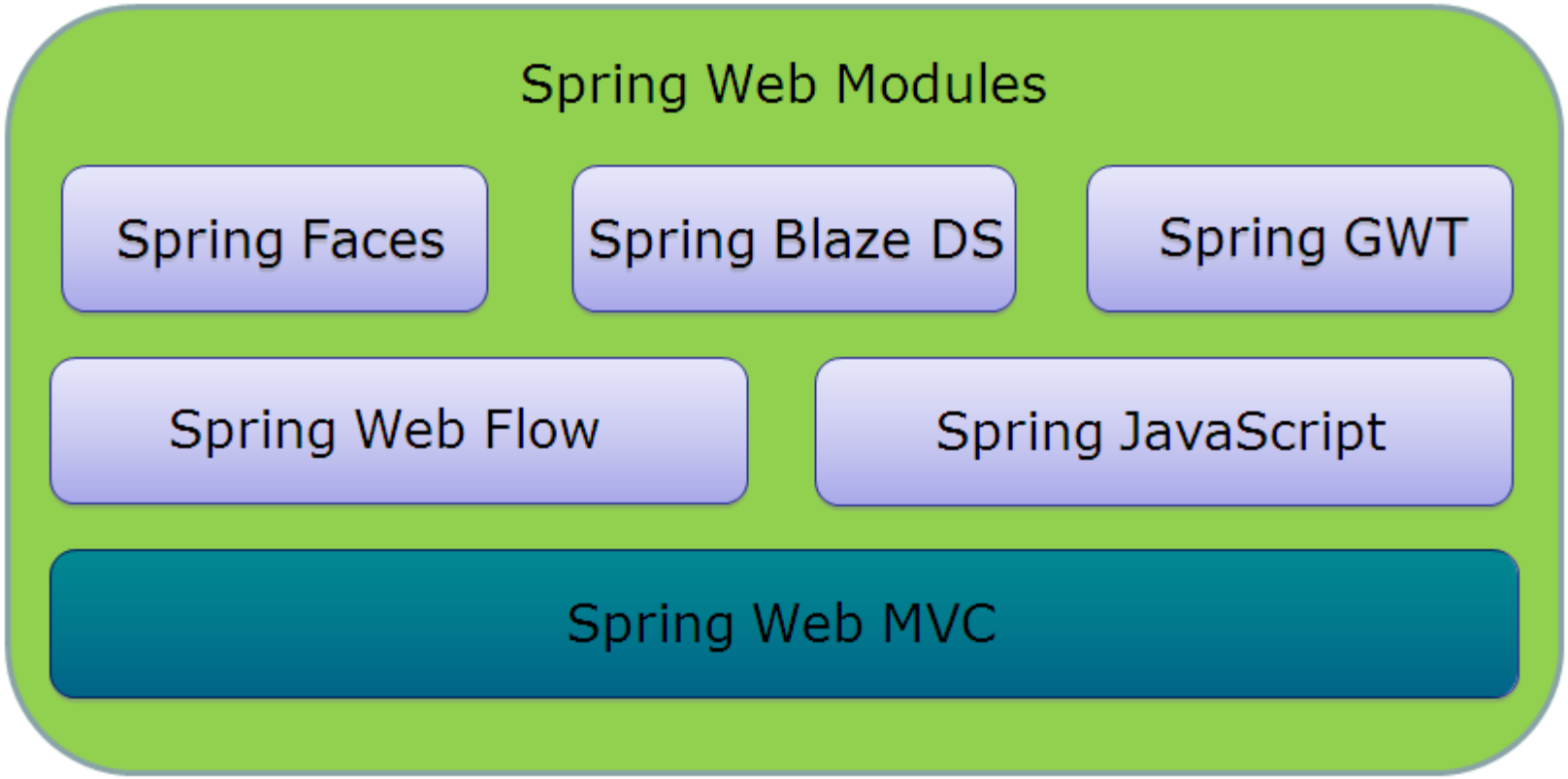
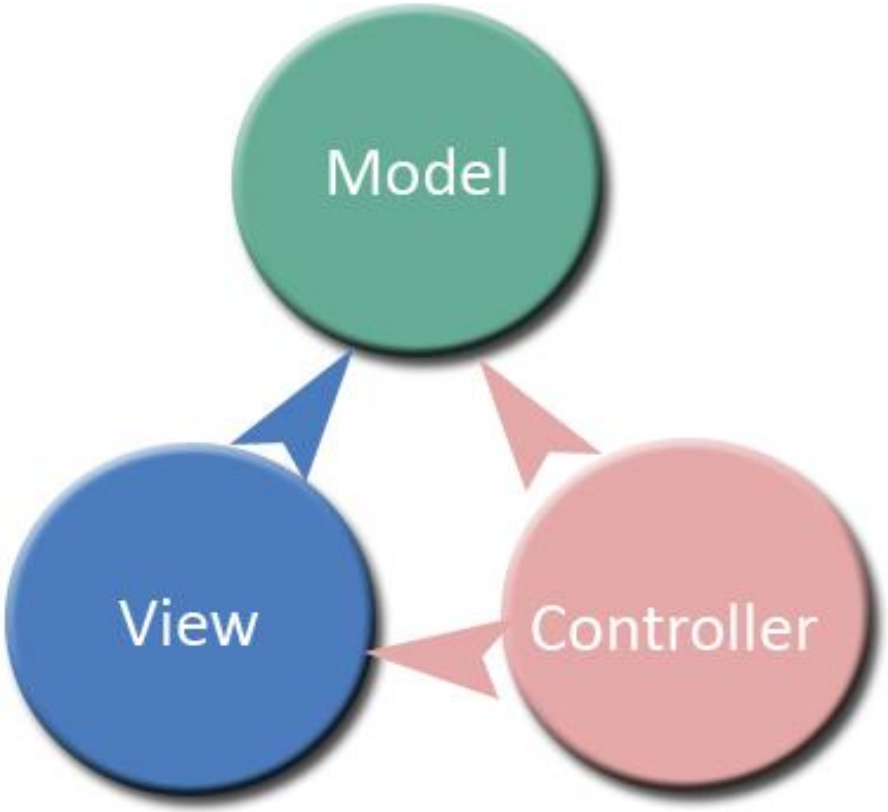
The Spring WEB stack



