M1IF03
Conception d'applications Web

OUTILS D'AIDE AU DÉVELOPPEMENT D'APPLICATIONS WEB

LIONEL MÉDINI

JANVIER 2023

Plan du cours

- Introduction
- Outils côté serveur
- Outils côté client
- Ressources externes
- Conclusion

Objectif de ce cours

- La réutilisation comme principe général de conception
  - Même démarche qu'en conception « classique »
    - Ne plus développer *from scratch*
    - Gagner du temps
    - Se placer dans des conditions réelles de conception
  - Spécificité des outils Web
    - Nombreux
    - Hétérogènes
    - Notion de framework plus répandue
  - ➔ Quel(s) outil(s) choisir ?

Objectif de ce cours

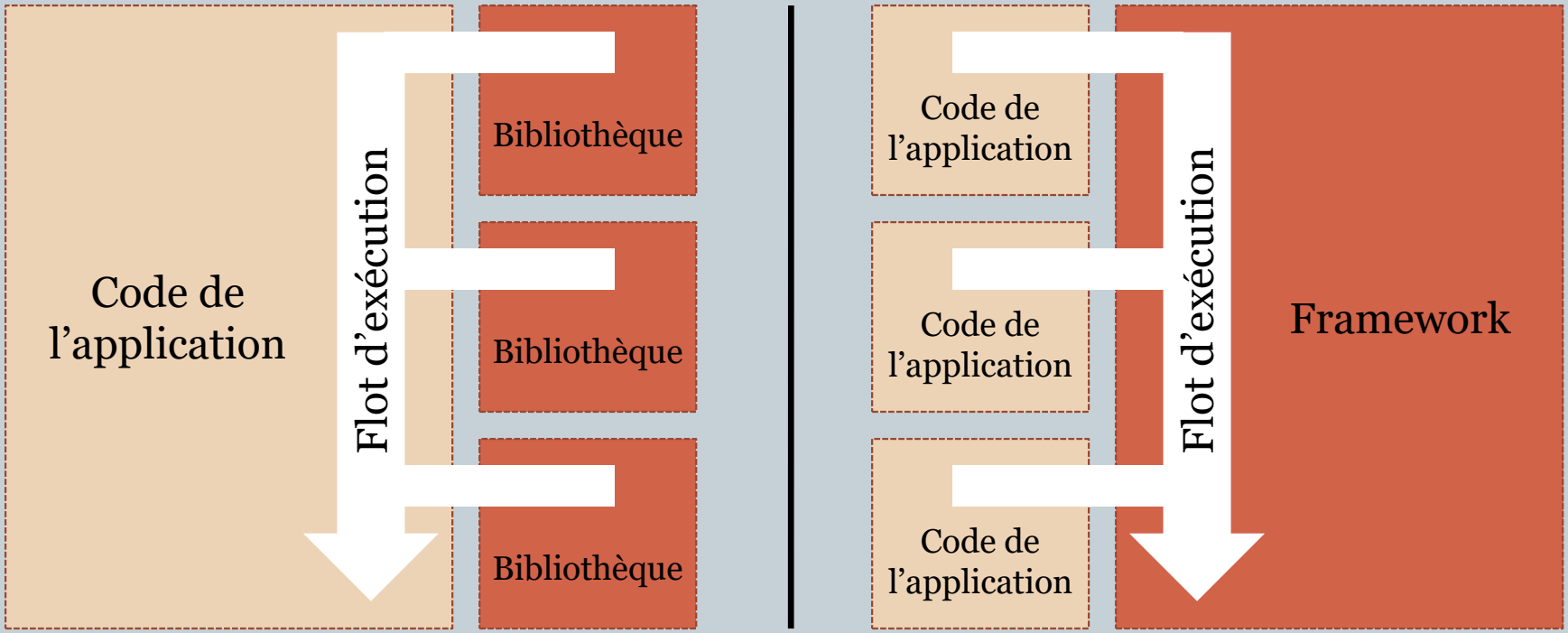
- Choisir un outil adapté aux besoins d'une application
  - Connaître l'existence des outils
  - Savoir les catégoriser
    - type : ce qu'il (ne) peut (pas) faire
    - bibliothèque *vs*. framework
    - localisation : serveur / client
    - langages / environnements de développement
    - fonctionnalités proposées
    - [in]compatibilités
  - Dans ce cours
    - Liste nécessairement non exhaustive
    - Présentations de certains outils
      - …Nécessairement succinctes

Rappel : 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 ?

