M1IF03
Conception d'applications Web

OUTILS D'AIDE À LA CONCEPTION WEB (CÔTÉ SERVEUR)

LIONEL MÉDINI

OCTOBRE-DÉCEMBRE 2018

Préambule : retours sur le TP2

- Questions / erreurs communes :
  - Configuration :
    - Préférez les annotations aux fichiers XML
  - URLs :
    - chemins virtuels sur le serveur != package.NomClasse
    - Les chemins virtuels indiqués dans la config sont à partir de la racine du contexte applicatif
    - Rappel : **PAS D'URL ABSOLUE CÔTÉ CLIENT !**

Objectif de ce cours

- Prendre conscience de l'existence de nombreux outils facilitant le développement Web
  - ne plus développer *from scratch*
  - gagner du temps
  - se placer dans des conditions réelles de conception
- Savoir les catégoriser
  - langages / environnements de développement
  - fonctionnalités proposées
  - [in]/compatibilités

➔ Savoir choisir un outil adapté aux besoins d'une application

- Dans ce cours
  - Présentation [relativement] détaillée de certains d'entre eux
  - Liste nécessairement non exhaustive
  - Présentations nécessairement succinctes

Plan du cours

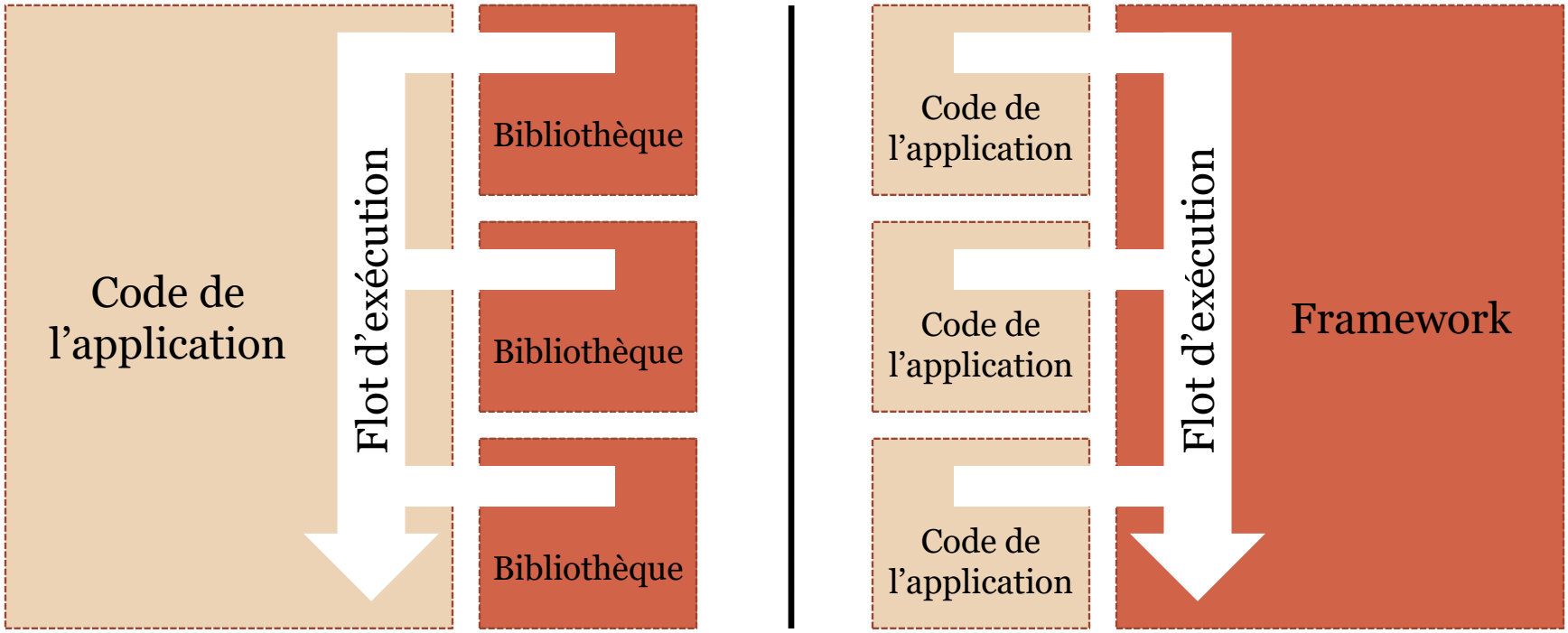
- Bibliothèques et frameworks
  - Principe de l'inversion de contrôle
  - Différence entre bibliothèques et frameworks
- Aperçu des outils
  - Bibliothèques Web
    - Bibliothèques de composants
    - Bibliothèques AJAX
  - Frameworks Web
    - MVC
    - Conteneurs légers
    - Conteneurs lourds
  - Autres types d'outils Web applicatifs
    - CMS
    - Portlets
    - APIs des applis Web connues
- Conclusion