MVC

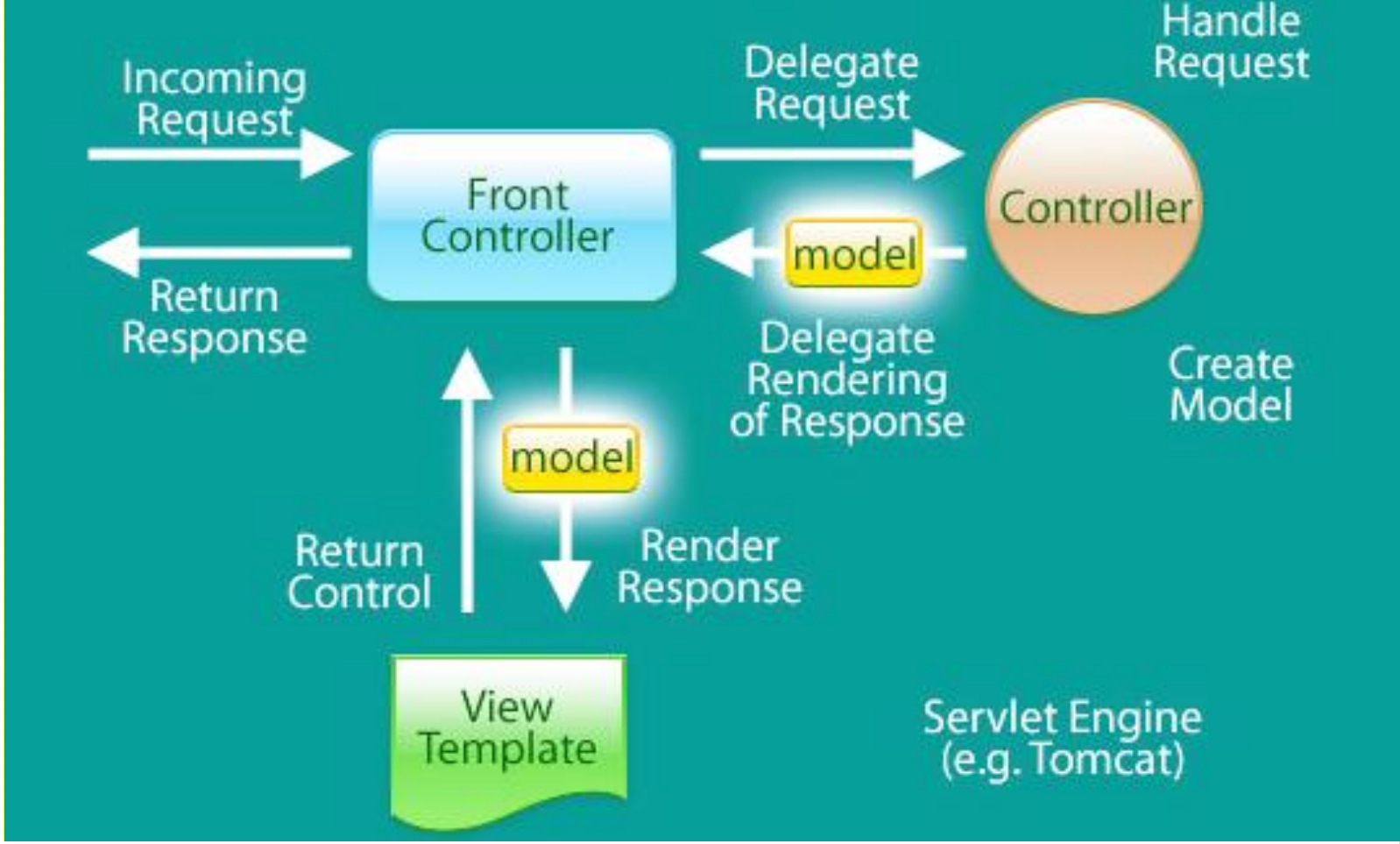


Web frameworks

- request-based
  - Struts, Spring MVC
- component-based
  - JSF, GWT, Wicket
- RIA
  - Flex

What is Spring MVC?

