Mindtree
Welcome to possible

Servlet Filter and Listener

Objectives

- Servlet Filters and Listener
- What is use of Servlet filters and Servlet context listeners?
- How to use data converts, listeners and validators in Servlet?

Servlet Filters

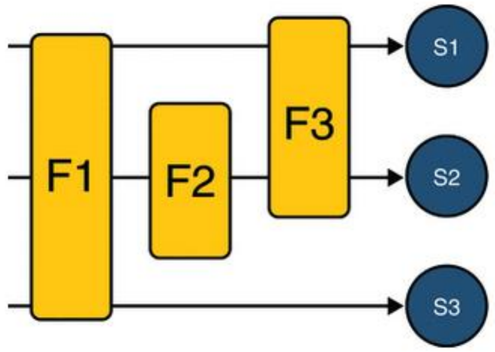
- A filter is a component that can intercept a request targeted for a servlet, JSP page, or static page, as well as the response before it's sent to the client.
- Filters makes it easy to centralize tasks that apply to all requests, such as access control, logging, and charging for the content or the services offered by the application.
- A filter has full access to the body and headers of the request and response, so it can also perform various transformations.
- You can configure a filter to act on a servlet or group of servlets.
- Zero or more filters can filter one or more servlets.

Servlet Filters

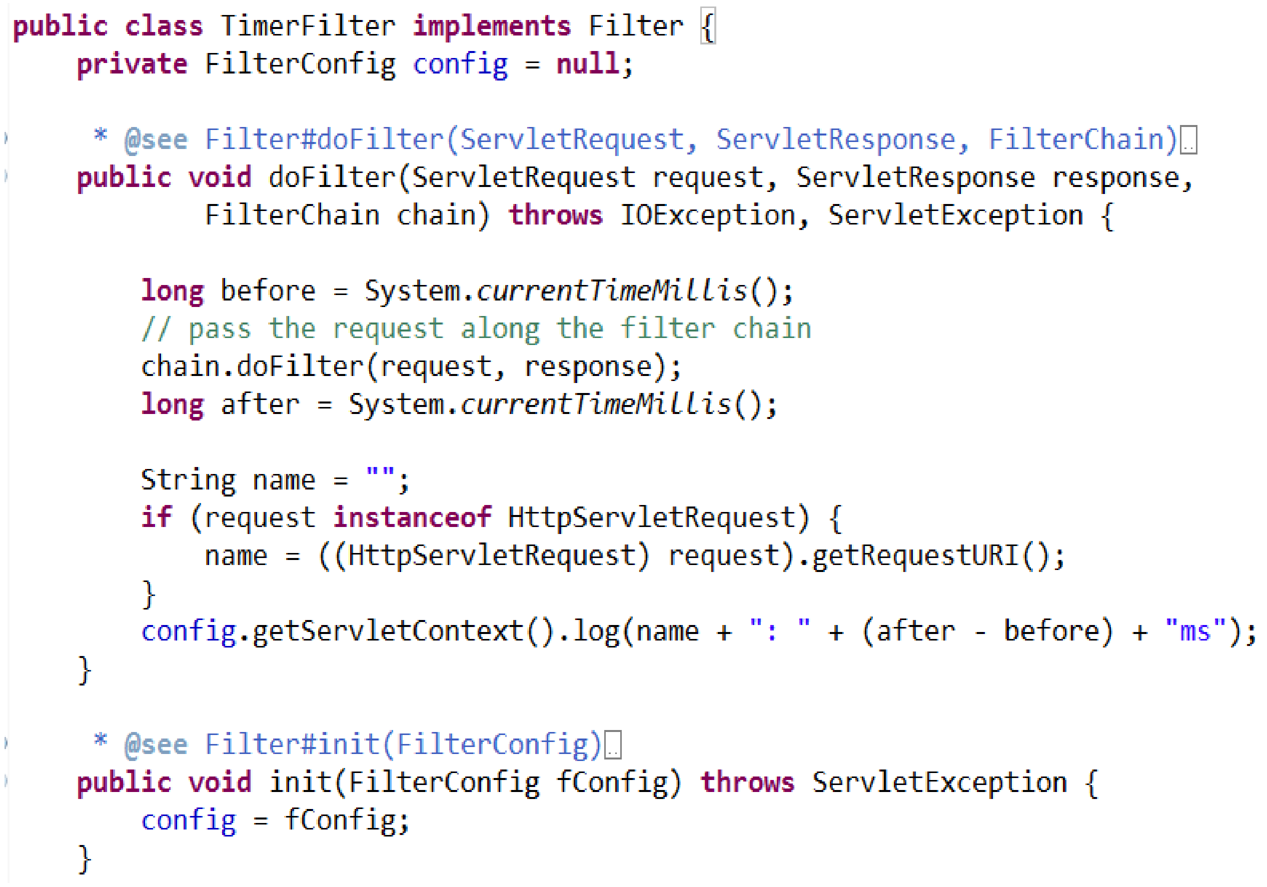
- A filter implements javax.servlet.Filter and defines its three methods:
  - void init(FilterConfig config) throws ServletException:
    - Called before the filter goes into service, and sets the filter's configuration object
  - void destroy()
    - Called after the filter has been taken out of service
  - void doFilter(ServletRequest req, ServletResponse res, FilterChain chain) throws IOException, ServletException
    - Performs the actual filtering work