Rappel : Inversion de contrôle

- Différence bibliothèque / framework



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

Plan du cours

- Introduction
- Outils côté serveur
- Outils côté client
- Ressources externes
- Conclusion

Bibliothèques côté serveur

- 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 côté serveur

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

→ De plus en plus sous forme de plugins de frameworks

Bibliothèques côté serveur

- 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…
  - → Cf. CM Templating

Frameworks côté serveur

- 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-services »
    - 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 côté serveur

- Fonctionnalités proposées
  - Framework applicatif “classique”
    - Format des composants métier (beans) prédéfini
    - Résolution du référentiel de dépendances
    - Programmation par configuration
  - Fonctionnalités Web
    - Routage des requêtes
    - MVC
    - Négociation de contenus
    - Templating des réponses
    - Renvoi automatique d’erreurs HTTP

Frameworks côté serveur

- 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 pull-based
  - Modèle MVC push-based
- Source

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

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

- Différents types de frameworks
  - Push-based
    - Un contrôleur qui utilise des *actions* ou *beans* 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

- Comparatif des frameworks Web

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

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)
  - ...

Struts

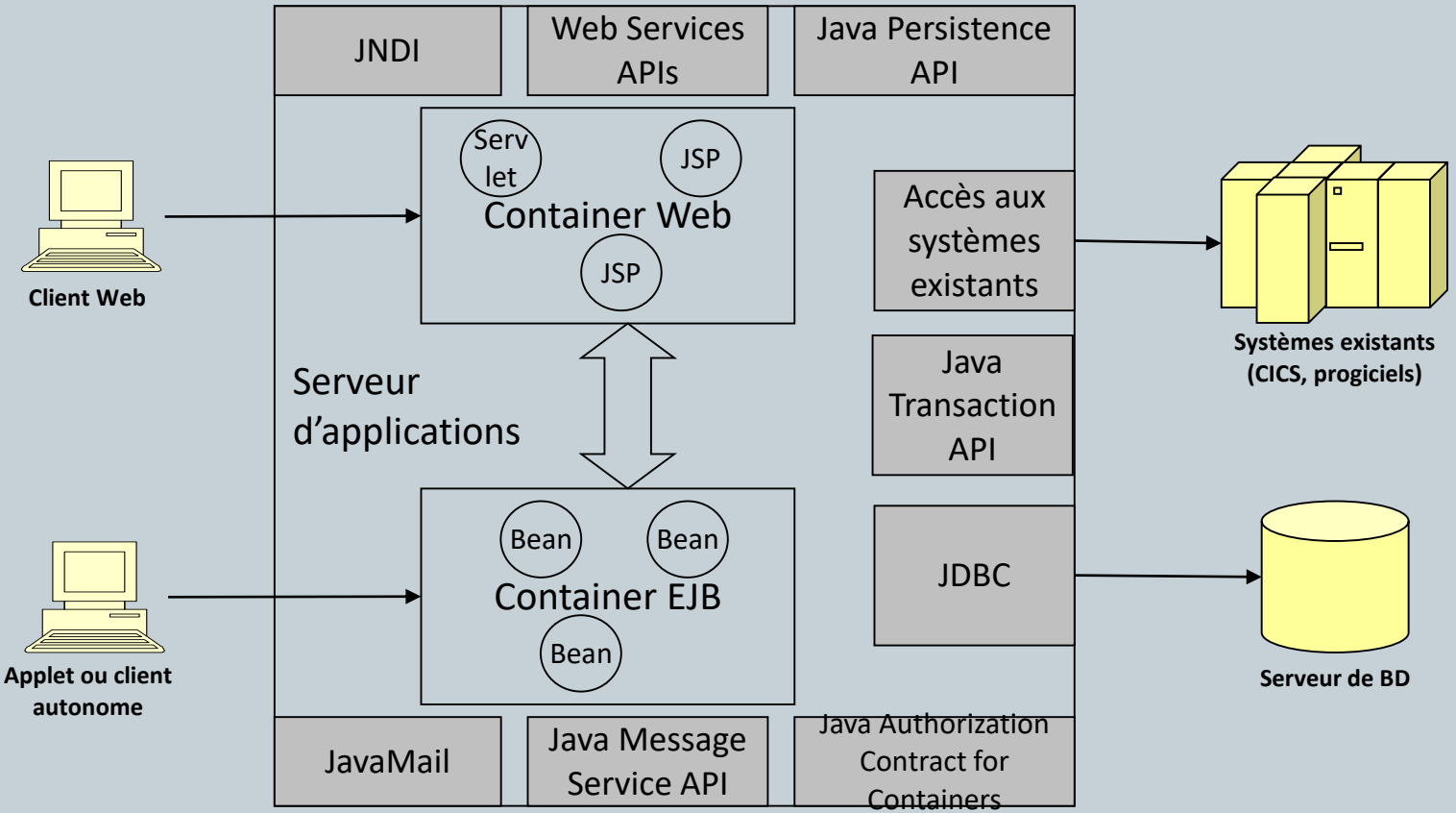
- Contrôleur
  - ActionServlet (contrôleur général) : intercepte les requêtes et les dispatche en fontion des URL (fichier struts-config.xml) vers les actions correspondantes
  - Actions (contrôleurs délégués) : gèrent la communication avec le modèle et renvoient les résultats à la vue
- Modèle
  - N'importe 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.