- web **component** of Spring Framework
- request based web framework



Request processing workflow

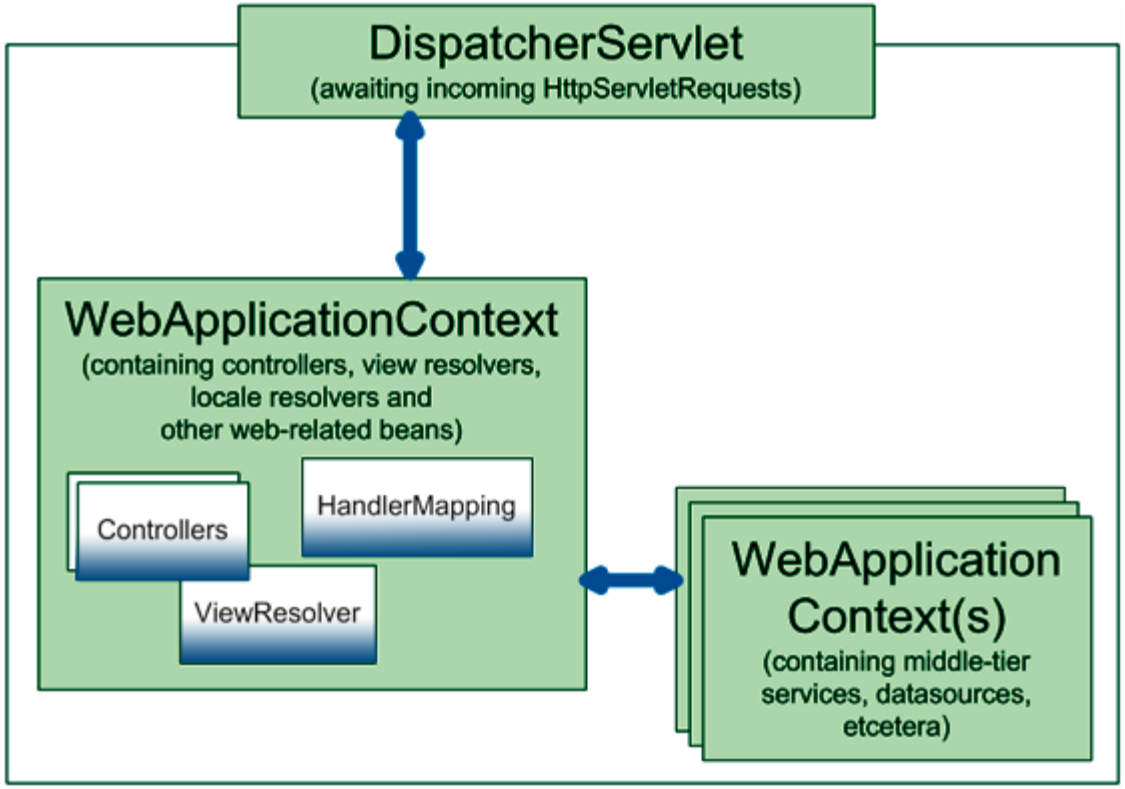
Front controller

```
<servlet>
  <servlet-name>Spring MVC Dispatcher Servlet</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  <init-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/web-config.xml</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
  <servlet-name>Spring MVC Dispatcher Servlet</servlet-name>
  <url-pattern>/app/*</url-pattern>
</servlet-mapping>
```

Application context

```
<web-app version="2.5">
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/root-context.xml</param-value>
  </context-param>

  <listener>
    <listener-class>
      org.springframework.web.context.ContextLoaderListener
    </listener-class>
  </listener>
</web-app>
```

Context


DispatcherServlet
(awaiting incoming HttpServletRequests)
WebApplicationContext
(containing controllers, view resolvers,
locale resolvers and
other web-related beans)
Controllers
HandlerMapping
ViewResolver
WebApplication
Context(s)
(containing middle-tier
services, datasources,
etcetera)

UrlRewrite

http://www.tuckey.org/urlrewrite/

```
<filter>
  <filter-name>UrlRewriteFilter</filter-name>
  <filter-class>org.tuckey.web.filters.urlrewrite.UrlRewriteFilter</filter-class>
</filter>
<filter-mapping>
  <filter-name>UrlRewriteFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

Mapping