Inversion de contrôle

- Principe général
  - Une application (Web) complexe fait nécessairement appel à du code externe pour gérer des services non métier
    - sécurité
    - persistance
    - …
  - ➔ Qui contrôle le flot d'exécution d'une application ?
    - votre code
    - un des outils que vous utilisez
  - En programmation classique
    - D'où provient le main ?
  - En MVC
    - Qui dirige le contrôleur ?

Inversion de contrôle

- Différence bibliothèque / framework



  - Remarque : dans la littérature, on trouve l'appellation « framework » pour beaucoup de choses qui n'en sont pas

Aperçu des outils

- La réutilisation comme principe général de conception
  - Même démarche qu'en conception « classique »
    - AWT, Swing, JavaFX…
  - Spécificité des outils Web
    - Nombreux
    - Hétérogènes
    - Notion de framework plus répandue
  - Problématique : quel(s) outil(s) choisir ?

Bibliothèques Web

- But
  - Ensemble de composants pour réaliser une ou plusieurs fonctionnalités
- Spécificités de la plupart des bibliothèques dédiées au Web
  - Dédiées à la couche interface
  - Dédiées aux communications AJAX
  - Services spécifiques aux serveurs Web (sécurité)
- Choix d'une bibliothèque
  - Diffusion / adoption par une communauté
  - Adaptée aux autres outils utilisés
  - Doit être transparent pour l'utilisateur

Bibliothèques Web

- Bibliothèques de services côté serveur
  - Persistance
    - Java : non spécifiquement dédié au Web
      - ORMs, JPA
    - PHP : bibliothèques de code MySQL
      - Flat-file SQL
  - Sécurité
    - Java : WebCastellum
  - Authentification
    - Java : JGuard (basé sur JAAS) ; JXplorer (support LDAP)
    - PHP : PHP OpenID Library
  - Compilateurs JavaScript
  - …

Bibliothèques Web

- Bibliothèques d'interface
  - Tags JSP
    - JSF
  - Templates de sites
    - Présentation basique
      - De moins en moins utilisées (au profit de CSS)
    - Composants de pages
      - Générés dynamiquement
      - Exemples : tableaux de données, formulaires, menus…
  - JavaScript
    - Composants d'interface
    - Composants de programmation
      - Tri, organisation, ajout, transformation XSLT

Bibliothèques Web

- Bibliothèques AJAX
  - Bibliothèques « directes »
    - Bibliothèques de fonctions JavaScript pour faciliter le codage
      - Peu structurées, ne sont utilisables que pour de petites applications
    - Éventuellement, des outils côté serveur facilitant la génération de pages liées à ces bibliothèques
      - Nécessitent d'avoir une vue claire de l'application
    - Exemples
      - Génériques : jQuery, SAJAX, DHTMLX, Fleejix.js, JsHTTPRequest, My Library
      - Java : JSP Tags Library
      - PHP : XAJAX, PhpLiveX
      - .Net : DotNetRemoting Rich Web Client SDK for ASP.NET , ASP.Net AJAX
      - …

Bibliothèques Web

- Bibliothèques AJAX
  - Bibliothèques « indirectes »
    - Dédiées à un langage de programmation
    - Utilisent un compilateur pour générer du JavaScript
      - Programmation plus claire / propre qu'avec plusieurs langages
      - Code généré côté client non maîtrisé
    - Exemples
      - Java
        - DWR, GWT, IceFaces
      - Python
        - Pyjamas
      - .Net
        - ASP.Net AJAX
      - …

Bibliothèques Web indirectes : Exemple

- Google Web Toolkit
  - Présentation
    - Bibliothèque de composants et de génération d'applications Web
    - Bibliothèque de composants de communication HTTP asynchrone
    - Compilateur Java→JavaScript
    - Existence d'une bibliothèque d'extensions : GWTexT
  - Remarque
    - S'utilise plutôt comme une API de développement d'applications classiques que Web-based
  - Site Web

    http://code.google.com/webtoolkit/
  - JavaDoc

    http://google-web-toolkit.googlecode.com/svn/javadoc/latest/index.html

Frameworks Web

- Remarque préliminaire
  - Un serveur Web est déjà un framework en soi
- Un framework Web est une couche d'abstraction supplémentaire par rapport au serveur
  - Il doit apporter une valeur ajoutée supplémentaire
    - Pattern MVC
    - « Orienté-AJAX »
    - Prise en charge d'aspects (services) annexes :
      Transactions, sécurité, communication avec des objets distants…
    - « philosophie » particulière :
      Dédié à la GED, à la communication entre utilisateurs, à l'éducation…
  - Il doit être nécessaire pour la réalisation du cahier des charges

