
- `tb@boolean@withIdentity` : permet d'ajouter un cartouche d'identification demandant les noms et prénoms ; par défaut à vrai pour les devoirs sauf ceux de master (se manipule par les macros `\withIdentity` et `\withoutIdentity`)

2 Types de documents

La classe permet de créer des documents de deux types :
- des transparents (en utilisant la classe `beamer`) si l'option cours est spécifiée,
- des sujets (en utilisant la classe article) sinon.

Elle est conçue pour produire des documents en français ou en anglais. Par défaut la langue est le français et une option `english` permet de basculer en anglais. Pour avoir un document dans les deux langues, il existe une option `francais` à utiliser en complément de l'option `english`. La langue par défaut correspond à l'option positionnée en deuxième. Dans le texte, il est alors possible de changer de langue (voir la documentation de `babel`).

Pour les langues, elle utilise les paquetages `babel` (option `francais` et `english`), `inputenc` (option `latin9`) et `fontenc` (option `T1`). De plus, pour utiliser des fontes modernes permettant de produire des documents postscripts aussi bien que des pdf, le document est composé avec les fontes du paquetage `lmodern`.

[4] La différence par rapport à l'option correction est que cela permet d'avoir le contenu des environnements sujet et correction.

Un document peut contenir une option `handout` qui spécifie que le document est destiné à l'impression, le comportement est alors spécifique à chaque type de document et sera décrit dans les sous sections les concernant.

2.1 Cours

Ce type de documents repose sur la classe beamer, seules certaines options de mises en page sont fixées. L'exemple de document minimal ci-dessous produit le document présenté figure 2.

```
\documentclass[cours]{tb}
  \title{Le titre}
  \subtitle{Le sous titre}
  \author{L'auteur}
  \date{la date}
\begin{document}
  \begin{frame}[plain]
    \titlepage
  \end{frame}
  \begin{frame}{Plan}
    \tableofcontents
  \end{frame}
\section{Une première section}
  \begin{frame}{Un transparent}
  \end{frame}
\section{Une autre}
  \begin{frame}{Un deuxième transparent}
  \end{frame}
\end{document}
```

Remarque :

Attention, par défaut, le texte en dehors des environnement `frame` est ignoré y compris la définition de macro !

Le document essaye de suivre la charte graphique de l'école (cette partie a été développée à partir du travail de Christophe Lohr).

Il est conseiller de parcourir la documentation de `beamer` pour connaître ses possibilités.

Dans la version imprimable, lorsque l'on utilise l'option `handout`, liens du document ne sont pas ajoutés et le document est généré en quatre transparents par page.

2.2 Autres documents

Par défaut, les options a4 et 12pt sont positionnées[5]. Des exemples de documents simples sont donnés en annexe page 12.

[5] Ce n'est pas encore (?) modifiable.