```
<urlrewrite default-match-type="wildcard">
  <rule>
    <from>/</from>
    <to>/app/welcome</to>
  </rule>
  <rule>
    <from>/**</from>
    <to>/app/$1</to>
  </rule>
  <outbound-rule>
    <from>/app/**</from>
    <to>/$1</to>
  </outbound-rule>
</urlrewrite>
```

WebApplicationContextUtils(1)

```
public class ExchangerServlet extends HttpServlet {

  private AccountService accountService;

  @Override
  public void init() throws ServletException {
    ServletContext sc = super.getServletContext();
    ApplicationContext context =
        WebApplicationContextUtils.getWebApplicationContext(sc);

    accountService = context.getBean(AccountService.class);
  }
}
```

WebApplicationContextUtils(2)

```
public class ExchangerFilter implements Filter {
  private AccountService accountService;

  @Override
  public void init(FilterConfig config) throws ServletException {
    ServletContext sc = config.getServletContext();
    ApplicationContext context =
        WebApplicationContextUtils.getWebApplicationContext(sc);

    accountService = context.getBean(AccountService.class);
  }
}
```

Controller

```
@Controller
@RequestMapping
@RequestParam
@PathVariable
```

Mapping requests

- by path

```
@RequestMapping("/welcome")
```

- by HTTP method

```
@RequestMapping(value = "/welcome", method=RequestMethod.GET)
```

- by presence / value of query parameter

```
@RequestMapping(params = {"find=ByMake", "form" })
```

- by presence / value of request header

```
@RequestMapping(value = "/welcome", headers="accept=text/*")
```

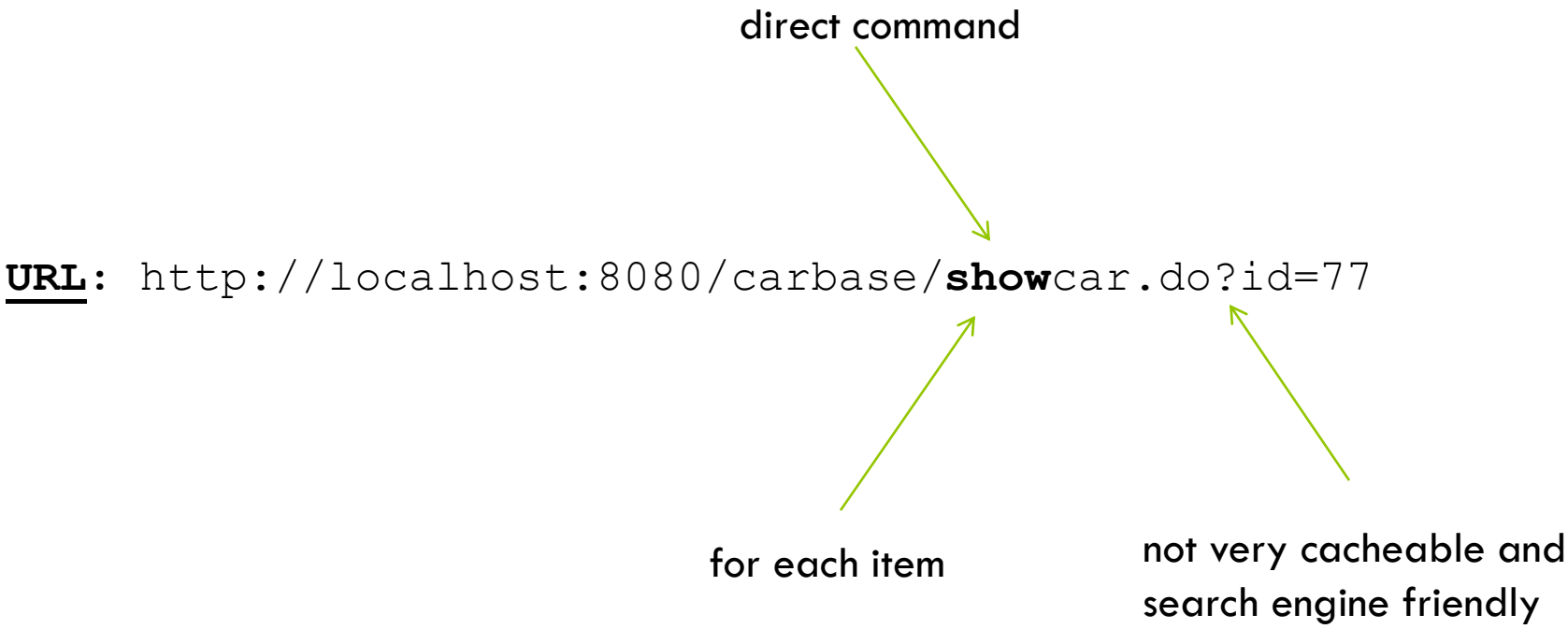
Simple Controller

```
@Controller
public class WelcomeController {

   @RequestMapping("/welcome")
   public void welcome() {
   }
}
```

Use case controller