Frameworks Web

- Fonctionnalités proposées
  - Composants
    - Format des composants prédéfini
    - Gestion du workflow de composants métier (beans)
  - Génération de vues
    - Structure des pages qui reviennent régulièrement
      - Ex. : tableaux de données, graphiques, etc.
    - Génération de sorties sous forme de services Web
    - Souvent liée à l'utilisation d'un moteur de templates (cf. CM templating)
  - Programmation par configuration
    - Introspection pour générer des éléments de pages ou de la logique métier
    - Ex. : intégrer un ORM capable de persister directement des objets

Frameworks Web

- Fonctionnalités annexes
  - Sécurité
    - Authentification, gestion des droits, des rôles, limitation des accès
  - Interfaces avec les BD
    - Dispense d'avoir à écrire du code spécifique à une base (API standardisées, ORM, transaction, migration de versions…)
  - Réécriture d'URLs
    - Permet d'éviter les URL CGI
    - Ex. : /page.cgi?cat=science&topic=physics →/page/science/physics
  - …

Frameworks Web MVC

- Problématique
  - Structurer l'ensemble des servlets et JSP d'une application
  - Organiser le flot de contrôle de l'application
- Historique de la structuration d'applications
  - Modèle 1 : des JSP dans tous les sens…
  - Modèle 2 : des servlets pour contrôler le flot, des JSP pour les traitements
  - Modèle MVC push-based
  - Modèle MVC pull-based
- Source

  http://struts.apache.org/1.x/userGuide/introduction.html

Frameworks Web MVC

- Différents types de frameworks
  - Push-based
    - Un contrôleur qui utilise des *actions* pour calculer les contenus
    - Ces contenus sont « poussés » à la couche vue
    - Exemples
      - Java : Struts, Spring
      - Python : Django
      - Ruby : Ruby on Rails
      - PHP : Symfony, CakePHP
      - .Net : ASP .Net MVC

Frameworks Web MVC

- Différents types de frameworks
  - Pull-based (ou component-based)
    - La vue « tire » les contenus de plusieurs contrôleurs dédiés à des tâches spécifiques
    - Plusieurs contrôleurs utilisent des actions peuvent participer à la création d'une seule vue
    - *Cf.* contrôleurs de cas d'utilisation
    - Exemples
      - Java : Struts2, Tapestry, JBoss Seam
      - Python : Zope
      - .Net : DotNetNuke

Frameworks Web MVC

- Comparatif des frameworks Web

  http://en.wikipedia.org/wiki/Comparison_of_web_application_frameworks

Frameworks MVC : Struts

- Présentation
  - Framework MVC de type 2
  - Origine : Mai 2000, Craig R. McClanahan
  - URL : http://struts.apache.org/
  - Javadoc : http://struts.apache.org/1.x/struts-core/apidocs/
- Contenu
  - Un contrôleur principal et des contrôleurs délégués
  - Une bibliothèque de tags JSP spécifique
  - Un outil de gestion des formulaires
    - mapping formulaires / objets Java
    - validation des formulaires
  - Moteur de templates (Tiles)
  - ...

Frameworks MVC : Struts

- Contrôleur
  - ActionServlet (contrôleur général) : intercepte les requêtes et les dispatche en fontion des URL (fichier struts-config.xml) vers les actions correspondantes
  - Actions (contrôleurs délégués) : gèrent la communication avec le modèle et renvoient les résultats à la vue
- Modèle
  - N'importe quelles classes connues par les actions (POJO)
  - JavaBeans : standardisent les propriétés accédées par la vue
- Vue
  - La plupart du temps, des JSP qui affichent le résultat des traitements du modèle
  - Peuvent être étendues : JSF, AJAX, etc.