Filter-to-Servlet Mapping

- Filter F1 is mapped to servlets S1, S2, and S3, filter F2 is mapped to servlet S2, and filter F3 is mapped to servlets S1 and S2.



Filter Example



```java
public class TimerFilter implements Filter {
    private FilterConfig config = null;

     * @see Filter#doFilter(ServletRequest, ServletResponse, FilterChain)
    public void doFilter(ServletRequest request, ServletResponse response,
            FilterChain chain) throws IOException, ServletException {

        long before = System.currentTimeMillis();
        // pass the request along the filter chain
        chain.doFilter(request, response);
        long after = System.currentTimeMillis();

        String name = "";
        if (request instanceof HttpServletRequest) {
            name = ((HttpServletRequest) request).getRequestURI();
        }
        config.getServletContext().log(name + ": " + (after - before) + "ms");
    }

     * @see Filter#init(FilterConfig)
    public void init(FilterConfig fConfig) throws ServletException {
        config = fConfig;
    }
```

When the server calls init(), the filter saves a reference to the fconfig in its config variable, which is later used in the doFilter() method to retrieve the ServletContext.

When the server calls doFilter(), the filter times how long the request handling takes and logs the time once processing has completed.

Filter Configure in web.xml

```xml
<filter>
  <display-name>TimerFilter</display-name>
  <filter-name>TimerFilter</filter-name>
  <filter-class>com.mindtree.web.TimerFilter</filter-class>
</filter>

<filter-mapping>
  <filter-name>TimerFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

- This configures the filter to operate on all requests to the server (static or dynamic), just what we want for our timing filter.