Java Enterprise Edition

- Caractéristiques
  - Plutôt une spécification qu'un framework
  - Intégrée aux serveurs d'applications Java
  - But : développement, déploiement et exécution des applications distribuées
- Mêmes principes de base que les infrastructures middleware…
  - Communication synchrone (RMI/IIOP) et asynchrone (JMS) entre objets distribués, serveurs de nom (JNDI), intégration d'objets CORBA (JavaIDL)…
- …plus de nombreux services techniques supplémentaires
  - Génération d'objets transactionnels distribués (EJB), gestion du cycle de vie des objets, gestion des transactions (JTA et JTS), sécurité, accès aux BD (JDBC), interfaces graphiques dynamiques (Servlets , JSP, JSF)…

Java Enterprise Edition

- L'architecture Java EE



Java Enterprise Edition

- Implémentations
  - Glassfish, WebLogic (Oracle)
  - WebSphere (IBM)
  - WildFly (RedHat)
  - Geronimo (Apache)
- Il existe aussi des implémentations partielles
  - Tomcat, TomEE (Apache)
  - Jetty (Eclipse)

Java Enterprise Edition

- Avantages
  - Très complet
  - Simplifié depuis la spécification EJB 3.0
- Inconvénient
  - Trop complet / lourd ?...

Java Platform, Enterprise Edition (Java EE)

- Une spécification pour frameworks
  - Page d'accueil, description, documentation, implémentations
  - Packages : ~~javax.~~ jakarta. …
- Contenu
  - Web
    - Servlet API (servlets, JSP, filtres…), Unified Expression Language (el)
    - Java Server Faces : couche de présentation « au-dessus » de JSP
  - Services Web
    - JAX-RS : une API pour services Restful
  - Entreprise
    - Enterprise JavaBeans (EJB) : beans transactionnels et distribués
  - Autres
    - Bas niveau : contexte, ressource, injection, annotations…
    - Haut niveau : messages, mail, persistance, transactions, sécurité…

Java API for RESTful Web Services (JAX-RS)

- Une partie de la spécification Java EE
  - Dédiée aux services REST
- Actuellement : JSR 370 (v. 2.1)
- Packages jakarta.ws.rs… de la spécification Jakarta EE
- Nécessite un contrôleur principal (~routeur)
  - Les contrôleurs délégués sont des POJOS annotés
- Implémentations
  - Apache CXF
  - Jersey
  - RestEasy
  - …

Java API for RESTful Web Services (JAX-RS)

