**Apache Wicket**

**Слайд 1. Apache Wicket** — фреймворк с открытым исходным кодом для создания веб-приложений. Разработан Джонатаном Локе в 2004 году. С июня 2007 года является проектом Apache Software Foundation.

**Фреймворк** (англ. **framework** — каркас, структура) — программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта. **Open Source:** лицензионный договор программного продукта разрешает исследовать его исходный код, при этом запрещая вносить в него изменения и/или распространять его копии. В этом случае продукт является открытым.

**Слайд 2. Подход Wicket:**

В отличие от многих фреймворков, которые управляются запросами, Wicket использует подход, основанный на повторном использовании компонентов и их сборке в веб-приложение. Состояние компонентов пользовательского интерфейса сохраняется, когда пользователь запрашивает новую страницу, и затем восстанавливается, если запрос повторяется. Каждый компонент состоит из класса Java и файла HTML. Помимо основанного на компонентах подхода, отличительной чертой Wicket является отсутствие XML-файлов конфигурации. Для общих настроек приложения, таких как идентификация главной страницы, вместо XML-файла конфигурации используется класс Java. Каждый виджет веб-приложения создается в классе Java и представляется в виде страницы HTML. Класс Java и страница HTML должны иметь одно и то же имя и находиться в одной и той же исходной структуре. Эти компоненты связаны друг с другом идентификатором Wicket.

1. ***Just Java + just HTML = Wicket.*** *Только Java и HTML* — Для настроек достаточно одного стандартного файла web.xml. Не нужно тратить время на изучение правил XML-файлов или нового языка;
2. ***Настоящее повторное использование*** — конечная страница может состоять из множества мелких блоков, объединенных в единое целое. Написанный код можно поместить в jar-файл и повторно использовать без изменений;
3. *Просто и мощно одновременно* — Wicket не загроможден множеством запутанных API;
4. *Парадигма программирования Wicket знакома разработчикам Java. Им не нужно знать ничего, кроме Java и HTML. AJAX можно использовать без знания JavaScript. Использование AJAX возможно без единой строчки на JavaScript* — Да, это возможно, потому что большинство кода JavaScript генерируются самим Wicket. Но, если вы желаете, можете внедрить JavaScript и вообще всё, что угодно. Ну, или почти всё.

**API** (интерфейс программирования приложений, интерфейс прикладного программирования) (англ. application programming interface, API [эй-пи-ай]) — набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) для использования во внешних программных продуктах. Используется программистами при написании всевозможных приложений.

**Слайд 3. Преимущества:**

* **Масштабируемость:** Wicket очень хорошо масштабируется. Если вы работаете в команде из 2 или из 20 человек, вы будете иметь все возможности для разделения функциональности до мельчайших кусочков. В зависимости от потребностей вашего проекта и ресурсов, вы можете разделить труд разработчиков до целых страниц, (многоразовых) панелей, или даже более специализированных компонентов.
* **Возможность максимально удобного распределения работы между Web-дизайнерами и разработчиками:** Из-за того, что Wicket использует обычные HTML-страницы для описания внешнего вида страниц и панелей, вы можете использовать дизайнеров и специалистов HTML/CSS для работы над внешним видом, и им не нужно знать Java или какой-либо другой язык или синтаксис.
* Wicket удобен тем, что в нём отсутствуют многочисленные XML-файлы, все настройки делаются в коде. Если требуются динамические настройки через XML, их всегда можно внедрить. Программный код полностью разделён с HTML. **Программный код состоит только из Java, а в HTML добавляются только простые wicket-теги.**

**Вывод. Зачем нам нужен Wicket?** HTTP не обеспечивает поддержку диалога между клиентом и сервером. Каждый запрос всегда рассматривается как независимый, в том смысле, что не существует зависимости с любым предыдущим запросом. Запросы не имеют данных о состоянии приложения на сервере. Но когда дело доходит до веб-приложений, мы должны заботиться о диалоге и состоянии. Язык Java хорош, для описания поведения приложения, но он не идеален для таких вещей, как разметка.