- If you connect to a page, the log output might look like this:

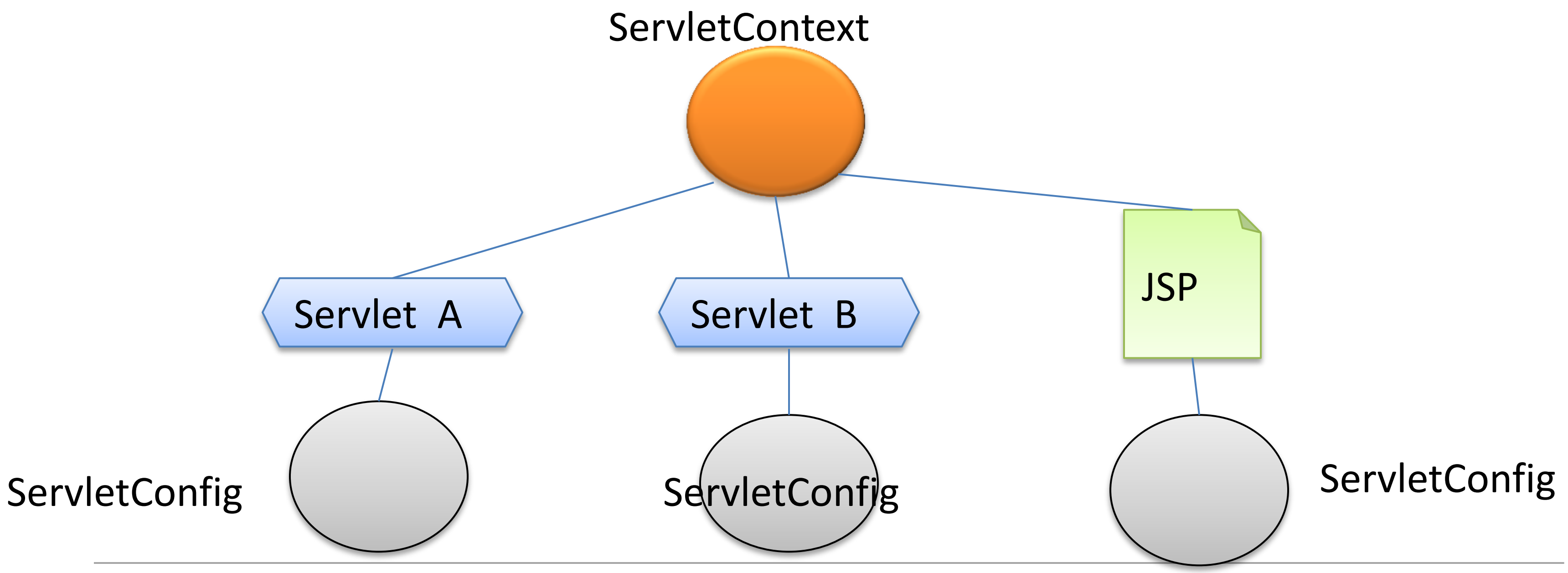
```
2014-05-25 10:14:11 /filterExample/index.html: 10ms
```

Servlet Listeners

- Listeners allow your application to react to certain events.
- Starting with Version 2.3 of the servlet specification, there are listener types for
  - servlet context
  - session and request lifecycle events ("created" and "destroyed" events)
  - session attribute events ("added" and "removed" events)
  - session activation and passivation events (used by a container that temporarily saves session state to disk or migrates a session to another server).
- All these listener types follow the standard Java event model.
- The interfaces define methods that correspond to events.
- The listener class is registered with the container when the application starts, and the container then calls the event methods at the appropriate times.

ServletContext

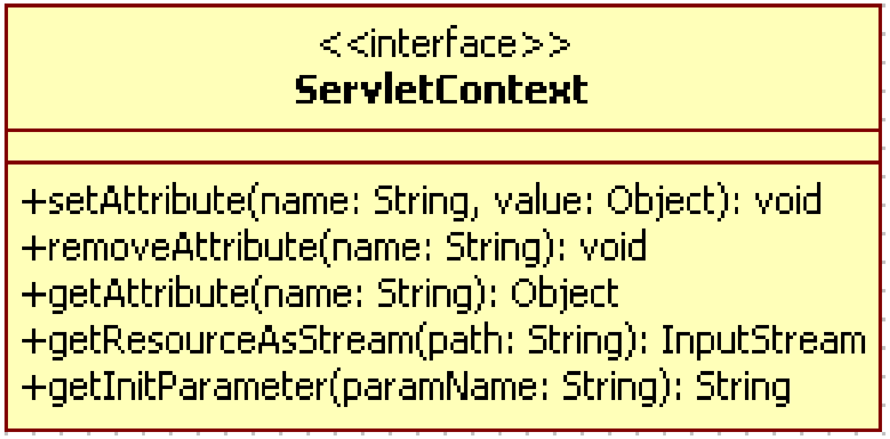
- ServletContext encapsulates a web application.
- ServletContext is created one per Web Application.
- Two main uses of ServletContext are:
  - Sharing information between Servlet's and users .
  - Accessing passive server resources like configuration files present in web application.



Mindtree

ServletContext

- void **setAttribute**(String name, Object object)
  Binds an object to a given attribute name in this ServletContext.
- void **removeAttribute**(String name)
  Removes the attribute with the given name from this ServletContext.
- Object **getAttribute**(String name)
  Returns the servlet container attribute with the given name, or null if there is no attribute by that name.
- java.io.InputStream **getResourceAsStream**(String path)
  Returns the resource located at the named path as an InputStream object.