```
@Controller
public class CarController {
  @RequestMapping("/showcar.do")
  public String show(@RequestParam("id") id, Model model) {
    model.addAttribute("car", Car.findCar(id));
    return "jsp/cars/car.jsp";
  }

  @RequestMapping("/carlist.do")
  public String list(Model model) { /** such business logic*/}

  /** such method handlers*/
}
```

Unfriendly URLs



REST

- **R**epresentation **S**tate **T**ransfer
- style of software architecture
- RPC is antipode

- http://en.wikipedia.org/wiki/Representational_State_Transfer

Http methods

- get
- post
  - when the resource URL is unknown (create item)
- put
  - when the resource URL is known (update item)
- delete

- post vs put
  - http://stackoverflow.com/questions/630453/put-vs-post-in-rest

RESTful mapping

Resource	GET	PUT	POST	DELETE
http://domain.com/cars	obtain list of item	update	create	X
http://domain.com/cars/7	obtain item	X	X	delete
http://domain.com/cars?form	create empty form	X	X	X
http://domain.com/cars/7?form	pre-populated form	X	X	X

RESTful URLs

URL: `http://localhost:8080/carbase/cars/11`

Typical actions

- simple list page
- filtered list page
- CRUD:
  - create
  - read (retrieve)
  - update
  - delete
- workflow
  - submit / approve / etc.

List page

URL: http://localhost:8080/carbase/cars

```
@Controller
@RequestMapping("/cars")
public class CarController {

  @RequestMapping(method = RequestMethod.GET)
  public String list(Model model) {
    model.addAttribute("cars", Car.findAllCars());
    return "cars/list";
  }
}
```

Detailed page

URL: http://localhost:8080/carbase/cars/**11**

```
@Controller
@RequestMapping("/cars")
public class CarController {

  @RequestMapping(value = "/{id}", method = RequestMethod.GET)
  public String show(@PathVariable("id") Long id, Model model) {
    model.addAttribute("car", Car.findCar(id));
    return "cars/show";
  }
}
```

Create

URL: http://localhost:8080/carbase/cars

```
@Controller
@RequestMapping("/cars")
public class CarController {

  @RequestMapping(method = RequestMethod.POST)
  public String create(Car car) {
    car.persist();
    return "redirect:/cars/" + car.getId();
  }
}
```

Update

URL: http://localhost:8080/carbase/cars/

```
@Controller
@RequestMapping("/cars")
public class CarController {

  @RequestMapping(method = RequestMethod.PUT)
  public String update(@Valid Car car, BindingResult result) {

    /** Spring Validator*/
    //result.hasErrors();
    car.merge();
    return "redirect:/cars/" + car.getId();
  }
}
```

Delete

URL: http://localhost:8080/carbase/cars/**11**

```
@Controller
@RequestMapping("/cars")
public class CarController {

  @RequestMapping(value = "/{id}", method = RequestMethod.DELETE)
  public String delete(@PathVariable("id") Long id) {
    Car.findCar(id).remove();
    return "redirect:/cars;
  }
}
```

Filtered page

URL: http://localhost:8080/carbase/cars

```
@Controller
@RequestMapping("/cars")
public class CarController {

  @RequestMapping(params="find=ByMake", method=RequestMethod.GET)
  public String findByMake(@RequestParam("make")Make make, Model m) {
    m.addAttribute("cars", Car.findCarsByMake(make).getResultList());
    return "cars/list";
  }
}
```

Delete and put through post

- Spring tag

```
<form:form action="/carbase/cars" method="PUT">
```

- html

```
<form id="car" action="/carbase/cars" method="post">
<input type="hidden" name="_method" value="PUT"/>
```

- server side

```
<filter>
  <filter-name>HttpMethodFilter</filter-name>
  <filter-class>
    org.springframework.web.filter.HiddenHttpMethodFilter
  </filter-class>
</filter>
```

Handler arguments

- HttpSession / HttpServletRequest / etc.
- Spring's WebRequest / NativeWebRequest
- path variable
- java.io.InputStream / java.io.OutputStream
- request's param / header / body / cookies
- command objects

- http://static.springsource.org/spring/docs/3.0.x/spring-framework-reference/html/mvc.html#mvc-ann-requestmapping-arguments

Return types

- ModelAndView
- Model / Map / ModelMap
- View
- String / void
- `@ResponseBody` / `@ModelAttribute`

- http://static.springsource.org/spring/docs/3.0.x/spring-framework-reference/html/mvc.html#mvc-ann-requestmapping-arguments

Additional annotations

- `@ModelAttribute`
- `@SessionAttributes`
- `@RequestHeader`
- `@CookieValue`
- `@RequestBody / @ResponseBody`

ModelAttribute

- maps a model attribute to the specific parameter

```
@RequestMapping(method = RequestMethod.POST)
public String create(@ModelAttribute("car") Car car) {}
```

- provide reference data for the model