Frameworks MVC : Spring

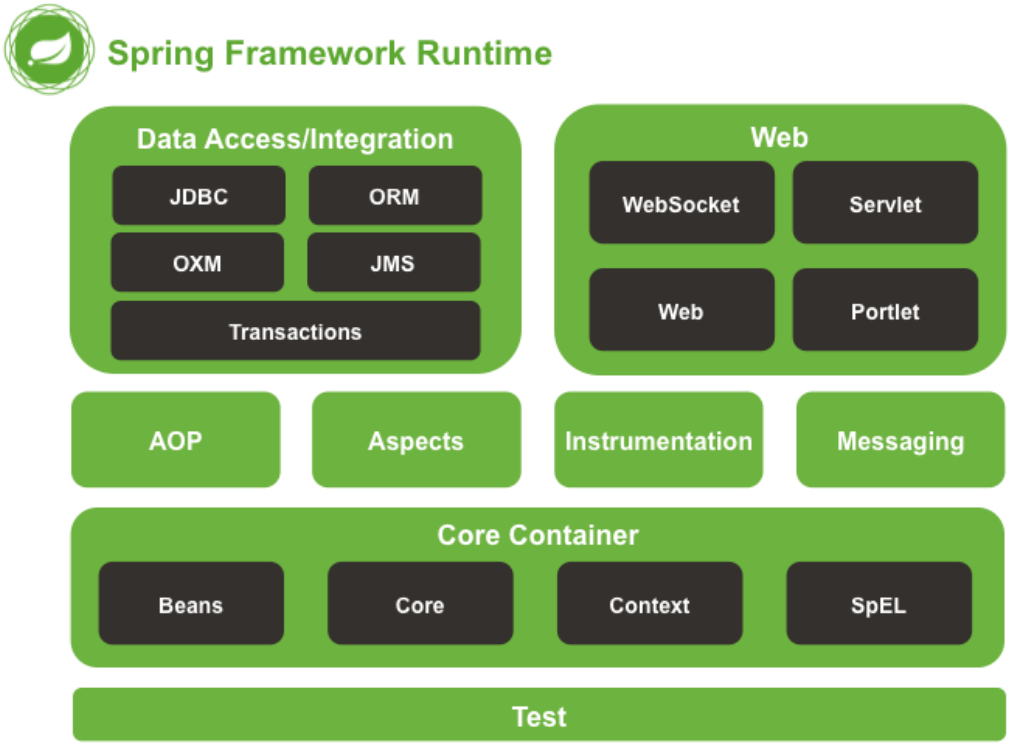
- Historique
  - Juin 2003 : sortie de la première version de Spring framework
  - 2004 : création de la société SpringSource par Rod Johnson
    - publication du livre“Expert One-on-One J2EE Design and Development” qui justifie la création de Spring
  - 2006 : sortie de la V. 2 de Spring
  - 2008 : rachat de Spring par VMWare
    - Sortie de la V. 3 du framework
    - Nombreux sous-projets : Spring Security, Spring Data, Spring AMQP…
- Version courante : 5.1.2
  - Doc : https://docs.spring.io/spring-framework/docs/5.1.2.RELEASE/spring-framework-reference/

Frameworks MVC : Spring

- Fondements
  - Réaction à Java 2 EE
    - EJB2 : trop complexes
    - Framework intégrant de (trop) nombreuses fonctionnalités
  - → Architecture autour d'un « conteneur léger »
    - → Les composants sont des POJO
  - → Intégration de fonctionnalités fournies par d'autres projets Open Source
    - → Struts, Hibernate, JUnit, AspectJ, JSF…
  - → La configuration tient une part centrale de la conception
    - → « Opinionated »

Frameworks MVC : Spring

- Architecture globale



Source : http://docs.spring.io/spring-framework/docs/current/spring-framework-reference/html/images/spring-overview.png

Frameworks MVC : Spring

- Spring Core container
  - Rôle
    - Implémente le pattern IoC
      - Fournit un conteneur
      - Gère et met en œuvre les composants (beans)
    - Applique la configuration
      - Injection de dépendances par constructeur ou par setters
    - Fournit un contexte applicatif
    - Fournit des services annexes
      - AOP, communication orientée message, événements, services spécifiques à l'application (Web…)

Frameworks MVC : Spring

- Spring Core container
  - Interfaces
    - `org.springframework.beans.BeanFactory`
      - Instancie les beans
      - Injecte les dépendances / gère la configuration
    - `org.springframework.context.ApplicationContext`
      - Dérive de la précédente
      - Représente le conteneur (!)
      - Rajoute des services : AOP, messages, événements…

Frameworks MVC : Spring

- Spring Core container
  - Implémentations de ApplicationContext
    - Dans les applications standalone
      - `ClassPathXmlApplicationContext` ou `FileSystemXmlApplicationContext`
      - Dépend de la méthode d'accès au fichier de config Spring
      - À instancier dans la classe principale de l'application
      - Exemples

```
ApplicationContext context = new ClassPathXmlApplication
Context("beans.xml");
```

ou

```
ApplicationContext context = new ClassPathXmlApplication
Context(new String[] {"services.xml", "daos.xml"});
```