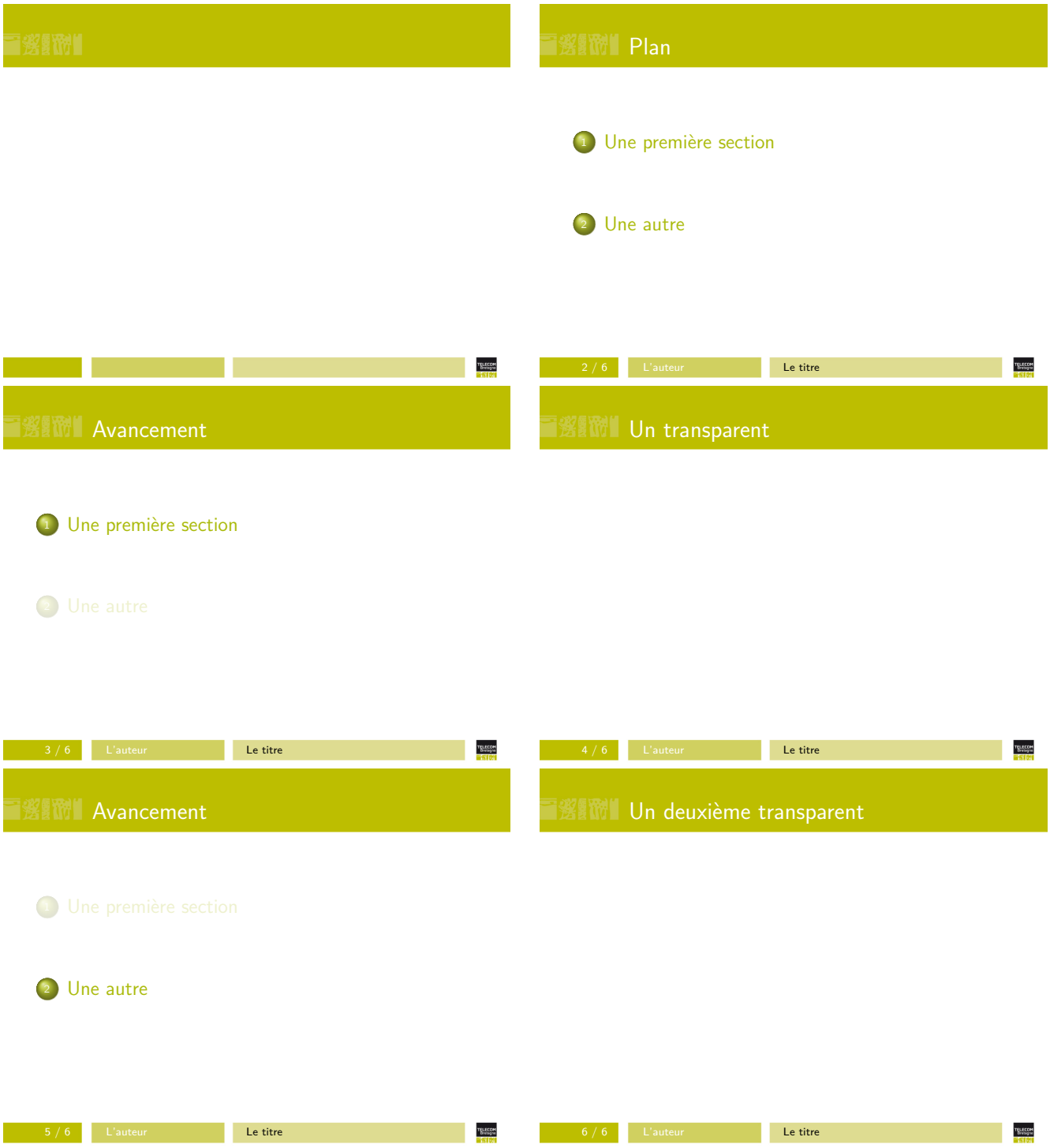


FIG. 2 – Un document de cours minimal

2.2.1 Basique

Les auteurs La macro usuelle `\author` peut être utilisée dans tous les documents. L'apparition de son contenu dans les pages de titre est contrôlée par l'état du booléen présenté en section 1.3 : `\tb@boolean@withAuthors`. Son affichage a lieu par défaut uniquement pour les cours et les fiches programmes.

Entête et pied de pages Les documents ont des entêtes et des pieds de pages imposés par défaut :
– en haut à gauche : (code –) MATIÈRE,
– en haut à droite : (numéro) (– **titre court**),
– en bas à gauche : TELECOM Bretagne (– type),
– en bas au centre : *le numéro de page*,
– en bas à droite : la date

La page de titre Pour les documents de type texte, la page de titre est produite à partir de la macro usuelle de LaTeX, `\maketitle`. Son utilisation provoque l'affichage de :

(numero –) titre

(sous-titre)

(date)

(AUTEURS)

(Correction)

Les informations entre parenthèses n'apparaissent que si elles ne sont pas vides. L'affichage des auteurs et de la date sont contrôlés par les booléens présentés en section 1.3. Enfin, correction n'apparaît que si le document est en mode correction (voir section 3).

Dans le mode `devoir` apparaît, en plus, un cartouche pour que les élèves puissent s'identifier :

2.2.2 Devoir

Pour les devoirs, les choix suivants sont faits par défaut :
– le numéro vaut : Contrôle,

– le titre vaut : type(, code) (le code n'apparaît que s'il n'est pas vide),
– le sous titre vaut : matière,
– la date est affichée,
– le cartouche est affiché (sauf pour le master),
– les entêtes et pieds de pages suivants sont modifiés :
  – en haut à droite : le numéro,
  – en bas au centre : *le numéro de page* sur *le nombre total de page*,

2.2.3 Fiche Programme

Pour les fiches programmes, le titre et le sous titre sont définis par défaut. Ils valent :
– le numéro vaut : Fiche Programme,
– le titre vaut : type
– le sous titre vaut : matière,
– les auteurs sont affichés,
– les entêtes et pieds de pages suivants sont modifiés :
  – en haut à droite : le numéro,

3 La gestion de corrections

Une option correction existe pour tout type de document. Cette option permet gérer un document contenant deux types d'information : un sujet (pour les élèves) et une correction (pour les enseignants et éventuellement les élèves).

Dans le paquetage sont définis deux environnements : `sujet` et `correction`. Par défaut (pas d'option correction), les environnements `correction` sont ignorés et les sujets traités.

