Container

Application framework

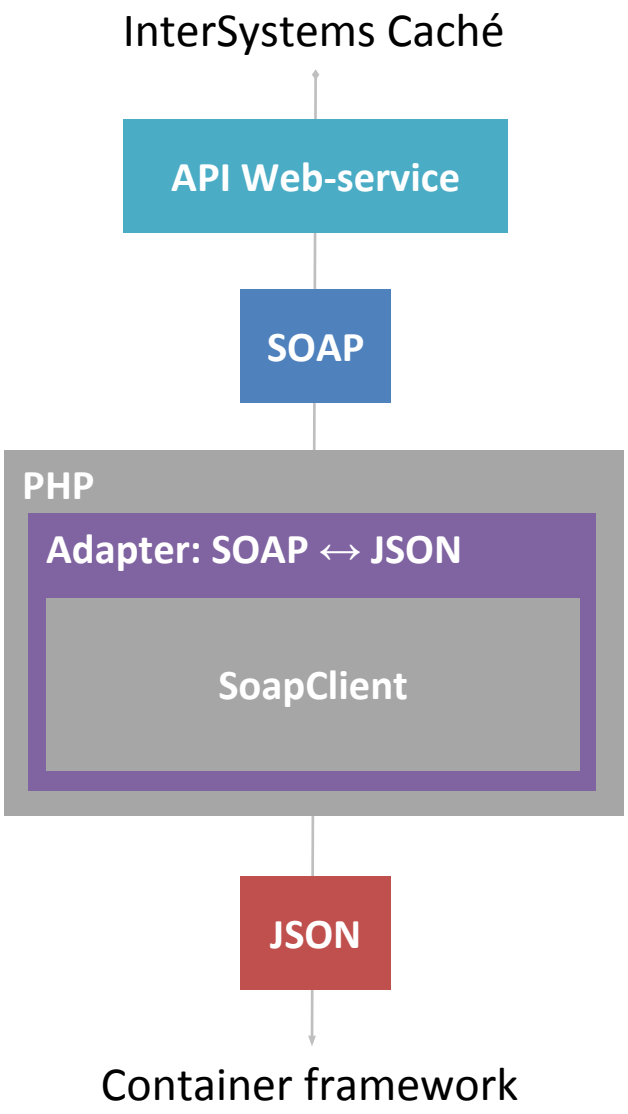
Container – это фреймворк для создания «тонких клиентов» взаимодействующий по протоколу SOAP с API веб-сервисами приложения. Обладающий богатой функциональность для взаимодействия с веб-сервисами и построения на основе json-данных кроссбраузернного и кроссплатформенного веб-интерфейса.

