**ГЛАВА 1. ОБЗОР ИНСТРУМЕНТОВ РАЗРАБОТКИ**

**1.1. Bootstrap**

Bootstrap представляет собой свободный фреймворк интерфейсов для быстрой и простой веб-разработки. Он предоставляет шаблоны дизайна, основанные на HTML и CSS для разметки, форм, кнопок, таблиц, навигации, диалоговых окон, каруселей изображений и многого другого, а также дополнительные плагины JavaScript. На основе Bootstrap можно легко создавать сайты с «отзывчивым дизайном» – сайты, которые одинаково хорошо выглядят на всех типах устройств, от небольших телефонов до настольных компьютеров.

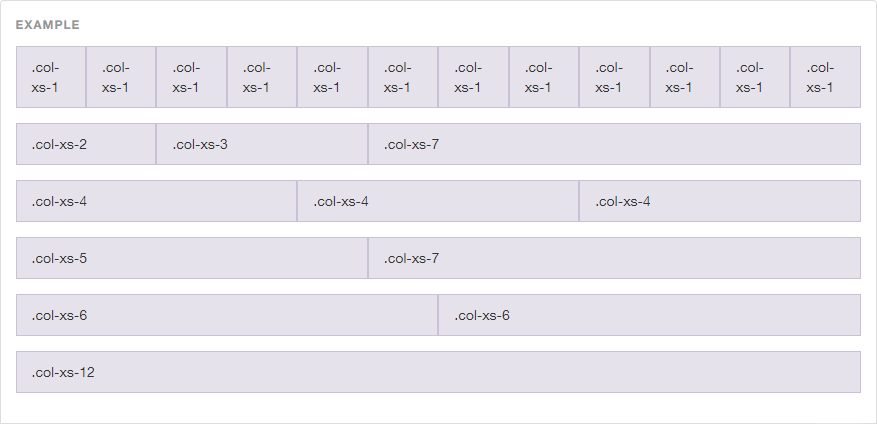
Данный фреймворк поддерживает самые последние версии популярных браузеров. Однако, в разных браузерах отображаемые элементы не всегда выглядят одинаково. Старые версии Internet Explorer (IE) не поддерживают некоторых свойств CSS3 и HTML5, используемых фреймворком, поэтому следует учесть это, обеспечивая поддержку этих браузеров.

Классический макет страницы включает в себя шапку сайта, основной контент, боковую колонку и подвал сайта. Верстка подобной страницы требует расчета ширины каждого элемента. Шапка с подвалом в большинстве случаев принимает значение ширины 100% относительно размера экрана. Основная часть контента и боковая колонка могут иметь размеры 70/30 или 85/25, а при уменьшении размеров до мобильной версии и вовсе могут растягиваться на весь экран.

Вот для таких целей и нужна сетка Bootstrap. Сетка состоит из групп рядов и колонок, расположенных внутри одного или нескольких контейнеров. Она функционирует как таблица, в которой есть свои ряды и столбцы, максимальное количество столбцов 12.

Сетку можно делать внутри другой сетки сколько угодно. Если делать все блоки сайта с использованием сетки, то самостоятельно писать медиа запросы для их адаптивности вообще не придется.

Помимо сетки существует огромное количество всевозможных компонентов: навигационные меню, формы, таблицы, модальные окна, вкладки, оповещения, всплывающие подсказки и тому подобное.



Преимущества фреймворка заключаются в высокой скорости разработки макетов страниц сайта, так как Bootstrap содержит огромный набор готовых решений и элементов. Все элементы фреймворка адаптивны под любые устройства и корректно отображаются во всех современных браузерах, что обеспечивает кроссбраузерность и адаптивность сайта. Фреймворк также легок в использовании и позволяет даже человеку с базовыми знаниями о HTML и CSS свободно создавать веб-страниц с его использованием. У Bootstrap очень хорошая документация с большим количеством примеров готового кода.

Bootstrap содержит так же ряд недостатков. Во-первых, большое количество информации, которое содержится в предлагаемых шаблонах. При самостоятельном создании макета, разработчик использует только необходимый минимум для реализации задуманных идей. Шаблон должен учитывать множество других вариантов, то есть быть универсальным. Решить эту проблему можно, контролируя компоненты, которые будут добавлены в css-файл. Второй недостаток заключается в шаблонности дизайна. Часто сайты, созданные с помощью фреймворка, похожи один на другой: одинаковая структура, кнопки, навигационные элементы. Теряется уникальность дизайна, что не всегда уместно. Этот недостаток тоже можно устранит, минимизировав использование готовый решений и изменяя шаблон в соответствии с индивидуальным решением дизайнера и запросов заказчика.

Bootstrap значительно упрощает процесс создания кроссбраузерного адаптивного сайта с привлекательным дизайном и набором JavaScript эффектов. Однако для получения максимума от данного фреймворка, необходимо потратить время на изучение всех его возможностей.

**2.2 Spring Framework**

Spring Framework– универсальный фреймворк с открытым исходным кодом для Java-платформы. Также существует для платформы .NET Framework, названный Spring.NET. Несмотря на то, что Spring не обеспечивал какую-либо конкретную модель программирования, он стал широко распространённым в Java-сообществе главным образом как альтернатива и замена модели Enterprise JavaBeans. Spring предоставляет бо́льшую свободу Java-разработчикам в проектировании; кроме того, он предоставляет хорошо документированные и лёгкие в использовании средства решения проблем, возникающих при создании приложений корпоративного масштаба.

Между тем, особенности ядра Spring применимы в любом Java-приложении, и существует множество расширений и усовершенствований для построения веб-приложений на Java Enterprise платформе. По этим причинам Spring приобрёл большую популярность и признаётся разработчиками как стратегически важный фреймворк.