- Définit un ensemble d'annotations
  - Chemin relatif de la ressource
    - @Path
  - Méthodes HTTP
    - @GET, @PUT, @POST, @DELETE, @HEAD
  - Négociation de contenus
    - @Produces
  - Types de contenus acceptés
    - @Consumes
  - Paramètres
    - @PathParam, @QueryParam, @MatrixParam, @HeaderParam, @CookieParam, @FormParam, @DefaultValue

Java API for RESTful Web Services (JAX-RS)

- Exemple

```
@Path("/notifications")
public class NotificationsResource {

    @GET
    @Path("/get/{id}")
    @Produces(MediaType.APPLICATION_JSON)
    public Response getNotification(@PathParam("id") int id) {
        return Response.ok()
          .entity(new Notification(id, "john", "test
notification"))
          .build();
    }
}
```

  - Source : Baeldung

Frameworks MVC : Spring

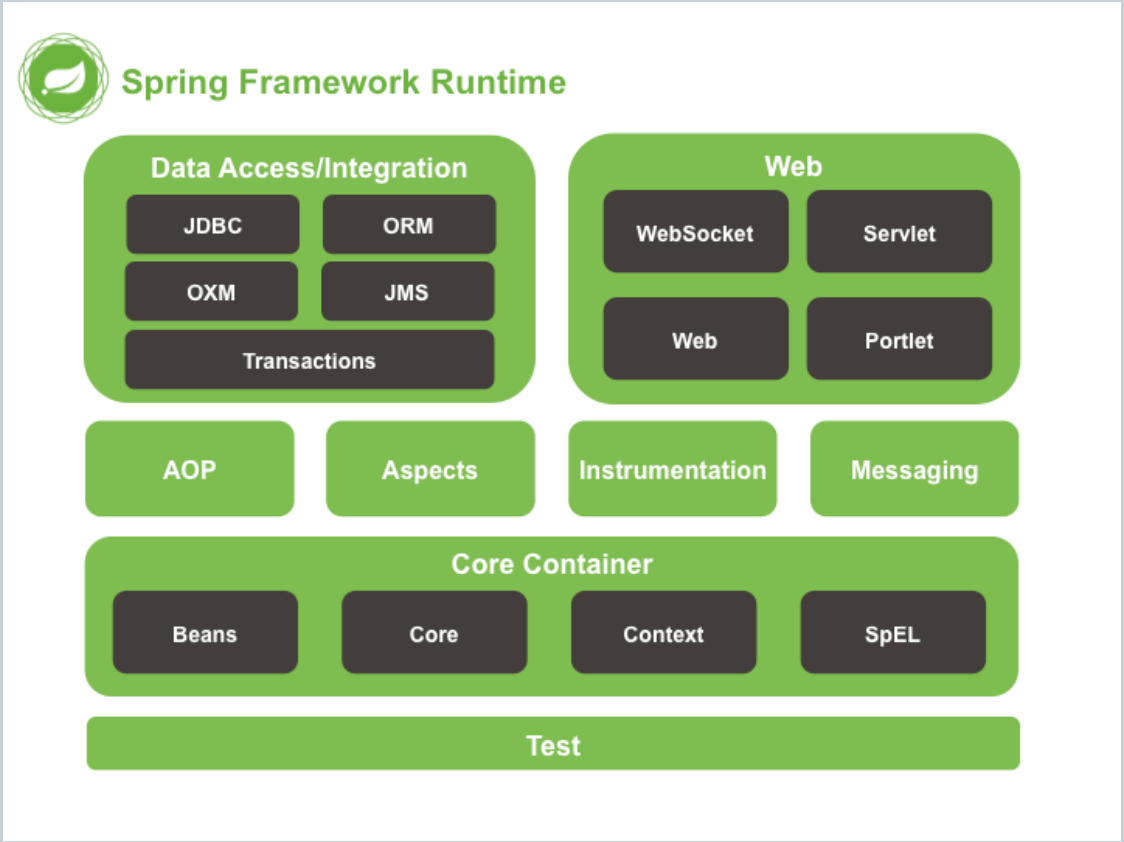
- Historique
  - Juin 2003 : sortie de la première version de Spring framework
  - 2004 : création de la société SpringSource par Rod Johnson
    - publication du livre“Expert One-on-One J2EE Design and Development” qui justifie la création de Spring
  - 2006 : sortie de la V. 2 de Spring
  - 2008 : rachat de Spring par VMWare
    - Sortie de la V. 3 du framework
    - Nombreux sous-projets : Spring Security, Spring Data, Spring AMQP…
- Version courante : 5.3.2
  - Doc : https://docs.spring.io/spring-framework/docs/current/reference/html/