*Подключить вспомогательный* ***подключаемый модуль*** *Wicket для NetBeans состоит из следующих компонентов:*

* ***org-netbeans-modules-web-wicket.nbm.****Предоставляет функциональность Wicket, описанную в данном учебном курсе.*
* ***org-netbeans-modules-wicket-templates.nbm.****Предоставляет шаблоны файлов Wicket для создания типичных артефактов Wicket, таких как страницы и панели.*
* ***org-netbeans-modules-wicket-library.nbm.****Предоставляет файлы Wicket JAR и устанавливает их в Диспетчере библиотек IDE. Таким образом, вам не потребуется загружать дистрибутив Wicket с веб-сайта Wicket, так как все необходимые компоненты содержатся в подключаемом модуле.*

*Скачать:* [*http://plugins.netbeans.org/plugin/3586/wicket-support*](http://plugins.netbeans.org/plugin/3586/wicket-support)

*Чтобы установить вспомогательный подключаемый модуль Wicket в среде IDE, перейдите в Диспетчер расширений (выберите Сервис > Подключаемые модули в главном меню), затем перейдите на вкладку 'Загруженные' и установите модуль.*

***Создание приложения Wicket с нуля:***

*При создании нового веб-приложения в среде IDE на последней панели мастера веб-приложений вы найдете ряд настроек, которые будут очень полезны в контексте приложения Wicket.*

1. *Выберите команду "Файл" > "Новый проект". В области "Категории" выберите "Web". В области "Projects" выберите "Web Application". Нажмите 'Далее'.*
2. *На панели 'Платформы' выберите Wicket.*

**Слайд 4. Рассмотрение исходной структуры приложения, созданного на платформе Wicket.**

Мастер создания веб-приложений IDE создал множество файлов. Такая структура не является обязательной. В Wicket-приложении, каждая страница состоит из файла HTML-разметки и сопутствующего Java-класса. Оба файла должны находиться в одной папке пакета.

**Веб-дескриптор.** Начнем с файла web.xml. Это обычный дескриптор развертывания, общий для всех веб-приложений, соответствующих серверной спецификации. Для того, чтобы наше приложение запустилось, нужно добавить следующие настройки:

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://java.sun.com/xml/ns/javaee

http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd">

<filter>

<filter-name>WicketApplication</filter-name>

<filter-class>org.apache.wicket.protocol.http.WicketFilter</filter-class>

<init-param>

<param-name>applicationClassName</param-name>

<param-value>com.myapp.wicket.Application</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>WicketApplication</filter-name>

<url-pattern>/wicket/\*</url-pattern>

</filter-mapping>

**Класс приложения Wicket - Application.java.** Этот файл Java содержит общие настройки приложения. **Метод getHomePage()** - минимальное требование для общего класса приложения. Он указывает первую (главную) страницу, которая отобразится после развертывания приложения.

**Главная страница Wicket - HomePage.java.** - добавляет метку на главную страницу. Представление виджетов Wicket, созданных в этом файле, осуществляется в файле с таким же именем - **HomePage.html** и в той же исходной структуре.

Класс добавляет виджет «Label» на главную страницу с двумя параметрами: идентификатор (ключевое слово), по которому Wicket определяет, куда добавить текст, и непосредственно содержимое.

super(); - инициализирует родительский класс

**Идентификатор.** Каждый виджет веб-приложения создается в классе Java и представляется в виде страницы HTML. Класс Java и страница HTML должны иметь одно и то же имя и находиться в одной и той же исходной структуре. Эти компоненты связаны друг с другом идентификатором Wicket.

Файл **HomePage.html** содержит атрибут **wicket:id**, который указывает на то, что это заместитель определенного объекта, созданного файлом Java.

**<wicket:extend>** - указывает, какая часть разметки будет использована для расширения разметки базовой страницы.

В файле HomePage.java расширяется элемент **BasePage.** Это класс, который будут расширять наши веб-страницы. Каждый класс, расширяющий BasePage, наследует экземпляр HeaderPanel и FooterPanel. Это нужно для того, чтобы у всех веб-страниц были одинаковые верхние и нижние колонтитулы.

**Файл BasePage.html** определяет общую структуру страницы, а BasePage.java определяет функционал и сами данные, которые размещаются. Также здесь есть ссылка на таблицу стилей CSS, сгенерированную средой IDE. Эта таблица находится в папке 'Веб-страницы' в окне 'Проекты'.

Интерес представляет специальный тег **<wicket:child/>**, при помощи которого указывается, каким образом дочерние страницы будут расширять разметку **BasePage.html**.

**<section>.** Задаёт раздел документа, может применяться для блока новостей, контактной информации, глав текста, вкладок в диалоговом окне и др. Раздел обычно содержит заголовок. Допускается вкладывать один тег <section> внутрь другого.

**Class.** Задает стилевой класс, который позволяет связать определенный тег со стилевым оформлением. В значении допускается указывать сразу несколько классов, разделяя их между собой пробелом.

**HeaderPanel.html, FooterPanel.html.** Данные файлы являются полноценной HTML-страницей с тегами <html> и <body>. Однако для отображения компонента будет использована только та часть разметки, которая находится внутри специального тега <wicket:panel>.

