Ćwiczenie 1:

Zastąp istniejącą konfigurację opartą o adnotację, konfiguracją XML. Popraw poniższy komponent umożliwiając wstrzykiwanie zależności w oparciu o konstruktor oraz przygotuj odpowiednią konfigurację w pliku XML. Załóż, że zależności już istnieją w kontekście oraz występują pod taką samą nazwą jak w poniżej zaprezentowanej klasie.

```
@Component
public class TasksService {

	@Resource
	MessageSource messages;

	@Autowired
	SqlMapClient sqlMapClient;

	@Autowired
	Validator validator;

	//..

}
```

Instrukcja rozwiązania

1. Usuń wszystkie adnotację z klasy
2. Dodaj odpowiedni konstruktor, używający pól klasy
3. W pliku konfiguracyjnym, dodaj konfigurację komponentu, wykorzystując znacznik <bean /> oraz <constructor-arg />

Rozwiązanie

```
public class TasksService {

	MessageSource messages;

	SqlMapClient sqlMapClient;

	Validator validator;

	public TasksService(MessageSource messages,
			SqlMapClient sqlMapClient,
			Validator validator) {
		this.messages = messages;
		this.sqlMapClient = sqlMapClient;
		this.validator = validator;
	}

	//..

}
```

```
<bean id="taskService" class="pl.devcastzone.spring.todo.TasksService">
	<constructor-arg ref="validator" />
	<constructor-arg ref="messageSource"/>
	<constructor-arg ref="sqlMapClient" />
</bean>
```

Ćwiczenie 2:

Dla powyższego przykładu, zamiast konfiguracji XML, zastosuj konfigurację JavaConfig. Załóż, że wszystkie zależności zostały już utworzone w pliku konfiguracyjnym `app-context.xml`.

Instrukcja rozwiązania

1. Utwórz plik konfiguracyjny JavaConfig

Rozwiązanie

```
@Configuration
@ImportResource("classpath:META-INF/spring/app-context.xml")
public class AppConfiguration {

	@Autowired
	MessageSource messages;

	@Autowired
	SqlMapClient sqlMapClient;

	@Autowired
	Validator validator;

	@Bean
	public TasksService tasksService() {
		return new TasksService(messages, sqlMapClient, validator);
	}

	//..

}
```

Ćwiczenie 3:

Zastąp automatyczną konfigurację Spring MVC, konfiguracją XML. Kontroler oraz aktualna konfiguracja opisane są poniżej

```
@Controller
public class HomeController {


	@RequestMapping(value = "/", method = RequestMethod.GET)
	public String home(@RequestParam("name") String name, Model model) {

		model.addAttribute("serverTime", System.currentTimeMillis());

		return "home";
	}

}
```

```
<mvc:annotation-driven />
<context:component-scan base-package="pl.devcastzone.todo" />

<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
	<property name="prefix" value="/WEB-INF/views/" />
	<property name="suffix" value=".jsp" />
</bean>
```

Instrukcja rozwiązania

1. Z klasy kontrole usuń adnotacje odpowiadające automatycznej konfiguracji.
2. Zmodyfikuj kontroler aby rozszerzał klasę `AbstractController`, a całą logikę przenieś do metody `handlerRequestInternal()`
3. Dodaj do pliku konfiguracyjnego odpowiednią implementacją komponentów `HandlerAdapter` (`SimpleControllerHandlerAdapter`) oraz `HandlerMapper` (`BeanNameUrlHandlerMapper`)
4. Dodaj definicję kontrolera, wykorzystując atrybut *name* znacznika *<bean />*

Rozwiązanie

```
public class HomeController extends AbstractController {

	protected ModelAndView handleRequestInternal(HttpServletRequest request,
			HttpServletResponse response) throws Exception {
		ModelAndView model = new ModelAndView("home");
		model.getModel().put("serverTime", System.currentTimeMillis());

		return model;
	}

}
```

```
<bean class="org.springframework.web.servlet.mvc.SimpleControllerHandlerAdapter" />
<bean class="org.springframework.web.servlet.handler.BeanNameUrlHandlerMapping" />
```

```
<bean name="/tasks"
	class="pl.devcastzone.HomeController" />
```