```
+setAttribute(name: String, value: Object): void
+removeAttribute(name: String): void
+getAttribute(name: String): Object
+getResourceAsStream(path: String): InputStream
+getInitParameter(paramName: String): String
```

- String getInitParameter(String name)
  Returns a String containing the value of the named context-wide initialization parameter, or null if the parameter does not exist.

Mindtree

Code Snippet: Reading context parameters

The context – parameters configured in web.xml file

```
<context-param>
  <description>Background color for all servlets</description>
  <param-name>backgroundColor</param-name>
  <param-value>lavender</param-value>
</context-param>
```

Servlet code to read Context parameters

```
private String bgColor;

public void init() {
    /*
     * Get Servlet Context object instantied by the container.
     * Web Container creates one ServletContext object per application.
     */
    ServletContext context = getServletContext();
    /*
     * Read Context initialization parameter
     */
    bgColor = context.getInitParameter("backgroundColor");
}
```

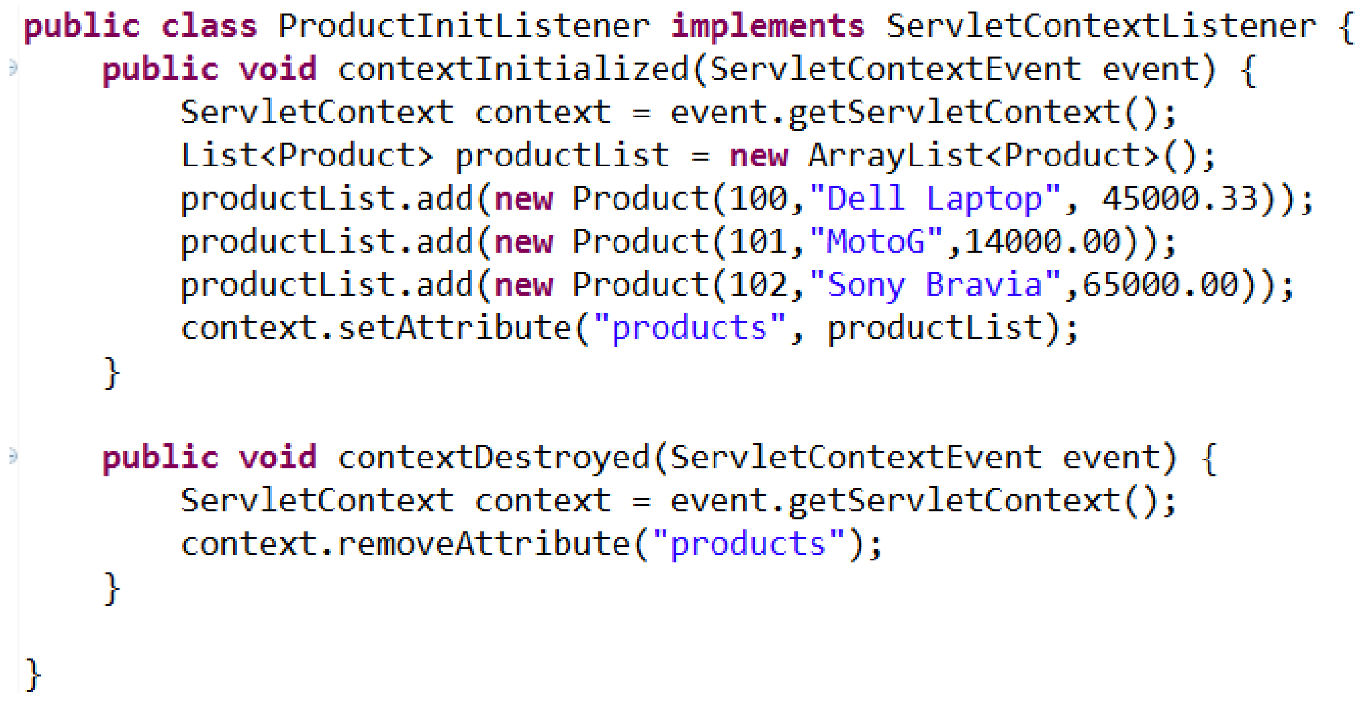
ServletContextListener

- Implementations of ServletContextListener interface receive notifications about changes to the servlet context of the web application they are part of.
- To receive notification events, the implementation class must be configured in the deployment descriptor for the web application.
- All ServletContextListener's are notified of context initialization before any filter or servlet in the web application is initialized and after all servlets and filters have been destroyed.