Si l'option correction est choisie :
– Le mot Correction apparaît en rouge dans le titre,
– Le contenu des environnements `sujet` sont ignorés,
– Le contenu des environnements `correction` sont traités puis affiché dans un cadre rouge et avec un fond légèrement rougi.

Par exemple, le code suivant :

```
1  \documentclass{tb}
2    \author{L'auteur}
3    \date{la date}
4  \begin{document}
5    \maketitle
6
7    \begin{sujet}
8      Quand c'est le sujet comme ici, ce texte apparaît.
9    \end{sujet}
10
11   La classe \java{Stock} gère un ensemble de produits. La taille
12   maximale d'un stock est fixée à sa création. Le contenu de ce stock
13   est rangé dans un tableau.
14
15   Dans un premier temps, les méthodes ne gèrent pas les erreurs
```

```
16   d'ajout d'un produit dans un stock plein ou de retrait d'un produit
17   d'un stock vide.
18
19   Vous devez réaliser la classe \java{Stock} décrite ci-dessous \ldots
20
21 \begin{correction}
22   Et voilà la correction :
23
24   Bien insister sur l'association UML qui est une relation
25   \emph{structurelle} (qui « dure ») et qui se traduit en Java par des
26   attributs.
27
28   \inputJava{Stock.java}
29 \end{correction}
30
31 \end{document}
```

produit le document suivant :

numero to define – titre to define

Quand c'est le sujet comme ici, ce texte apparaît.

La classe `Stock` gère un ensemble de produits. La taille maximale d'un stock est fixée à sa création. Le contenu de ce stock est rangé dans un tableau.

Dans un premier temps, les méthodes ne gèrent pas les erreurs d'ajout d'un produit dans un stock plein ou de retrait d'un produit d'un stock vide.

Vous devez réaliser la classe `Stock` décrite ci-dessous ...

1

et si on ajoute l'option correction :

numero to define – titre to define

Correction

La classe `Stock` gère un ensemble de produits. La taille maximale d'un stock est fixée à sa création. Le contenu de ce stock est rangé dans un tableau.

Dans un premier temps, les méthodes ne gèrent pas les erreurs d'ajout d'un produit dans un stock plein ou de retrait d'un produit d'un stock vide.

Vous devez réaliser la classe `Stock` décrite ci-dessous ...

Et voilà la correction :
Bien insister sur l'association UML qui est une relation *structurelle* (qui « dure ») et qui se traduit en Java par des attributs.

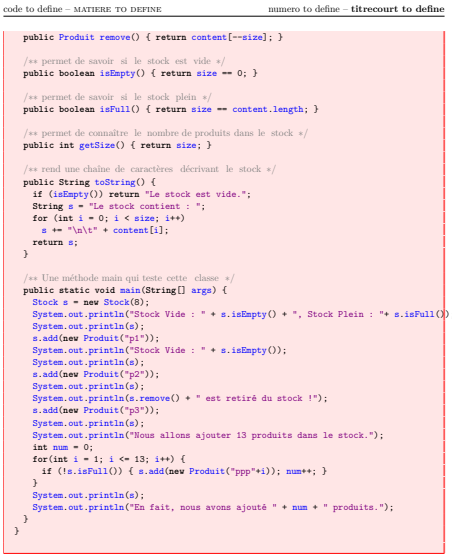
```
/**
 * Un stock d'objets instances de Produit dont la taille est fixée à la création.
 * @author F.Dagnat
 */
public class Stock {
  /** le tableau contenant les produits */
  private Produit[] content;
  /** le nombre de produits déposés */
  private int size = 0;

  /** un constructeur avec comme paramètre la taille du stock
   * @param s la taille du stock */
  public Stock(int s) { content = new Produit[s]; }

  /** rajoute un nouveau produit dans le stock
   * @param p le produit qui est rajouté */
  public void add(Produit p){
    if (p==null) return;
    content[size++] = p;
  }

  /** retire le <b>dernier</b> produit ajouté au stock et le rend en résultat */
```