```
@ModelAttribute("makes")
public Collection<Make> populateMakes() {
  return Make.findAllMakes();
}
```

SessionAttributes

- list the names or types of model attributes which should be stored in the session

```
@Controller
@SessionAttributes("car")//@SessionAttributes(value={}, types={})
public class CarController {
  public String updateForm(@PathVariable("id") Long id, Model model) {
    m.addAttribute("car", Car.findCar(id));
  }

  public String update(Car request, SessionStatus status) {
    status.setComplete();
  }
}
```

RequestHeader

- typical request header

host = localhost:8080

user-agent = Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.2.13) Gecko/20101203 Firefox/3.6.13 (.NET CLR 3.5.30729; .NET4.0E)

accept = text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

accept-language = en-us,en;q=0.5

- obtain request header

```
@RequestMapping("/welcome")
public void welcome(@RequestHeader("user-agent") String agent) {}
```

- narrow mappings

```
@RequestMapping(value = "/welcome", headers="accept=text/*")
public void welcome() {}
```

CookieValue

- get the JSESSIONID of the cookie

```
@RequestMapping(value = "/welcome")
public void welcome(@CookieValue("JSESSIONID") String session){
}
```

Data Representation

Approach

- template view
  - ViewResolver, View
  - HTML, Excel, PDF, etc.
- data view
  - HttpMessageConverter
  - XML, JSON, etc.

View resolver

- XmlViewResolver
- ResourceBundleViewResolver
- UrlBasedViewResolver
- InternalResourceViewResolver
- BeanNameViewResolver
- ContentNegotiatingViewResolver

- http://static.springsource.org/spring/docs/3.0.x/spring-framework-reference/html/mvc.html#mvc-viewresolver-resolver

View

- JSP & JSTL
- Tiles
- Velocity
- FreeMarker
- etc.
- prefix
  - redirect:
  - forward:

ResourceBundleViewResolver

- configuration

```
<bean class="org.springframework.web.servlet.view.ResourceBundleViewResolver">
  <property name="basename" value="view"/>
</bean>
```

- view.properties

```
welcome.(class)=org.springframework.web.servlet.view.JstlView
welcome.url=/WEB-INF/jsp/welcome.jsp

cars.(class)=org.springframework.web.servlet.view.JstlView
cars.url=/WEB-INF/jsp/cars.jsp
```

- controller

```
@Controller return "cars"
```

UrlBasedViewResolver

```
<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>
```

- use case

```
If @Controller return "cars/show"
view class will process "/WEB-INF/jsp/cars/show.jsp"
```

Tiles(1)

- configuration

```
<bean class="org.springframework.web.servlet.view.UrlBasedViewResolver">
  <property name="viewClass" value="org.springframework.web.servlet.view.tiles2.TilesView"/>
</bean>

<bean id="tilesConfigurer"
      class="org.springframework.web.servlet.view.tiles2.TilesConfigurer">
  <property name="definitions">
    <list>
      <value>/WEB-INF/layouts/layouts.xml</value>
      <value>/WEB-INF/**/views.xml</value>
    </list>
  </property>
</bean>
```

- views.xml

```
<definition extends="default" name="cars/show"></definition>
```

Tiles(2)

```
<bean id="tilesConfigurer"
      class="org.springframework.web.servlet.view.tiles2.TilesConfigurer">
  <property name="definitions">
    <list>
      <value>/WEB-INF/layouts/layouts.xml</value>
      <value>/WEB-INF/**/views.xml</value>
    </list>
  </property>
  <property name="preparerFactoryClass">
    <value>
      org.springframework.web.servlet.view.tiles2.SpringBeanPreparerFactory
    </value>
  </property>
</bean>
```

HttpMessageConverter

- reads the request body and writes the response
- converters mapped to content types

- http://static.springsource.org/spring/docs/3.0.x/spring-framework-reference/html/remoting.html#rest-message-conversion

- registered by default if jar present in classpath
  - Jackson, JAXB, Atom, RSS

ContentNegotiatingViewResolver

Strategies

- URI
  - www.domain.com/cars.**html**
  - www.domain.com/cars.**json**
- content negotiation
  - Accept: text/html…
  - Accept: text/xml…
  - Accept: application/pdf…

Example(1)

```
<bean class="org.springframework.web.servlet.view.ContentNegotiatingViewResolver">
  <property name="mediaTypes">
    <map>
      <entry key="atom" value="application/atom+xml"/>
      <entry key="html" value="text/html"/>
      <entry key="json" value="application/json"/>
    </map>
  </property>
  <property name="viewResolvers">
    <list>
      <bean class="org.springframework.web.servlet.view.BeanNameViewResolver"/>
      <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver"/>
    </list>
  </property>
  <property name="defaultViews">
   <list><bean class="org.springframework.web.servlet.view.json.MappingJacksonJsonView"/></list>
  </property>
</bean>
```

Example(2)