**Добавление виджета.**

Как и большинство других артефактов Wicket, виджет имеет два представления: представление Java и представление HTML. В представлении Java виджет создается. В представлении HTML он отображается. Как упоминалось ранее, для перехода от одного представления к другому можно использовать гиперссылку.

Откройте файл HomePage.html. Если палитра не открывается автоматически, выберите Окно > Палитра (Ctrl-Shift-8).

**Добавление повторно используемого компонента.**

Одно из преимуществ Wicket заключается в повторно используемых компонентах. В этом разделе показано, как с помощью матера создать панель, которая снова имеет два представления: представление Java и представление HTML. Панель будет создана таким образом, чтобы можно было повторно использовать баннер на веб-страницах и обеспечить одинаковый вид баннера на всем веб-сайте. Добавить панель на веб-страницу совсем не сложно.

Тег <span> предназначен для определения строчных элементов документа. В отличие от блочных элементов, таких как <table>, <p> или <div>, с помощью тега <span> можно выделить часть информации внутри других тегов и установить для нее свой стиль. Например, внутри абзаца (тега <p>) можно изменить цвет и размер первой буквы, если добавить начальный и конечный тег <span> и определить для него стиль текста. Чтобы не описывать каждый раз стиль внутри тега, можно выделить стиль во внешнюю таблицу стилей, а для тега добавить атрибут class или id с именем селектора.

**Добавление компонентов AJAX**

Вместо использования JavaScript для добавления в приложение Wicket асинхронных веб-компонентов (по технологии AJAX) можно использовать модель компонента Java со встроенными компонентами AJAX.

В представлении HTML класса BannerPanel перетащите элемент AJAX Text Input из палитры (Ctrl-Shift-8).

**Слайд 5. Stateless и Statefull страницы в Wicket**

**Страницы с состоянием** - это те страницы, которые, полагаясь на пользовательскую сессию, сохраняют своё внутреннее состояние. Противоположно, **страницы без состояния** - те, которые не изменяют своё внутреннее состояние во время жизненного цикла; не имеют необходимости занимать место в пользовательской сессии.

С точки зрения Wicket, самое большое их отличие в том, что страницам с состоянием присваивается версия, то есть каждый раз, когда внутреннее состояние страницы изменяется, ее версия сохраняется в сессии.

Stateless страницы не хранят никакого состояния из-за чего нет необходимости сохранять их от запроса к запросу.

Такие страницы не сохраняются ни в сессии, ни в истории посещённых страниц хранящейся на диске, таким страницам не присваивается версия, как это происходит со statefull страницами. При каждом запросе к такой странице, её экземпляр создаётся заново.

Если мы хотим узнать тип страницы, мы можем вызвать метод **isPageStateless ()** класса Page.

Для того, чтобы построить stateless-страницу мы должны соблюсти некоторые правила, чтобы быть уверенными, что страница не будет использоваться в сессии.

Stateful страницам присваивается версия для того, чтобы поддержать кнопку "Назад" браузера: когда эта кнопка нажата Wicket должен ответить, построив экземпляр той же страницы, которая использовалась ранее.

Новая версия страницы создается, когда stateful страницу запрашивают впервые или когда существующий экземпляр изменяется (например, изменение иерархии компонентов). Для определения каждой версии Wicket использует идентификатор, называемый id страницы. Это уникальный номер, он увеличивается каждый, когда создается новая версия страницы.

**Для того что бы страница стала stateless необходимо выполнить два условия:**

* **В первую очередь страница должна быть bookmarkable.**
* **Страница должна использовать только stateless компоненты.**

При выполнении этих условий, такая страница автоматически становится stateless. Но если на странице используется хотя бы один компонент, который не является stateless, то вся страница становится statefull, и полностью сереализуется и сохраняется в сессии, и далее в истории на диске.  
 **Рассмотрим подробнее эти два условия:**  
  
Bookmarkable страница означает, что этой странице можно самостоятельно назначить URL, и этот URL не содержит ни какой информации связанной с сессией (к примеру версии страницы), и её можно открыть непосредственно набрав адрес в браузере, даже если сессия ещё не начата.

Для того, что бы страница была Bookmarkable, она должна иметь конструктор не принимающий параметров или конструктор принимающий единственный параметр класса PageParameter.  
В случае если страница создаётся с помощью конструктора не удовлетворяющего этим условиям, страница не будет Bookmarkable.

Если страница не является Bookmarkable, то к ней можно получить доступ только создав её из другой страницы.  
  