1



code to define – MATIERE TO DEFINE — numero to define – **titrecourt to define**

```
  public Produit remove() { return content[--size]; }

  /** permet de savoir si le stock est vide */
  public boolean isEmpty() { return size == 0; }

  /** permet de savoir si le stock plein */
  public boolean isFull() { return size == content.length; }

  /** permet de connaître le nombre de produits dans le stock */
  public int getSize() { return size; }

  /** rend une chaîne de caractères décrivant le stock */
  public String toString() {
    if (isEmpty()) return "Le stock est vide.";
    String s = "Le stock contient : ";
    for (int i = 0; i < size; i++)
      s += "\n\t" + content[i];
    return s;
  }

  /** Une méthode main qui teste cette classe */
  public static void main(String[] args) {
    Stock s = new Stock(8);
    System.out.println("Stock Vide : " + s.isEmpty() + ", Stock Plein : "+ s.isFull());
    System.out.println(s);
    s.add(new Produit("p1"));
    System.out.println("Stock Vide : " + s.isEmpty());
    System.out.println(s);
    s.add(new Produit("p2"));
    System.out.println(s);
    System.out.println(s.remove() + " est retiré du stock !");
    s.add(new Produit("p3"));
    System.out.println(s);
    System.out.println("Nous allons ajouter 13 produits dans le stock.");
    int num = 0;
    for(int i = 1; i <= 13; i++) {
      if (!s.isFull()) { s.add(new Produit("ppp"+i)); num++; }
    }
    System.out.println(s);
    System.out.println("En fait, nous avons ajouté " + num + " produits.");
  }
}
```

TELECOM Bretagne – type to define — 2 — la date

4 Les autres fonctionnalités offertes

4.1 Inclusion de Java

Les fonctionnalités liées à l'inclusion de code Java sont regroupées au sein du fichier `java.sty`. Elles utilisent le paquetage `listings` et prédéfissent des styles de formatage.

Les trois fonctions sont :
– la macro `\java` qui permet d'insérer dans du texte des termes en utilisant la typographie java,
– l'environnement `Java` qui permet d'insérer un programme,
– la macro `inputJava` qui permet d'insérer un fichier contenant du Java.

L'environnement `Java` et la macro `\inputJava` prennent un argument optionnel qui peut contenir toutes les options du paquetage `listings`. Il est ainsi, par exemple, possible de modifier tous les choix de présentation. Parmi les options interéssantes surtout pour l'inclusion de fichier figure `firstline` et `lastline` qui permettent de n'inclure qu'une partie du fichier[6].

Par défaut :
– pour les cours, le code est toujours affiché en *footnotesize* et est numéroté (sauf en FC),
– pour les textes :
  – le code est affiché en :
    – normal, par l'environnement et dans les devoirs,
    – sinon, en *small* ;
  – pour l'inclusion (macro `\inputJava`), un cadre et un fond sont ajoutés (sauf pour les devoirs et à l'intérieur des corrections (*i.e.* dans l'environnement `correction`).

[6] Attention, tous les changements de lignes dans le fichier risque de décaler le contenu inclu par la macro !

4.2 Diverses macros

Les macros suivantes peuvent être utilisées :

- `\centereddbend` qui affiche  .
- `\afaire{quelque chose}` qui affiche **AFAIRE : quelque chose**.
- `\bfvert{toto}` qui affiche **toto**.
- `\fluo{titi}` qui affiche titi.
- `\moodle` qui affiche l'adresse de l'espace moodle du cours, par défaut utilise le type de cours ; il y a trois cours définit pour mineure Glop (668), pour majeure Glop (23) et pour option F2B201 aka IDL (174)[7]. Si un argument optionnel `uuu` est donné à la macro elle affiche l'adresse : `http://formations.telecom-bretagne.eu/fad/course/view.php?id=uuu`. Dans le fichier pdf (sans l'option `handout`), un lien vers la page est ajouté.
- `\mailtb{tutu}` qui crée un lien vers le mél de tutu (adresse `mailto:`) et affiche : `tutu@telecom-bretagne.eu`
- `\unSchema[1]{2}{3}{4}` qui permet d'inclure une figure (2) en la centrant et lui appliquant les modificateurs (1) (la syntaxe est celle de `\includegraphics`) ; (3) est la légende et (4) le label par lequel on pourra référencer la figure.
- `\annexes` qui permet de passer aux annexes et d'afficher une boite signalant que l'on passe aux annexes (voir page 12).
- Deux envirronements `remarque` et `remarques` :

```
\begin{remarque}
  Une seule remarque.
\end{remarque}
```

Remarque :

Une seule remarque.

```
\begin{remarques}
\item Une première remarque.
\item \dots
\item Une dèrnière remarque.
\end{remarques}
```

Remarques :

- *Une première remarque.*
- ...
- *Une dèrnière remarque.*

4.3 Des paquetages annexes

Un certains nombres de paquetages annexes sont fournis et ont (ou bientôt auront) une documentation associée (ils sont alors marqué par une *). Ils ne sont pas encore totalement indépendant de la classe `tb` mais j'y travaille.