```
package javax.servlet;

public interface ServletContextListener extends java.util.EventListener {

        public void contextDestroyed(ServletContextEvent sce);
        public void contextInitialized(ServletContextEvent sce);

}
```

ServletContextListener example

- Example illustrates storing list of products in ServletContext. This collection can be accessed by different resources of the web application

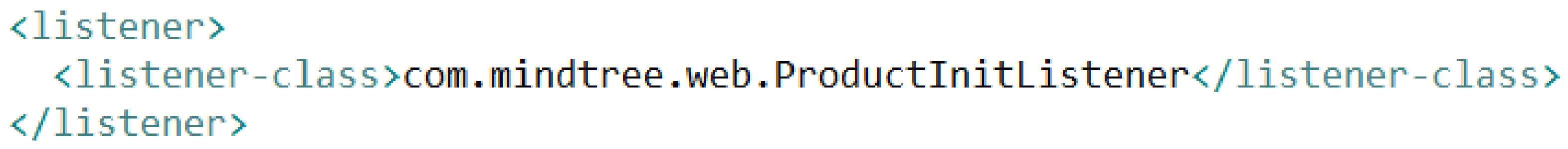


```java
public class ProductInitListener implements ServletContextListener {
    public void contextInitialized(ServletContextEvent event) {
        ServletContext context = event.getServletContext();
        List<Product> productList = new ArrayList<Product>();
        productList.add(new Product(100,"Dell Laptop", 45000.33));
        productList.add(new Product(101,"MotoG",14000.00));
        productList.add(new Product(102,"Sony Bravia",65000.00));
        context.setAttribute("products", productList);
    }

    public void contextDestroyed(ServletContextEvent event) {
        ServletContext context = event.getServletContext();
        context.removeAttribute("products");
    }

}
```

Mindtree

ServletContextListener example

- Entry in web.xml
  - Unlike Servlet and Filter Listeners do not have URL pattern, because they are invoked by events occurring within the container



```xml
<listener>
  <listener-class>com.mindtree.web.ProductInitListener</listener-class>
</listener>
```

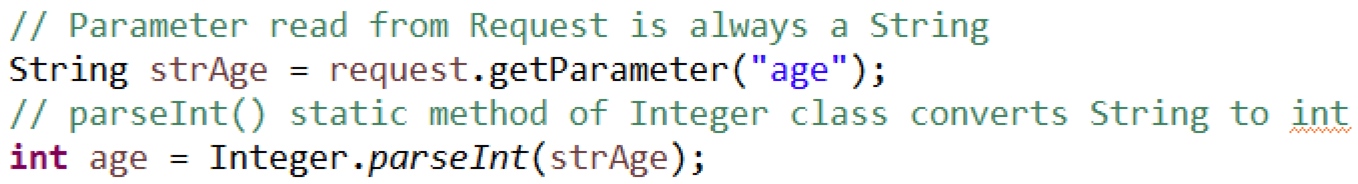
Processing Servlet request

- The getParameter( ) method
  - public java.lang.String getParameter(java.lang.String name)
  - Returns the value of a request parameter as a String, or null if the parameter does not exist. Request parameters are extra information sent with the request. For HTTP servlets, parameters are contained in the query string or posted form data.
  - You should only use this method when you are sure the parameter has only one value.
  - If the parameter might have more than one value, use getParameterValues(java.lang.String).

- Parameters:
  - name - a String specifying the name of the parameter
- Returns:
  - a String representing the single value of the parameter

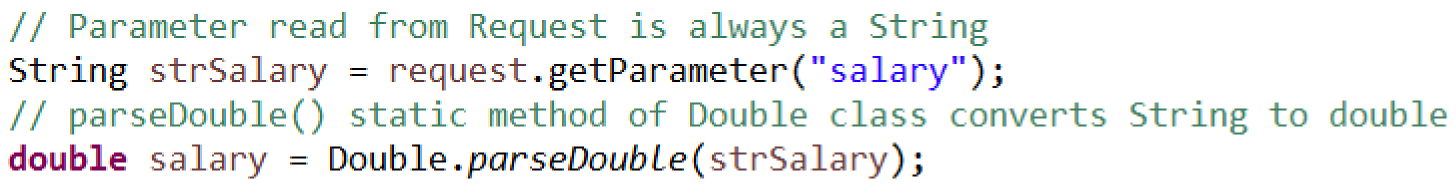
Processing Servlet request

- Converting String to int type