```
<bean class="org.springframework.web.servlet.view.ContentNegotiatingViewResolver">
  <property name="order" value="1" />
  <property name="mediaTypes">
    <map>
      <entry key="json" value="application/json"/>
      <entry key="xml" value="application/xml" />
    </map>
  </property>
  <property name="defaultViews">
    <list><bean class="org.springframework.web.servlet.view.json.MappingJacksonJsonView"/></list>
  </property>
</bean>
<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
  <property name="order" value="2" />
  <property name="prefix" value="/WEB-INF/views/"/>
  <property name="suffix" value=".jsp"/>
</bean>
```

Additional features

- Locales
- Themes
- File upload
- Handling mappings / exceptions

Locales

- LocaleResolver
  - AcceptHeaderLocaleResolver
  - CookieLocaleResolver
  - SessionLocaleResolver
- LocaleChangeInterceptor

```
<bean id="localeChangeInterceptor"
      class="org.springframework.web.servlet.i18n.LocaleChangeInterceptor">
      <property name="paramName" value="lang"/>
</bean>
```

http://localhost:8080/carbase/?**lang=ru**

Themes

- ThemeSource
  - ResourceBundleThemeSource
- ThemeResolver
  - FixedThemeResolver
  - SessionThemeResolver
  - CookieThemeResolver

File upload

- MultipartResolver

```
<bean id="multipartResolver"
      class="org.springframework.web.multipart.commons.CommonsMultipartResolver">

  <!-- one of the properties; the maximum file size in bytes -->
  <property name="maxUploadSize" value="100000"/>
</bean>
```

- MultipartFile

```
@RequestMapping(method = RequestMethod.POST)
public String upload (@RequestParam("file") MultipartFile file) {
}
```

Handling mappings

- interceptors
- default handler
- order
- lazy init handlers

```
<bean id="handlerMapping"
class="org.springframework.web.servlet.mvc.annotation.DefaultAnnotationHandlerMapping">
  <property name="interceptors">
    <bean class="myInterceptor"/>
  </property>
</bean>
```

Handler interceptors

```
public interface HandlerInterceptor {

  /** Called before HandlerAdapter invokes the handler.
   * @return true if the execution chain should proceed */
  boolean preHandle(Request, Response, Handler) {}

  /** Called after HandlerAdapter actually invoked the handler,
   * but before the DispatcherServlet renders the view.*/
  void postHandle(Request, Response, Handler, ModelAndView) {}

  /** Callback after rendering the view. */
  void afterCompletion(Request, Response, Handler, Exception) {}
}
```

Handling Exceptions

- application
  - HandlingExeptionResolver
- controller
  - ```
    @ExceptionHandler(Exception.class)
    public String handleException(Exception e) {
      return ClassUtils.getShortName(e.getClass());
    }
    ```
- method
  - ```
    try {} catch (Exception e) {}
    ```

Magic tags

```
<mvc:annotation-driven>
<mvc:interceptors>
<mvc:view-controller>
<mvc:resources>
<mvc:default-servlet-handler>
```

mvc:annotation-driven

- registers necessary beans
- support formatting
  - Number fields using the `@NumberFormat`
  - Date, Calendar, Long fields using the `@DateTimeFormat`
- support for reading and writing
  - XML, if JAXB is present in classpath
  - JSON, if Jackson is present in classpath
- support validating with `@Valid`

mvc:interceptors

```
<!-- register "global" interceptor beans to apply to all
registered HandlerMappings -->
<mvc:interceptors>

  <!-- applied to all URL paths -->
  <bean class="org.springframework.web.servlet.theme.ThemeChangeInterceptor"/>

  <!-- applied to a specific URL path -->
  <mvc:interceptor>
    <mvc:mapping path="/secure/*"/>
    <bean class="org.example.MyInterceptor" />
  </mvc:interceptor>
</mvc:interceptors>
```

mvc:view-controller

- immediately forwards to a view when invoked

```
<mvc:view-controller path="/" view-name="index"/>

<mvc:view-controller path="/resourceNotFound"/>
```

mvc:resources

```
<!-- Handles HTTP GET requests for /resources/** by efficiently
serving up static resources -->
<mvc:resources location="/, classpath:/META-INF/web-resources/"
               mapping="/resources/**"/>
```

- http://static.springsource.org/spring/docs/3.0.x/spring-framework-reference/html/mvc.html#mvc-static-resources

mvc:default-servlet-handler

```
<!-- Allows for mapping the DispatcherServlet to "/" by
forwarding static resource requests to the container's default
Servlet -->
<mvc:default-servlet-handler/>
```

Ajax

Getting JSON

- server

```
@RequestMapping(value="/availability", method=RequestMethod.GET)
public @ResponseBody AvailabilityStatus
                        getAvailability(@RequestParam String name) {
  return AvailabilityStatus.AVAILABLE;
}
```

- client