Ces paquetages sont :

- `automate` : des macros pour les automates (pour l'instant pas grand chose).
- `beamerthemeTELECOMBretagne` : le thème beamer suivant la charte graphique de l'école, il se contente de charger les trois fichiers suivant :
  - `beamercolorthemeTELECOMBretagne`,
  - `beamerinnerthemeTELECOMBretagne` et
  - `beameroutерthemeTELECOMBretagne`.
- `boites-tb` : des boites utilisées pour les sujets d'examen et les correction.
- `cours` : la partie cours (chargement du thème beamer entre autre)
- `exercice`* : la gestion des exercices.
- `fractal` : des macros pour faire des composants fractal
- `java` : le paquetages Java a été présenté plus tôt dans ce document.

[7] Dans le cas d'une utilisation plus large de la classe, il conviendrait de la redéfinir plus précisément.

– `object*` : macros tikz pour faire les dessins de gestion de la mémoire.
– `ocaml` : inclusion de listings ocaml.
– `posterGET` : poster au format GET.
– `qcm` : quelques macros pour qcm.
– `sauna` : des macros de sauna pour dessiner des composants (utilisée dans le cours de master).
– `texte` : la partie document autre que cours.
– `uml*` : des macros tikz pour les schémas UML.

5 Compiler un document

5.1 Exigences

Pour compiler les documents, il est fortement conseillé d'utiliser `pdflatex`. La classe doit fonctionner avec latex+dvips+ps2pdf, mais n'a pas été testée dans cet environnement depuis longtemps.

Les paquetages latex suivants doivent etre installés sur la machines et accessibles :
– standards (dans la Texlive[8] et dispo sur l'opteron) : *babel*, *inputenc*, *fontenc*, *lmodern*, *textcomp*, *xcolor*, *graphicx*, *ifthen*, *verbatim*, *fancyhdr*, *trace*, *beamer*, *manfnt*, *calc*, *lastpage*, *newfile*, *pgfpages*, *ifpdf*, *hyperref*, *amsthm*, *geometry*, *listings*, *amssymb*, *xkeyval*
– TB : *cours*, *texte*, *boites-tb*, *java*, *exercice* ; il y a alors deux manières :
  – copier le contenu du repertoire `svn.enstb.org/ens/utils/styles/trunk` dans votre arborescence latex locale (puis `texhash` bien sur...)
  – modifier votre variable d'environnement `TEXINPUTS` pour y ajouter le repertoire contenant le paquetage
– moins courant, pas sur CTAN pour l'instant : *translator* que l'on peut recupérer sur le CVS de beamer sur sourceforge par :
```
cvs -d:pserver:anonymous@latex-beamer.cvs.sourceforge.net:/cvsroot/latex-beamer login
cvs -z3 -d:pserver:anonymous@latex-beamer.cvs.sourceforge.net:/cvsroot/latex-beamer \
co -P translator
```
  Ensuite appliquer une des deux méthodes précédemment citées.

Si dans votre document vous utilsez les macros (non documentées) `pscode` ou `herepscode`, la commande latex doit avoir l'option `-shell-escape` et vous devez avoir le script Perl `buildmacro.pl` (situé dans `svn.enstb.org/ens/utils/scripts`). Perl doit du coup être installé ainsi que :
– `latexmk`,
– `ps2eps`,
– `epstopdf`,
– le paquetage latex *pstricks*

Normalement, après une période de transition, plus de documents ne devrait contenir de postscript (si ce n'est les archives bien sur...). En effet, pour simplifier à la fois la production des figures mais aussi se passer des compléxités liées à ce processus, les documents vont basculer en `tikz` (c'est déjà vrai pour les cours).

Une documentation détaillée est disponible là : `https://info.enstb.org/enseignement/tc/MineureGlop/HOWTO`. Elle parfois obsolète, mais c'est mieux que rien.

[8] La tetex n'étant plus maintenue, votre serviteur utilise maintenant la texlive. L'opteron ne l'utilise pas encore mais cela ne saurait tarder...