Frameworks MVC : Spring

- Spring Core container
  - Implémentations de ApplicationContext
    - Dans les applications Web
      - Instanciation par le conteneur Web à l'aide du fichier de configuration (web.xml)
      - Utilisation d'un ContextLoader
        - `org.springframework.web.context.ContextLoaderListener` (à partir de Servlet 2.4)
        - Remarque : `ContextLoaderServlet` (jusqu'à Servlet 2.3) ne fait plus partie de l'API Spring 3.0

Frameworks MVC : Spring

- Spring Core container
  - Implémentations de ApplicationContext
    - Dans les applications Web
      - Exemple de fichier web.xml

```
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>/WEB-INF/daoContext.xml
               /WEB-INF/applicationContext.xml</param-value>
</context-param>

<listener>
  <listener-class>org.springframework.web.context.ContextLoader
Listener</listener-class>
</listener>
```

Frameworks MVC : Spring

- Spring Core container
  - Implémentations de ApplicationContext
    - Dans les applications Web
      - Remarques :
        - Un IDE peut générer cette configuration automatiquement
        - Il faut s'assurer que le fichier web.xml est bien pris en compte par le conteneur de servlets
        - Les fichiers XML passés en paramètres permettent de créer des contextes

Frameworks MVC : Spring

- Spring Core container
  - Composants
    - Les beans Spring sont des POJOs
    - Instanciés par le conteneur (à partir d'un nom de classe)
    - Le nombre d'instances dépend de leur « scope »
      - Singleton (défaut)
      - Prototype : une instance par dépendance d'un autre bean
      - Request, session, global session : spécifique au conteneur Web
      - User-defined

Frameworks MVC : Spring

- Spring Core container
  - Configuration
    - Définit un ensemble de beans
    - Précise leur référentiel de dépendances
      - Valeurs d'initialisation
      - Collaborateurs
    - 3 syntaxes
      - XML
      - Annotations
      - Programmation
    - Remarque : il peut y avoir plusieurs configurations dans un même conteneur

Frameworks MVC : Spring

- Spring Core container
  - Configuration
    - Configuration par fichier XML

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/spring-beans.xsd">
  <bean id="toto" class="monPackage.beans.TotoBean" >
    <constructor-arg type="String" value="Bonjour"/>
    <property name="prop1" value="Il fait beau"/>
    <property name="prop2" ref="Titi"/>
  </bean>

  <bean id="Titi" class="monPackage.beans.TitiBean"/>
</beans>
```

Injection par constructeur

Injection par setter

Collaborateur

Frameworks MVC : Spring

- Spring Core container
  - Configuration :
    - Configuration par annotations
      - Annotations de classes
        - `@Component` : composant générique
        - `@Repository` : dérive de `@Component`, dédié à la persistence
        - `@Service` : dérive de @Component, dédié aux services (objets du modèle)
        - `@Controller` : dérive de @Component, dédié à la présentation (!)
      - Annotations internes aux classes (setters)
        - `@Required` : force le conteneur à injecter une valeur (définie explicitement ou par autowiring)
        - `@Autowired` : injection par résolution du référentiel de dépendances

Frameworks MVC : Spring

- Spring Core container
  - Configuration :
    - Configuration par annotations
      - Nécessite un fichier de configuration – presque – vide

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:context="http://www.springframework.org/schema/context"
 xsi:schemaLocation="http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans.xsd
  http://www.springframework.org/schema/context
  http://www.springframework.org/schema/context/spring-
context.xsd">
   <context:annotation-config/>
   <context:component-scan base-package="mon.package.de.base"/>
</beans>
```

Frameworks MVC : Spring

- Spring Core container
  - Configuration :
    - Exemple de bean annoté

```
@Service
public class SimpleMovieLister {
 private MovieFinder movieFinder;
 private ActorFinder actorFinder;
  @Required
  public void setMovieFinder(MovieFinder movieFinder) {
      this.movieFinder = movieFinder;
  }
  @Autowired
  public void setActorFinder(MovieFinder actorFinder) {
      this.actorFinder = actorFinder;
  }
}
```

Frameworks MVC : Spring

- Spring Core container
  - Configuration :
    - Configuration par programmation (1/2)
      - On crée une classe de configuration
        - Annotation : `@Configuration`
      - On y déclare les beans
        - Annotation : `@Component`
      - On instancie le contexte en lui passant cette classe en paramètre