```
// Parameter read from Request is always a String
String strAge = request.getParameter("age");
// parseInt() static method of Integer class converts String to int
int age = Integer.parseInt(strAge);
```

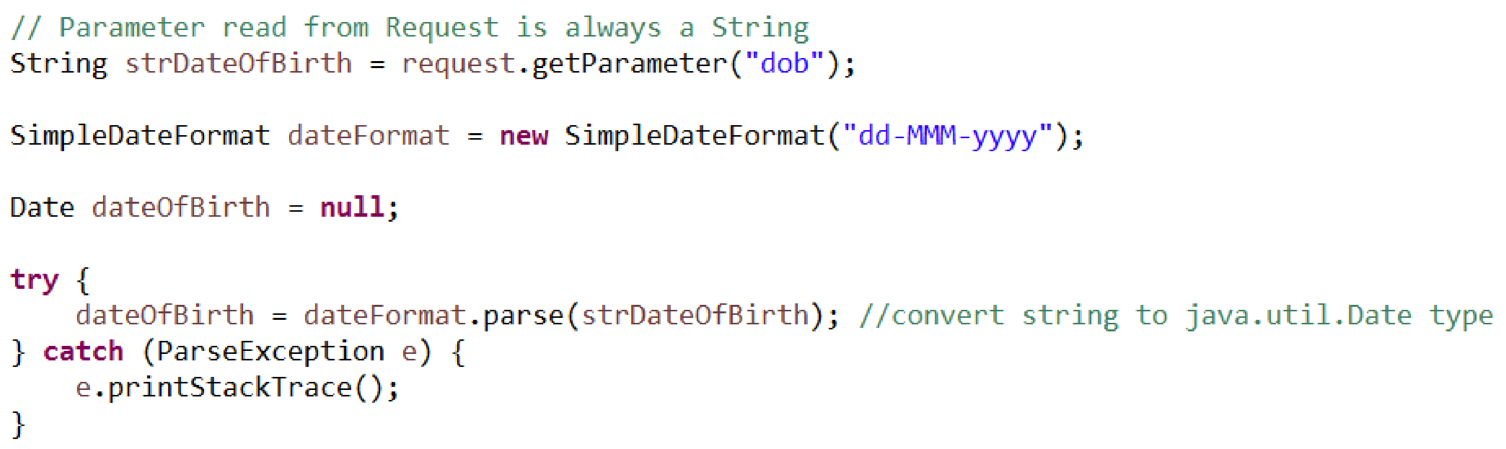
- Converting String to double type



```
// Parameter read from Request is always a String
String strSalary = request.getParameter("salary");
// parseDouble() static method of Double class converts String to double
double salary = Double.parseDouble(strSalary);
```

Processing Servlet request

- Converting String to java.util.Date
  - SimpleDateFormat is a concrete class for formatting and parsing dates in a locale-sensitive manner.
  - It allows for formatting (date -> text), parsing (text -> date)
  - SimpleDateFormat allows you to start by choosing any user-defined patterns for date-time formatting



```
// Parameter read from Request is always a String
String strDateOfBirth = request.getParameter("dob");

SimpleDateFormat dateFormat = new SimpleDateFormat("dd-MMM-yyyy");

Date dateOfBirth = null;

try {
    dateOfBirth = dateFormat.parse(strDateOfBirth); //convert string to java.util.Date type
} catch (ParseException e) {
    e.printStackTrace();
}
```

Mindtree

Welcome to possible

Name

Email

www.mindtree.com/social

India | USA | UK | Germany | Sweden | Belgium | France | Switzerland | UAE | Singapore | Australia | Japan | China