```
function checkAvailability() {
  $.getJSON("account/availability", {name: $('#name').val()},
    function(availability) {}
  );
}
```

Post JSON

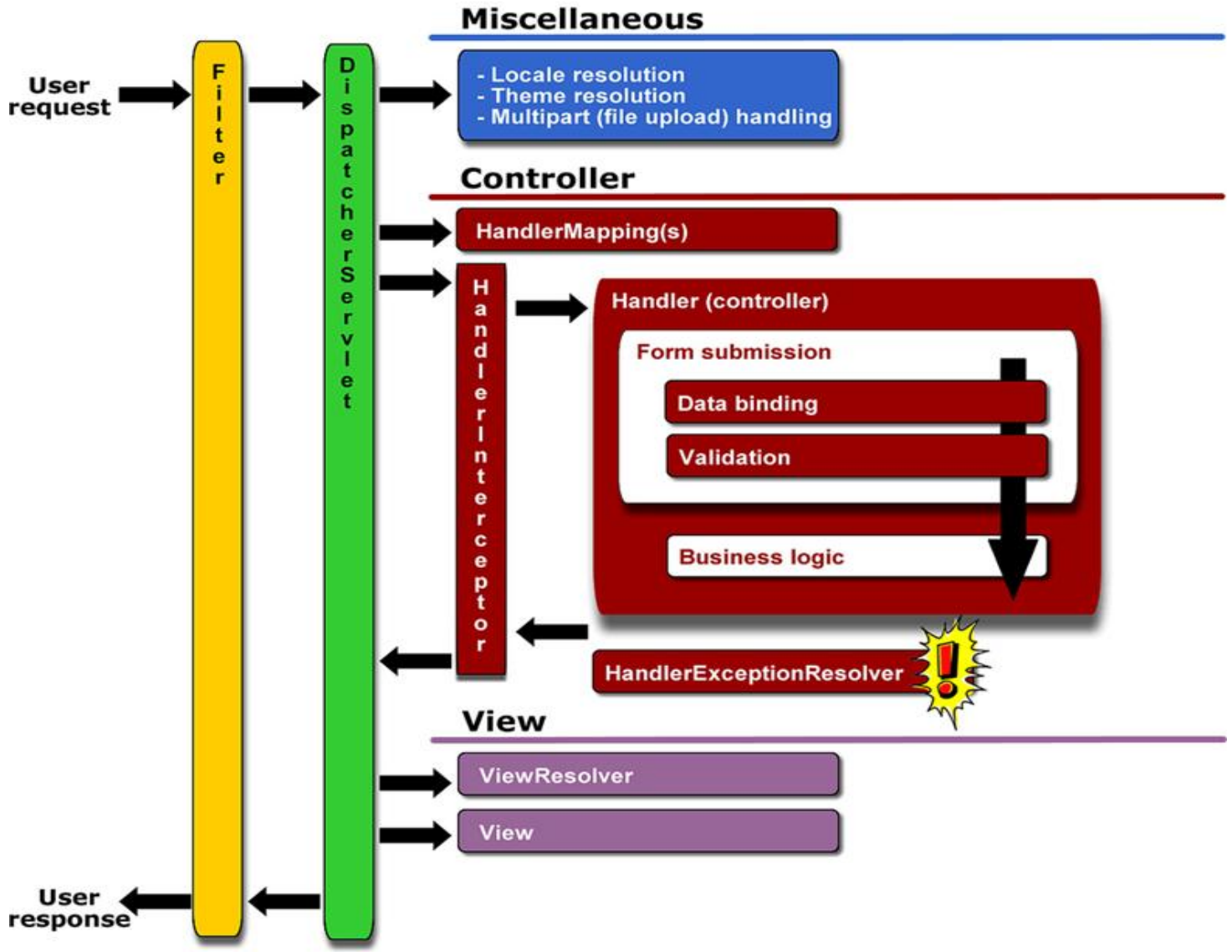
- client

```
$("#account").submit(function() {
  var account = $(this).serializeObject();
  $.postJSON("account", account, function(data) {
    $("#assignedId").val(data.id);
  });
  return false;
});
```

- server

```
@RequestMapping(method=RequestMethod.POST)
public @ResponseBody Map<String, ? extends Object>
                                    create(@RequestBody Account account) {
  return Collections.singletonMap("id", account.getId());
}
```

Spring MVC Request Lifecycle



Features

- clear separation of roles
- reusable business code
- flexible model transfer
- customizable binding and validation
- customizable handler mapping and view resolution
- pluggability

Spring MVC

- lightweight web framework
- controller is a Spring bean

Information

- reference
  - http://www.springsource.org/documentation
- samples
  - https://src.springsource.org/svn/spring-samples/
- blog
  - http://blog.springsource.com/category/web/
- forum
  - http://forum.springsource.org/forumdisplay.php?f=25

Questions

The end

noskov.d@Gmail.com

my LinkedIn profile

http://www.linkedin.com/in/noskovd

slideshare

http://www.slideshare.net/analizator/presentations