Frameworks MVC : Spring

- Spring Core container
  - Configuration :
    - Configuration par programmation (1/2)
      - Exemple

```
@Configuration
public class AppConfig {
  @Component
  public MyService myService() {
      return new MyServiceImpl();
  }
}
-------- dans le main --------
ApplicationContext ctx = new
        AnnotationConfigApplicationContext(AppConfig.class);
MyService myService = ctx.getBean(MyService.class);
myService.doStuff();
```

Frameworks MVC : Spring

- Spring Core container
  - Configuration :
    - Configuration par programmation (2/2)
      - Autre méthode
        - Instancier un contexte vide
        - Utiliser `context.register()`

```
public static void main(String[] args) {
  AnnotationConfigApplicationContext ctx = new
AnnotationConfigApplicationContext();
  ctx.register(AppConfig.class, OtherConfig.class);
  ctx.register(AdditionalConfig.class);
  ctx.refresh();
  MyService myService = ctx.getBean(MyService.class);
  myService.doStuff();
}
```

Frameworks MVC : Spring

- Spring Core container
  - Configuration :
    - Résolution automatique du référentiel de dépendances (autowiring)
      - S'applique spécifiquement à chaque bean

```
<bean id="Titi" class="TitiBean" autowire="constructor"/>
```

        - Annotation `@Autowired`
      - Valeurs
        - `no` (défaut) : pas d'autowiring
        - `byName` : par nom de propriété
        - `byType` : par type de propriété
        - `constructor` : par type d'arguments du constructeur

Frameworks MVC : Spring

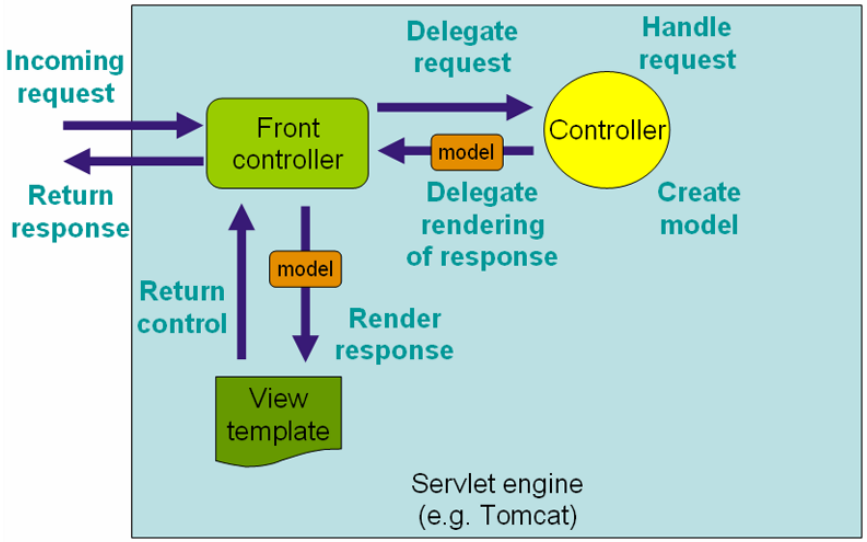
- Spring Core container
  - Configuration :
    - Gestion du cycle de vie
      - Il est possible de spécifier les méthodes de cycle de vie d'un bean dans la configuration
      - On appelle ces méthodes « initialization callback » et « destruction callback »

```
<bean id="exampleBean" class="examples.ExampleBean"
                       init-method="init"
                       destroy-method="destroy"/>
```

      - Spring fournit des mécanismes plus fins à l'aide des interfaces `LifeCycle` et `LifeCycleProcessor`

Frameworks MVC : Spring

- Spring Web MVC
  - MVC de type 2
    - Front controller : `DispatcherServlet` (fournie par Spring)
    - Contrôleurs délégués : composants (`@Controller`)



Source : http://docs.spring.io/spring/docs/3.2.x/spring-framework-reference/html/mvc.html

Frameworks MVC : Spring

- Spring Web MVC
  - Exemple de configuration (web.xml)

```
<web-app>
    <servlet>
        <servlet-name>example</servlet-name>
        <servlet-class>org.springframework.web.servlet.
DispatcherServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>example</servlet-name>
        <url-pattern>/example/*</url-pattern>
    </servlet-mapping>
</web-app>
```

    - Remarques
      - Cette configuration nécessite un fichier de configuration de composants nommé : `/WEB-INF/example-servlet.xml`
      - Mapper les URLs sur `/*` est une mauvaise idée…

Frameworks MVC : Spring

- Spring Web MVC
  - Exemple de contrôleur annoté

```
@Controller
@RequestMapping("/appointments")
public class AppointmentsController {
    private final AppointmentBook appointmentBook;

    @Autowired
    public AppointmentsController(AppointmentBook apptmentBook) {
        this.appointmentBook = apptmentBook;
    }

    @RequestMapping(method = RequestMethod.GET)
    public String get() {
        return "appointments/today";
    }
}
```