5.2 Script pour aider à la compilation

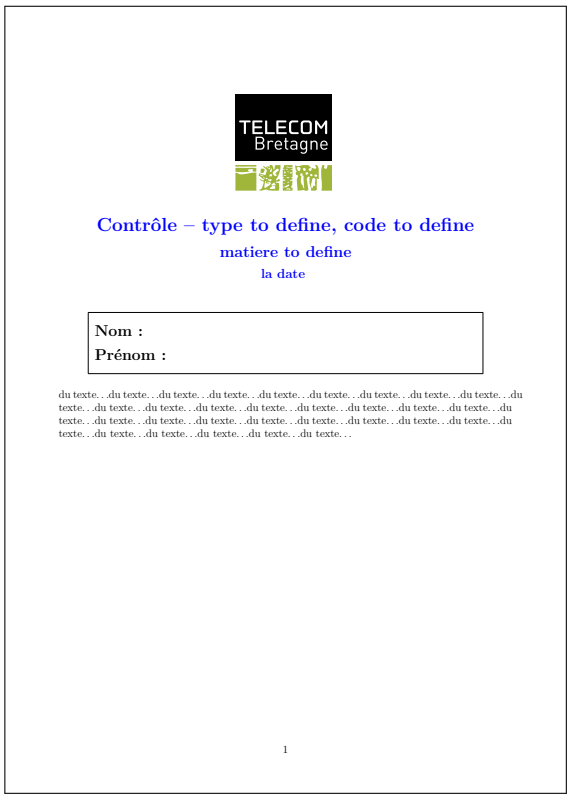
Un script, conçu et utilisé par votre serviteur, automatise un certain nombre de tâche pour produire les documents. Ce script est appelé `makeDoc.pl`. Il prend en paramètre une liste de fichier latex (l'extension tex peut être omise) et les compile et produit les pdf correspondant. Le processus de compilation est contrôlé par les options suivantes :

- `-d` : affichage des sorties de latex (par défaut, elles sont escamotées),
- `-f` : permet de forcer l'exécution complète du script (même en cas d'erreur, sauf si elles sont fatales),
- `-dd` : les fichiers générés par le processus ne sont pas effacés (aide au debogage),
- `-handout` : ajoute l'option `handout` aux documents,
- `-snm` : permet de conserver le fichier snm (si vous ne savez pas ce que c'est, c'est que vous n'avez pas besoin de cette option),
- `-aux` : permet de conserver le fichier aux (même remarque que ci-dessus),
- `-cor` : va générer la correction (utilise l'option `correction`),
- `norename` : par défaut le script renomme le fichier résultant (en sujet, correction ou cours selon son type), avec cette option le fichier résultant gardera le nom du fichier latex original.
- `mineure` : remplace l'option `majeure` du document par une option `mineure` ou l'ajoute. Les fichiers générés se voit ajouter l'extension `-mineure`.
- `cours=qqchose` : `qqchose` devient le nom utilisé pour les cours
- `sujet=qqchose` : `qqchose` devient le nom utilisé pour les sujets

Le script utilise `latexmk` pour automatiser le nombre de compilations (qui peut être grand).

Annexes

Options devoir :


TELECOM Bretagne

Contrôle – type to define, code to define

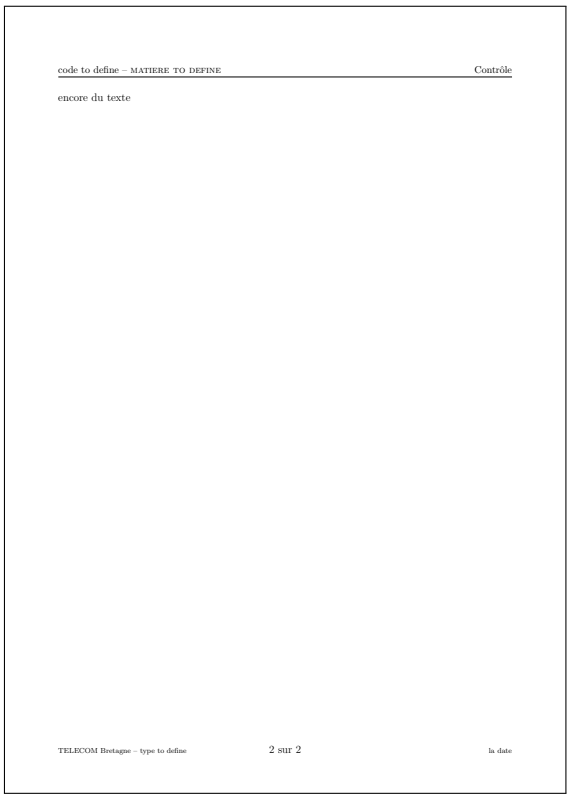
matiere to define

la date

Nom :
Prénom :

du texte...