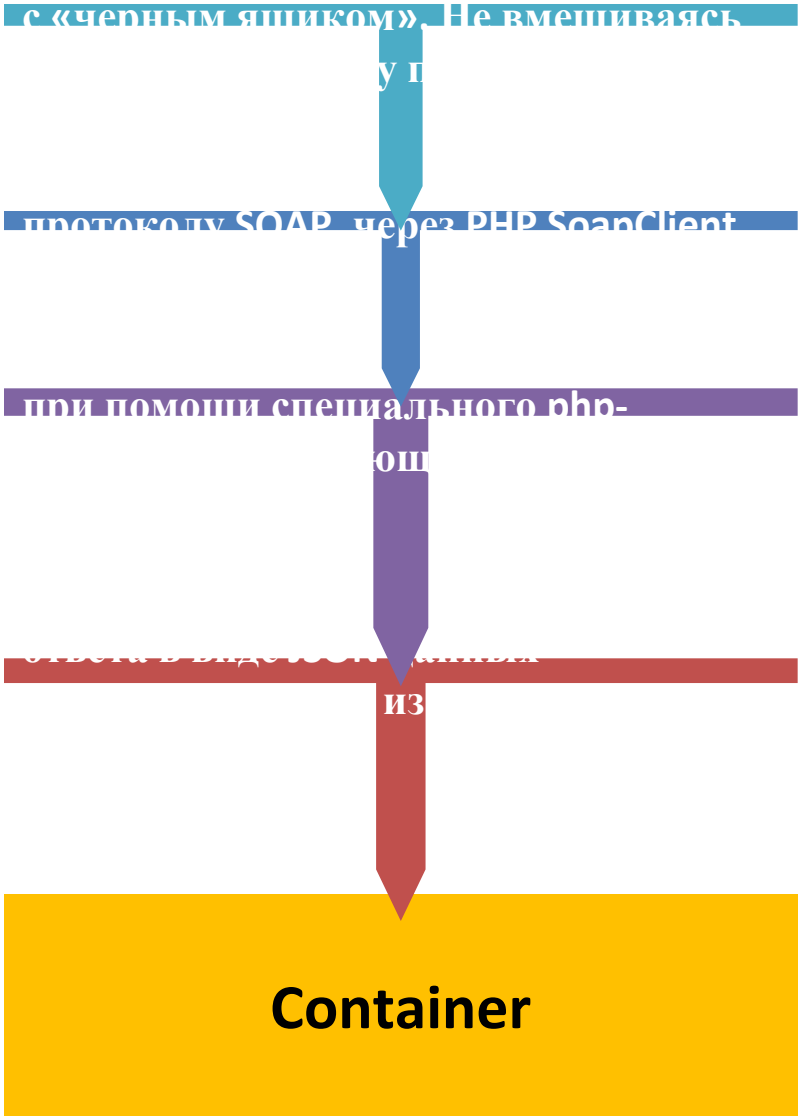
Ключевые возможности и преимущества

- Высокая скорость разработки
- Независимость от серверной платформы
- Независимость от DB как источника данных
- Настраиваемая модель маршрутизации и отчуждаемые ссылки
- Повсеместное использование **data-*** атрибутов
- JS-модули реализованы как **jquery**-плагины и практически все отчуждаемы
- Все преимущества **Twitter Bootstrap** и **lesscss** на уровне ядра
- Компоненты «из коробки»: автодополнение, выбор даты, валидация, модальные окна и многие другие.

Принцип работы

Трансляция данных Caché → Container





Свободный выбор платформы для разработки

Источником данных может выступать приложение созданное на любой платформе. Главным условием является наличие **API Web Service** (веб-сервисы).
Со стороны тонкого клиента, так же может использоваться любая платформа на которой есть средства трансляции данных веб-сервисов в JSON.
На данный момент в Container используется **php**, который можно заменить, на другой*.



& etc

* В ближайших планах создания модуля на **python**.

JSON-объект ответа веб-сервиса

```
{

   Result: //данные
   Error:  //true|false
   Status:{

      Code: //1|0
      Text: //описание статуса

   }

   SoapXml:{ //Отладочные soap-
   данные

      Headers:  //HTTP заголовки
      Request:  //xml запроса
      Response: //xml ответа

   }

}
```

Данные пришедшие с сервера приложения в ответ на запрос веб-метода. По умолчанию его значения null на, что и происходит проверка.

Принимает значения true или false в зависимости от того произошла ошибка или нет. Ошибкой является как SoapFalut так и Status.Code = 0.

Статус выполнения веб-метода содержит Code который принимает значения 1, если все хорошо и 0, если “не очень”, а так же Text – расшифровка статуса в «человеко-понятном виде».

Используется для отладки и приходит в запросе только в режиме этом режиме.

Построение JS-интерфейса

Основные js-модули

Requester
Формирование и выполнение ajax-запроса к серверу приложения. При получении ответа обработка ошибок и выполнение Callback-функцией, по умолчанию пытается в ответе найти Контейнеры, и выполнить процедуру построения интерфейса.

Container
Получает JSON-данные из **Requester** и строит на их основе интерфейс приложения. Работает рекурсивно строя DOM-дерево на основе иерархии вложенности(ChildrenList) контейнеров и учитывая правила **Twitter Bootstrap Scaffolding**. Если вместо вложенных Контейнеров присутствует список Контролов(ControlList), то для их обработка происходит в **Genform**.

Genform(Control)
Формирует на основе JSON-данных элементы интерфейса - Контролы. Определяя правила взаимодействия их с пользователем и отображения.

Контейнеры

Логические блоки страницы, например: сайдбар, шапка сайта, тело страницы, модальное окно. Внутри контейнера может содержаться, либо иерархия таких же контейнеров (согласно вложенности JSON-данных), либо контролы.

Основные параметры описания:

- **ClientId**

Присваивается как атрибут id html-элемента контейнера

- **ParamList**

Список параметров исходя из которых определяются правила поведения и отображения контейнера.

- **TemplateStructure**

Определяет специфику построения и в ряде случаев поведения дочерних контролов и контейнеров.

- **Number**

Парамет сортировки

Контролы

Функциональные элементы, например кнопка, ссылка или чистый html-код. У каждого контрола есть параметр — тип(**Type**), определяющий правила генерации элемента, если их нет, то создается html-элемент с соответствующим именем.

Основные параметры описания:

- **ServerId**

Используется элементами формы для идентификации параметра как значения атрибута name html-элемента контрола.

- **ParamList**

Список параметров исходя из которых определяются правила поведения и отображения контролов.

- **AttrList**

Список параметров напрямую транслируемый в атрибуты html-элемента.

- **Number**

Парамет сортировки

- **Type**

Определят тип контрола. По умолчанию, если нет специфических правил, то создается html-элемент с таким именем.