Spring обеспечивает решения многих задач, с которыми сталкиваются Java-разработчики и организации, которые хотят создать информационную систему, основанную на платформе Java. Из-за широкой функциональности трудно определить наиболее значимые структурные элементы, из которых он состоит. Spring не всецело связан с платформой Java Enterprise, несмотря на его масштабную интеграцию с ней, что является важной причиной его популярности.

Spring, вероятно, наиболее известен как источник расширений (features), нужных для эффективной разработки сложных бизнес-приложений вне тяжеловесных программных моделей, которые исторически были доминирующими в промышленности. Ещё одно его достоинство в том, что он ввел ранее неиспользуемые функциональные возможности в сегодняшние господствующие методы разработки, даже вне платформы Java.

Этот фреймворк предлагает последовательную модель и делает её применимой к большинству типов приложений, которые уже созданы на основе платформы Java. Считается, что Spring реализует модель разработки, основанную на лучших стандартах индустрии, и делает её доступной во многих областях Java.

Spring может быть рассмотрен как коллекция меньших фреймворков или фреймворков во фреймворке. Большинство этих фреймворков может работать независимо друг от друга, однако они обеспечивают большую функциональность при совместном их использовании. Эти фреймворки делятся на структурные элементы типовых комплексных приложений:

* Inversion of Control-контейнер: конфигурирование компонентов приложений и управление жизненным циклом Java-объектов.
* Фреймворк аспектно-ориентированного программирования: работает с функциональностью, которая не может быть реализована возможностями объектно-ориентированного программирования на Java без потерь.
* Фреймворк доступа к данным: работает с системами управления реляционными базами данных на Java-платформе, используя JDBC- и ORM-средства и обеспечивая решения задач, которые повторяются в большом числе Java-based environments.
* Фреймворк управления транзакциями: координация различных API управления транзакциями и инструментарий настраиваемого управления транзакциями для объектов Java.
* Фреймворк MVC: каркас, основанный на HTTP и сервлетах, предоставляющий множество возможностей для расширения и настройки (customization).
* Фреймворк удалённого доступа: конфигурируемая передача Java-объектов через сеть в стиле RPC, поддерживающая RMI, CORBA, HTTP-based протоколы, включая web-сервисы (SOAP).
* Фреймворк аутентификации и авторизации: конфигурируемый инструментарий процессов аутентификации и авторизации, поддерживающий много популярных и ставших индустриальными стандартами протоколов, инструментов, практик через дочерний проект Spring Security (ранее известный как Acegi).
* Фреймворк удалённого управления: конфигурируемое представление и управление Java-объектами для локальной или удалённой конфигурации с помощью JMX.
* Фреймворк работы с сообщениями: конфигурируемая регистрация объектов-слушателей сообщений для прозрачной обработки сообщений из очереди сообщений с помощью JMS, улучшенная отправка сообщений по стандарту JMS API.
* Тестирование: каркас, поддерживающий классы для написания модульных и интеграционных тестов.

**2.3 Spring MVC**

Spring имеет собственную MVC-платформу веб-приложений, которая не была первоначально запланирована. Разработчики Spring решили написать её как реакцию на то, что они восприняли как неудачность конструкции (тогда) популярного Apache Struts, а также других доступных веб-фреймворков. В частности, по их мнению, было недостаточным разделение между слоями представления и обработки запросов, а также между слоем обработки запросов и моделью.

Класс DispatcherServlet является основным контроллером фреймворка и отвечает за делегирование управления различным интерфейсам, на всех этапах выполнения HTTP-запроса. Об этих интерфейсах следует сказать более подробно.

Как и Struts, Spring MVC является фреймворком, ориентированным на запросы. В нем определены стратегические интерфейсы для всех функций современной запросно-ориентированной системы. Цель каждого интерфейса – быть простым и ясным, чтобы пользователям было легко его заново имплементировать, если они того пожелают. MVC прокладывает путь к более чистому front-end-коду. Все интерфейсы тесно связаны с Servlet API. Эта связь рассматривается некоторыми как неспособность разработчиков Spring предложить для веб-приложений абстракцию более высокого уровня. Однако эта связь оставляет особенности Servlet API доступными для разработчиков, облегчая все же работу с ним. Наиболее важные интерфейсы, определенные Spring MVC, перечислены ниже:

* HandlerMapping: выбор класса и его метода, которые должны обработать данный входящий запрос на основе любого внутреннего или внешнего для этого запроса атрибута или состояния.
* HandlerAdapter: вызов и выполнение выбранного метода обработки входящего запроса.
* Controller: включен между Моделью (Model) и Представлением (View). Управляет процессом преобразования входящих запросов в адекватные ответы. Действует как ворота, направляющие всю поступающую информацию. Переключает поток информации из модели в представление и обратно.
* View: ответственно за возвращение ответа клиенту в виде текстов и изображений. Некоторые запросы могут идти прямо во View, не заходя в Model; другие проходят через все три слоя.
* ViewResolver: выбор, какое именно View должно быть показано клиенту.
* HandlerInterceptor: перехват входящих запросов. Сопоставим, но не эквивалентен сервлет-фильтрам (использование не является обязательным и не контролируется DispatcherServlet-ом).
* LocaleResolver: получение и, возможно, сохранение локальных настроек (язык, страна, часовой пояс) пользователя.
* MultipartResolver: обеспечивает Upload — загрузку на сервер локальных файлов клиента.

Spring MVC предоставляет разработчику ясное и прозрачное разделение между слоями в MVC и запросах, стратегия интерфейсов — каждый интерфейс делает только свою часть работы, итерфейс всегда может быть заменен альтернативной реализацией, интерфейсы тесно связаны с Servlet API, высокий уровень абстракции для веб-приложений.