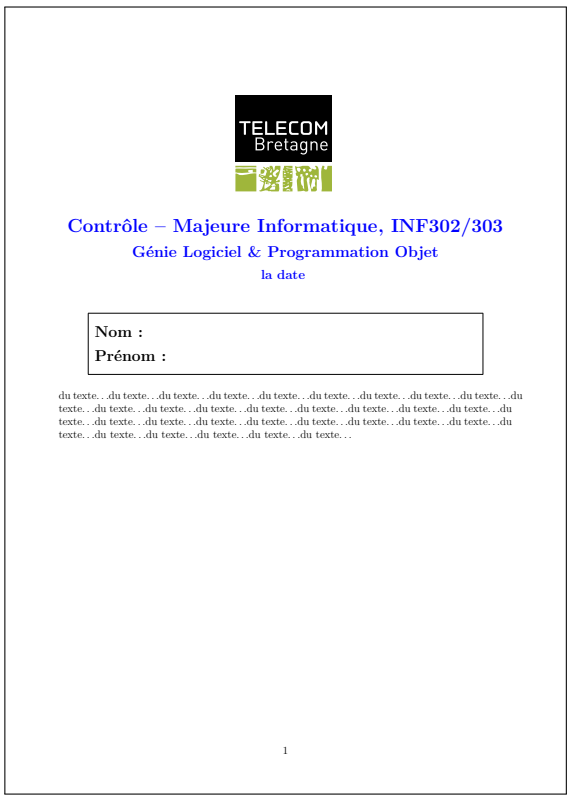
1


code to define – MATIERE TO DEFINE Contrôle

encore du texte

TELECOM Bretagne – type to define 2 SUR 2 la date

Options devoir,majeure,po,gl :


TELECOM Bretagne

Contrôle – Majeure Informatique, INF302/303

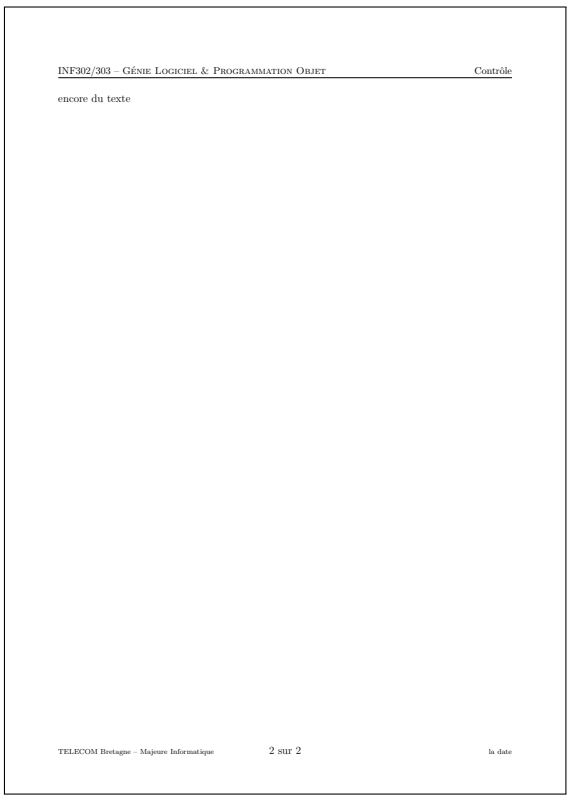
Génie Logiciel & Programmation Objet

la date

Nom :
Prénom :

du texte...