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 : https://docs.spring.io/spring-framework/docs/4.3.x/spring-framework-reference/html/images/spring-overview.png

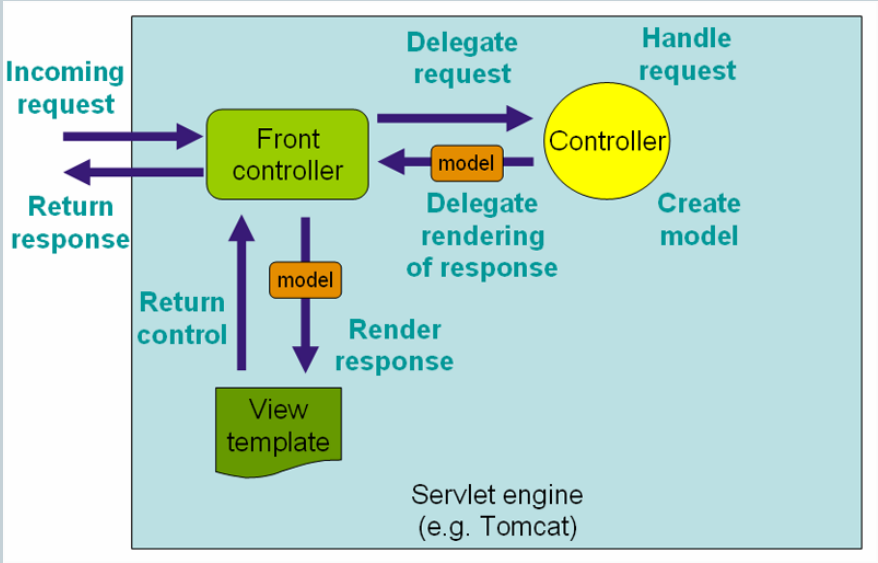
Frameworks MVC : Spring

- Spring Core container
  - 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 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
  - 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
  - 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>
```

    - 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
  - 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>
```

Spring RESTful Web Services (exemple)

- Exemple

```
@RestController
@RequestMapping("books-rest")
public class SimpleBookRestController {

    @GetMapping("/{id}", produces = "application/json")
    public Book getBook(@PathVariable int id) {
        return findBookById(id);
    }

    private Book findBookById(int id) {
        // ...
    }
}
```

  - Source : Baeldung

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

- Systèmes de gestion de contenus (CMS)
  - Vers de nouvelles architectures

	« Traditionnelle » (aka couplée)	Découplée	Headless
Côté client	-	• Front-End (templating) par défaut	-
Côté serveur	• Front-end (templating) • Back office • SGBD	• (Restful) API • Back office • SGBD	• (Restful) API • Back office • SGBD

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, 286 et 362), Liferay
    - PHP : PhpPortlet

Plan du cours

- Introduction
- Outils côté serveur
- Outils côté client
- Ressources externes
- Conclusion

Bibliothèques Web

- Différents types de bibliothèques
  - En fonction des finalités
    - Requêtage AJAX
      - Bibliothèques génériques (jQuery)
      - Clients spécifiques pour une Web API (éventuellement autogénérés)
    - Composants d'interface
    - Composants de programmation
      - Tri, organisation, ajout, transformation XSLT
  - …De plus en plus délaissées au profit des specs HTML5
    - implémentées dans les navigateurs
    - implémentées dans des polyfills
    - prises en charge par les outils de build (Babel)

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
      - Généralistes
        - jQuery ($), Undescore.js (_), Lodash (_)
      - AJAX
        - Axios…
      - …

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 côté client

- Souvent dédiés aux SPA
- Fonctionnalités proposées
  - Framework applicatif “classique”
    - Gestion des composants et de leurs dépendances
  - Fonctionnalités Web
    - Gestion des routes (SPA)
    - MV* (MVC, MVP, MVVM) ou Model-View-Whatever
    - Requêtage asynchrone
    - Templating
  - Spécificités du côté client
    - Pas d’ “application context”
    - → Utiliser les “browsing contexts” / stores (state management pattern)