Frameworks MVC : Spring

- Spring Web MVC
  - Méthodes de service (Handler methods)
    - Annotées avec `@RequestMapping` (ou `@GetMapping`, `@PostMapping`…)
    - Permettent
      - De récupérer les paramètres de la requête
      - De faire du data binding entre les paramètres et le modèle
      - D'appeler les beans concernés
      - De passer les infos (`Model`) nécessaires à la vue pour générer la réponse
    - Signature « flexible »
      - Paramètres
        - Model, `@ModelAttribute`
        - Paramètres de la requête : `@RequestParam`
        - Paramètres « classiques des servlets : `ServletRequest`, `ServletResponse`, `HttpSession`
        - …
      - Valeurs de retour
        - String : nom de vue (cf. slide précédent)
        - Objet `View`
        - Objet `ModelAndView`
        - String annotée `@ResponseBody`
        - …
      - https://docs.spring.io/spring/docs/current/spring-framework-reference/web.html#mvc-ann-methods

Frameworks MVC : Spring

- Spring Web MVC
  - Méthodes de service (Handler methods)
    - Exemples

```
@RequestMapping
@ModelAttribute
public void populateModel(@RequestParam String number, Model model) {
    model.addAttribute(accountRepository.findAccount(number));
    // add more ...
}
```

```
@PostMapping("/login")
public ModelAndView login(LoginData loginData) {
    if (LOGIN.equals(loginData.isValid())) {
        return new ModelAndView("success", new User("test"));
    } else {
        return new ModelAndView("failure", null);
    }
}
```

Frameworks MVC : Spring

- Spring Web MVC
  - View resolving
    - Objectif : faire correspondre une vue au retour du contrôleur
    - Interface `View`
      - Traite la requête en fonction d'une technologie de vue (JSP, JSF…)
    - Interface `ViewResolver`
      - Fournit un mapping entre nom de vue et objet `View`

Frameworks MVC : Spring

- Spring Web MVC
  - View resolving
    - Exemple de configuration

```
<bean id="viewResolver"
class="org.springframework.web.servlet.view.UrlBasedViewResolver">
  <property name="viewClass"
value="org.springframework.web.servlet.view.JstlView"/>
  <property name="prefix" value="/WEB-INF/jsp/"/>
  <property name="suffix" value=".jsp"/>
</bean>
```

Frameworks MVC : Spring

- Conclusion
  - Avantages
    - Légèreté du framework
    - S'appuie sur des solutions open source éprouvées
    - Possibilité de « plugger » d'autres fonctionnalités
    - Configuration explicite des applications
    - Très utilisé
    - Documentation abondante
  - Faiblesses
    - Complexité croissante
      - Beaucoup de sous-projets
      - 3 types de configurations possibles
    - Choix entre Spring et Java EE moins évident
      - EJB 3.0 plus simples

Autres types d'outils Web applicatifs

- Systèmes de gestion de contenus (CMS)
  - Outils collaboratifs → gestion des utilisateurs
  - Outils de gestion électronique de documents (GED)
    → moteur de workflow
    → support de stockage et de publication de différents types de contenus
    → templates de pages et éditeurs WYSIWYG
  - Modules divers en fonction de la finalité du framework
  - Exemples
    - PHP : SPIP, PHP-Nuke, Joomla!, WordPress
    - Python : Plone, Zend
    - Java : OpenCMS, AlFresco

Autres types d'outils Web applicatifs

- Outils à base de portlets / widgets
  - Principe : juxtaposer plusieurs contenus issus d'applications différentes dans la même interface Web
  - Souvent associé à la notion de portail
  - Exemple : le portail étudiant de l'université, iGoogle, NetVibes
  - Technologies
    - Java : WSRP (JSR 168 et 286), Liferay
    - PHP : PhpPortlet

Autres types d'outils Web applicatifs

- APIs d'applications Web externes
  - Principe : interfacer son application avec une plus connue
  - Nombreux exemples dans le Web 2.0 :
    Google (Calendar, Mail, Wave…), FaceBook, YouTube, Ebay…
  - ➔ Un moyen rapide d'améliorer vos applications
  - ➔ Permet d'attirer des utilisateurs
  - ➔ Ne doit pas vous faire perdre de vue la finalité initiale de votre application

  - Liste de 20 000+ APIs disponibles (octobre 2018)
    http://www.programmableweb.com/apis/directory