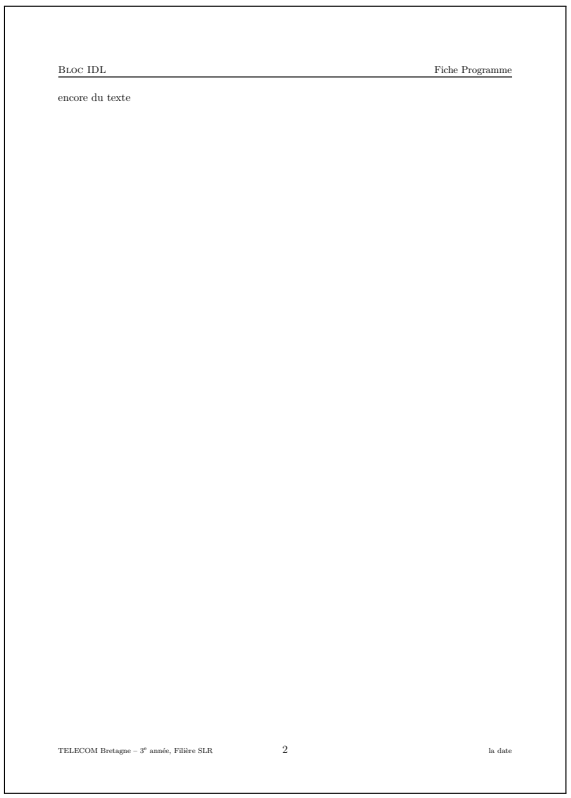
1


INF302/303 – GÉNIE LOGICIEL & PROGRAMMATION OBJET Contrôle

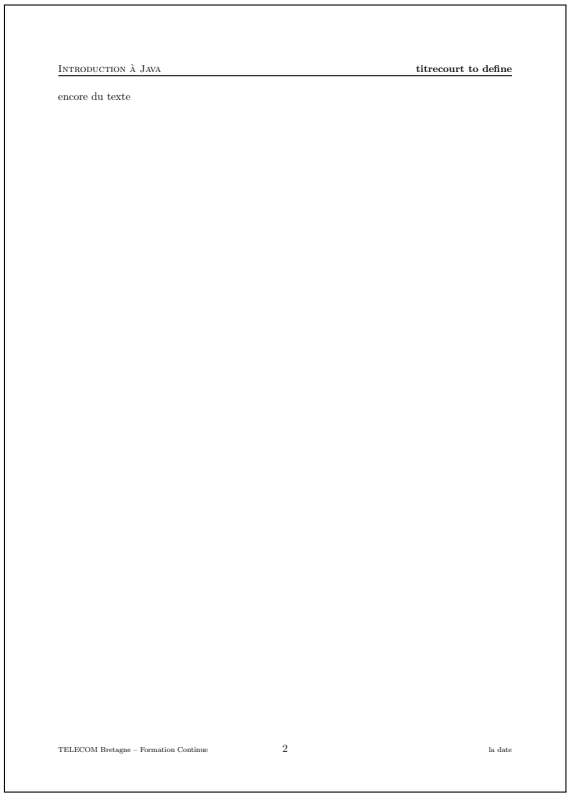
encore du texte

TELECOM Bretagne – Majeure Informatique 2 SUR 2 la date

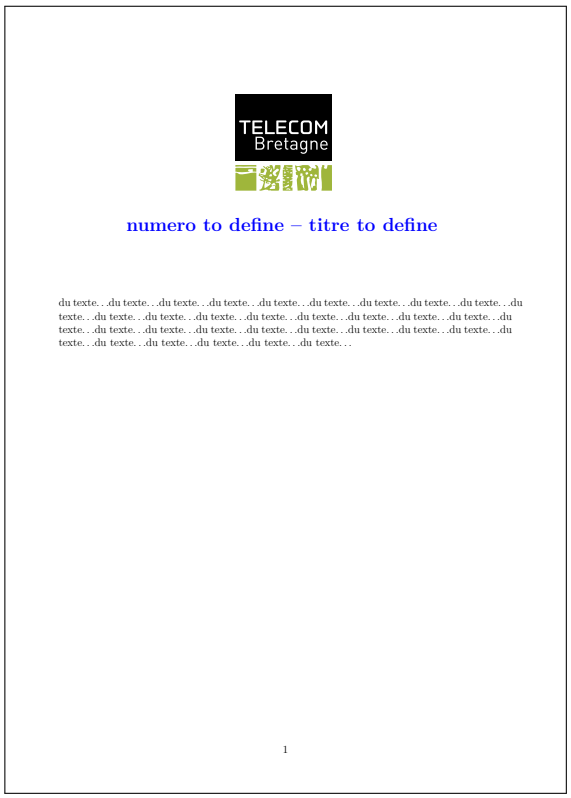
Options option,ficheprog :

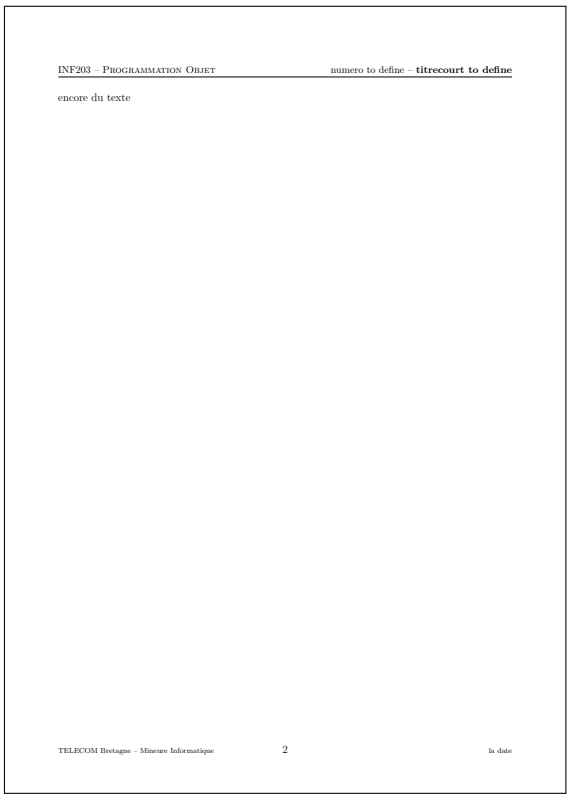


Options fc,base :



Options mineure,po :


TELECOM
Bretagne
numero to define – titre to define
du texte...
1


INF203 – Programmation Objet
numero to define – titrecourt to define
encore du texte
TELECOM Bretagne – Mineure Informatique
2
la date