Является компонент stateless или statefull определяется с помощью метода getStatelessHint(), который возвращает true если компонент stateless и false если нет.

**Слайд 6.**  
К stateless относятся следующие компоненты:

* Label, MultiLineLabel, EnumLabel
* Panel, Border, BoxBorder, Include, TabbedPanel, FeedbackPanel
* BookmarkablePageLink, ExternalLink, ImageMap, StatelessLink
* StatelessForm
* Button, SubmitLink
* TextField, PasswordTextField, TextArea, HiddenField, RequiredTextField, DateTextField
* CheckBoxMultipleChoice, CheckGroupSelector
* ListMultipleChoice
* Select, SelectOption, SelectOptions
* Palette
* DataGridView, DataTable, Tree, TreeTable,
* ListItem, ListView, Loop, PageableListView, PropertyListView
* PagingNavigation, PagingNavigator
* BaseTree, LabelTree, LinkTree

К statefull относятся следующие компоненты:

* Link, ResourceLink
* Form
* CheckBox, CheckGroup, Check
* DropDownChoice, ListChoice
* Radio, RadioChoice, RadioGroup
* ImageButton
* PagingNavigationIncrementLink, PagingNavigationLink
* AJAX компоненты (существует библиотека, с некоторыми stateless AJAX компонентами [github.com/jolira/wicket-stateless](https://github.com/jolira/wicket-stateless))

**Важно:** Link и Form являются statefull. Для использования этих компонентов в stateless страницах предназначены StatelessLink и StatelessForm.

**Слайд 6. Ссылки.**

1. Обработчик нажатия.

Подключаем компонент org.apache.wicket.markup.html.link.Link.

Вызываем метод onClick ().

add(new Link("id") {

@Override

public void onClick() {

setResponsePage(SecondPage.class, pageParameters);

}

});

Если мы хотим провести к другой странице, мы должны использовать метод setResponsePage(): метод либо создает объект нового класса, таким образом, экземпляр класса новой страницы будет создаваться каждый раз, когда мы нажимаем на ссылку, либо вызывает ранее созданный.

1. Автоматическое создание ссылок с тегом <wicket-link>.

import org.apache.wicket.markup.html.link.Link;

<!--Ссылка без Java-кода (автоматическое создание ссылок с тегом wicket:link) -->

<wicket:link>

<a href="SecondPage.html">Ссылка на вторую страницу №1</a><br/>

</wicket:link>

Ссылка страницы может быть создана непосредственно в файле разметки без написания Java-кода.

**Атрибут href**. Задает адрес документа, на который следует перейти (адрес файла в package).

Этот тег также может быть использован для управления веб-ресурсами, такими как: картинки, CSS и JavaScript файлы и т. д.

1. Ссылка на внешний ресурс.

Используем компонент org.apache.wicket.markup.html.link.ExternalLink.

String example = "фреймворк Wicket";

String googleQuery = "http://www.google.com/search?q=" + example.replace(" ", "+");

add(new ExternalLink("externalSite", googleQuery));

<!--Ссылка на внешний ресурс -->

<a wicket:id="externalSite">Ищите нас в Google!</a>

**Слайд 8.Генерация адресов.**

1. **PageParameters (индексированные и именные)**

Бывают именные(named) и индексированные (Indexed).

Индексированные: отображаются как сегменты URL, расположенные перед именными параметрами.

PageParameters pageParameters = new PageParameters();

pageParameters.add("name", "SecondPage");

pageParameters.set(0, "zero");

pageParameters.set(1, "one");

Если мы определим параметры страницы, то они будут использованы для построения URL.

add(new Link("id") {

@Override

public void onClick() {

setResponsePage(SecondPage.class, pageParameters);

}

});

1. **Монтирование страницы.**

**Init() –** этот метод вызывается, когда наше приложение загружается веб-сервером, в нем размещаются код с настройками приложения.

@Override

public void init()

{

super.init();

//mountPage("/pageMount", HomePage.class);

mountPage("/pageMountt", SecondPage.class);

}

**С параметрами:**

mountPage("/pageMount/${foo}/otherSegm", MountedPageWithPlaceholder.class);

**Пакет:**

mountPackage("/mountPackage", StatefulPackageMount.class);

**Подключение CSS:**

/\*@Override

public void renderHead(IHeaderResponse response) {

//response.render(CssReferenceHeaderItem.forReference(new PackageResourceReference(this.getClass(), "style.css")));

PackageResourceReference cssFile = new PackageResourceReference(this.getClass(), "style.css");

CssHeaderItem cssItem = CssHeaderItem.forReference(cssFile);

response.render(cssItem);

}\*/