Conclusion

- La réutilisation comme principe général de conception
  - Objectif : limiter le plus possible les développements à la logique métier
  - Spécificités des outils Web
    - peut-être le domaine le plus exploré et où il y a le plus d'outils disponibles
    - Évolution rapide des technologies (et des modes)
    - Cependant, de nombreuses technos à l'intérieur d'une même application
      - ➔ Autant de fonctionnalités pour lesquelles trouver des outils
      - ➔ Ne pas « se perdre dans la stack »

Conclusion

- La réutilisation comme principe général de conception
  - Sélectionner les outils disponibles…
    - Un framework
    - Des bibliothèques
  - …en fonction de vos besoins
    - Nécessite d'avoir correctement spécifié les besoins et réalisé le travail d'analyse

Conclusion

- La réutilisation comme principe général de conception
  - Vérifier la compatibilité
    - Entre les outils
    - Avec les navigateurs
    - Avec les autres systèmes avec lesquels vous voulez vous interfacer
  - Évaluer le travail d'intégration

Conclusion

- Choix d'un framework
  - Identifier le gain : services proposés / lourdeur de l'outil
  - S'attacher à la finalité d'un framework et non à ce que l'on peut faire avec
    - Les utilisateurs / autres développeurs peuvent être perdus par une utilisation non standard d'un outil
      - Utilisabilité
      - Maintenance
  - Évolutivité des solutions proposées
    - Penser à l'évolution de votre application
      - Passage à l'échelle
      - Nouveaux services
      - Intégration de technologies futures

Conclusion

- Modularité : penser composants dès les spécifications
  - Précision de la phase de conception et d'analyse (cahier des charges)
  - Rechercher l'existant avant de développer (bibliothèques disponibles)
  - Si l'interface d'une bibliothèque ne correspond pas à vos besoins :
    - Pouvez-vous / devez-vous modifier vos specs ?
    - Éventuellement, utiliser un pattern adapter
    - Sinon, le produit est-il fait pour vous ?

Conclusion

- Modularité : penser composants dès les spécifications
  - Utiliser des solutions standard
    - Surtout si vos applications s'insèrent dans un SI existant et si d'autres peuvent devoir s'interfacer avec
    - Prévoir la possibilité de changer radicalement d'interface
      - RIA / RDA
      - Adaptation aux navigateurs / terminaux mobiles
      - Services Web

Conclusion générale

- Tendances actuelles des technologies Web
  - Standards du W3C
    - Balises sémantiques (HTML5)
    - Mise en forme avancée (CSS Grid, Selectors…)
    - Interaction dynamique avec les contenus (Canvas, audio, vidéo)
    - Prise en charge de nouveaux matériels (Device APIs)
    - Amélioration de la prise en charge des interactions utilisateur (Drag'n'drop, Web Components)
  - ECMAScript / JavaScript
    - Programmation structurée (classes ES5, TypeScript)
    - Programmation bas niveau (APIs HTML5)
    - Déport de code côté client (frameworks applicatifs)
    - Communication asynchrone / non bloquante (Web workers, Fetch API)
  - Performance
    - Mode déconnecté (Service workers, Progressive Web Apps)
    - Packaging d'applications (Web Packaging)
    - Push serveur (HTTP/2)

➔ M1IF13 ;-)

Références

- Références utilisées pour ce cours
  - Bibliothèques et frameworks
    - Général
      - http://sourceforge.net/softwaremap/index.php
      - http://en.wikipedia.org/wiki/Web_application_framework
    - Listes et comparatif d'outils
      - http://en.wikipedia.org/wiki/Comparison_of_web_application_frameworks
      - http://java-source.net/open-source/content-managment-systems

Références

- Références utilisées pour ce cours
  - Bibliothèques et frameworks
    - Spécifiques
      - OpenID : https://openid.pbworks.com/Libraries
      - LDAP : http://en.wikipedia.org/wiki/List_of_LDAP_software
      - AJAX : http://en.wikipedia.org/wiki/Ajax_framework
        http://chandlerproject.org/Projects/AjaxLibraries
        http://ajaxpatterns.org/Java_Ajax_Frameworks
        http://ajaxpatterns.org/PHP_Ajax_Frameworks

Références

- Références utilisées pour ce cours
  - Spring
    - http://spring.io
    - http://docs.spring.io/spring/docs/3.2.4.RELEASE/spring-framework-reference/html/overview.html
    - http://docs.spring.io/spring/docs/3.2.5.BUILD-SNAPSHOT/spring-framework-reference/html/beans.html
    - http://www.jmdoudoux.fr/java/dej/chap-spring.htm