Frameworks côté client

- Fonctionnalités Spécifiques au côté client
  - Sécurité
    - Authentification
  - Composants d'interface
    - Modale, menu, progress bar, Material…
  - APIs HTML5
    - Requêtage asynchrone, WebSockets
    - Progressive Web Apps
    - …
  - Sources externes
    - Maps / Leaflet, Youtube…

Frameworks côté client

- Liste de frameworks
  - …mêlant frameworks et bibliothèques :-/
- Annuaires de composants
  - https://bit.dev/
  - https://www.jqwidgets.com/
  - https://vuecomponents.com/
  - https://angular.io/resources
  - https://www.primefaces.org/primereact/
  - …

Autres outils côté client

- Outils de développement
  - IDEs « orientés-client »
- Outils de gestion de projet
  - Qualité de code
  - Test
- Outils de build
  - Compilation / transpilation
  - Packaging
- …

Plan du cours

- Introduction
- Outils côté serveur
- Outils côté client
- Ressources externes
- Infrastructure
- Conclusion

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 24 000+ APIs disponibles (décembre 2021)
    http://www.programmableweb.com/apis/directory

Plan du cours

- Introduction
- Outils côté serveur
- Outils côté client
- Ressources externes
- Conclusion

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

- Retour sur l'approche DevOps : vers un changement de mentalités
  - Une « philosophie culturelle » (Amazon)
    - une approche globale pour maîtriser toute la « stack », du code au déploiement
  - Des bonnes pratiques
    - En M1IF01 : dépôts de code / tests / intégration / déploiement…
    - Dans ce cours : partie infrastructure
    - Dans les 2 : modularité, composants, réutilisabilité
    - À venir : automatisation de la stack…
  - Des outils
    - À vous de vous les approprier…

Conclusion

- Tendances actuelles des technologies Web
  - Standards du W3C
    - Balises sémantiques (HTML5)
    - Mise en forme avancée (CSS Grid, Selectors…)
    - Contenus dynamiques (Canvas, audio, vidéo)
    - Interactions de haut niveau (Drag'n'drop, Web Components)
    - Prise en charge du matériel (Device APIs)
  - ECMAScript / JavaScript
    - Programmation structurée (classes ES6, TypeScript)
    - Programmation bas niveau (APIs HTML5)
    - Déport de code côté client (frameworks applicatifs)
    - Communication asynchrone / non bloquante (XHR, fetch, workers)
  - Performance
    - Mode déconnecté (Service workers, Progressive Web Apps)
    - Packaging d'applications (Web Packaging)
    - Push serveur (HTTP/2)

➔ M1IF13 ;-) TIW8

Retour sur la définition d' « application Web »

- Plusieurs visions
  - Données : mashup vs. top-down vs. bottom-up
  - Services : métier de l'application vs. génériques vs. API externes
  - Outils : frameworks / bibliothèques / Web APIs
  - Outils de développement : IDE / stack serveur / stack client
  - Distribuée : serveur / API / client
  - Couches : infra / application
  - Cycle de vie : conception / développement / déploiement / run / maintenance
  - …

➔ Il faut être capable de « projeter » votre Web app sur chacune de ces dimensions

Takeaway message

Le Web, c'est le truc sur lequel on trouve le plus de choses sur le Web

…mais c'est toujours plus facile quand on sait quelle question poser

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
      - CMS : https://medium.com/@sgourebi/architecture-d%C3%A9coupl%C3%A9e-ou-headless-avec-drupal-a6a10f723c5
        https://www.brightspot.com/solutions/decoupled-cms-and-headless-cms-platforms

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

Références

- Références utilisées pour ce cours
  - DevOps
    - https://en.wikipedia.org/wiki/DevOps
    - https://aws.amazon.com/devops/what-is-devops/
    - https://www.youtube.com/watch?v=_I94-tJlovg
    - https://www.atlassian.com/devops
    - https://www.softwaretestinghelp.com/top-5-software-configuration-management-tools/